IMT School for Advanced Studies, Lucca

Lucca, Italy

A Logic-Based Approach to Specify and Design Spatio-Temporal Behaviours of Complex Systems

PhD Program in Computer Science (CDSS \ CS)

XXVIII Cycle

By

Laura Nenzi

2016

The dissertation of Laura Nenzi is approved.

Program Coordinator: Prof. Rocco De Nicola, IMT School for Advanced Studies, Lucca

Supervisor: Prof. Rocco De Nicola, IMT School for Advanced Studies, Lucca

Supervisor: Prof. Luca Bortolussi, University of Trieste

Tutor: Prof. Rocco De Nicola, IMT School for Advanced Studies, Lucca

The dissertation of Laura Nenzi has been reviewed by:

Prof. Adelinde M. Uhrmacher, University of Rostock

Prof. Alexandre Donzé, University of California

IMT School for Advanced Studies, Lucca

2016

First of all, I have to thank Luca Bortolussi, who has supervised me since I started working on my Master thesis. He brought me in a fantastic travel. He passed me the passion for this job and the skills to be independent.

If, at the end of this experience, I feel a bit a computer scientist, instead of only a mathematician, the credit belongs to Michele Loreti. I can never thank him enough for all his teachings and his infinite patience and availability.

I would like to thank Rocco De Nicola, my supervisor and coordinator of the PhD Program in Computer Science, for allowing me to undertake and travel this road, for trusting me and for always supporting me.

My sincere thanks also goes to the reviewers: Adelinde M. Uhrmacher and Alexandre Donzé, for their precious comments and questions.

This thesis is also the result of many interesting collaborations and exchanges of ideas and helps. A special thanks goes to my fantastic PhD colleagues, in particular Roberta Lanciani, a polar star for me, and Marco Tinacci. I would like to thank also all the co-authors with whom I had the pleasure to work, and the members of the European project QUANTICOL, in particular Jane Hilston.

Finally, thanks to all the people supporting me and bearing with me during this adventure: mamma e papà, the rest of the family, Ivan, Margherita, Elena, and all the other friends.

Contents

List of Figures

List of Tables

Acknowledgements

Most part of this thesis has been published. In particular: Chapter 5 is based on (BBNS13; BBNS15), coauthored by Luca Bortolussi, University of Trieste, Guido Sanguinetti, University of Edinburgh, and Ezio Bartocci, Vienna University of Technology. Chapter 6 is based on (NB14), a joint work with Luca Bortolussi, and (NBC+15), coauthored by Luca Bortolussi, Michele Loreti, University of Florence, Mieke Massink, CNR-ISTI, Pisa, and Vincenzo Ciancia, CNR-ISTI, Pisa. Chapter 7 is based on (BBM+15), joint work with Luca Bortolussi, Guido Sanguinetti, Ezio Bartocci, and Dimitrios Milios, University of Edinburgh. Finally, the jSSTL Java tool, presented in Chapter 8, has been developed in collaboration with Michele Loreti and Luca Bortolussi.

Vita

December 10, 1984	Born, Venice, Italy.
October 2008 - June 2009	Visiting Student, University of Warwick, UK, Granted by the Erasmus program.
October 2006 - December 2010	Bachelor Degree in Mathematics, University of Padova, Italy.
August 2012 - October 2012	Visiting Student, School of Informatics, University of Edinburgh, UK, Granted by the Mobility program Units.
October 2010 - December 2012	Master Degree in Mathematics Final mark: 110/110 University of Trieste, Italy.
February 2013- Now	PhD Candidate, IMT Lucca, Italy.
April 2013- Now	Active member of the EU-FET project QUANTICOL (nr. 600708).
November 2014 - May 2015	Visiting Student, Saarland University, Germany, Granted by the Erasmus program.
March 2015 - Now	Research Project Fellowship Holder, IMT Lucca, Italy.

Publications

Journal papers:

1. E. Bartocci, L. Bortolussi, L. Nenzi, G. Sanguinetti, "System Design of Stochastic Models using Robustness of Temporal Properties," in *Theoretical Computer Science*, vol. 587, pp. 3-25, 2015.

Conference papers:

2. L. Nenzi, L. Bortolussi, V. Ciancia, M. Loreti, M. Massink, "Qualitative and Quantitative Monitoring of Spatio-Temporal Properties," in Proc. of *Runtime Verification 2015: The 15th International Conference on Runtime Verification*, Vienna, Austria, 2015.

3. E. Bartocci, L. Bortolussi, L. Nenzi, D. Milios, G. Sanguinetti, "Studying Emergent Behaviours in Morphogenesis using Signal Spatio-Temporal Logic," in Proc. of *HSB 2015: the 4nd Intern. Workshop on Hybrid Systems and Biology*, Madrid, Spain, 2015.

4. L. Bortolussi, L. Nenzi, "Specifying and monitoring properties of stochastic spatio-temporal systems in signal temporal logic," in Proc. of *VALUETOOLS 2014: the 8th International Conference on Performance Evaluation Methodologies and Tools*, Bratislava, Slovakia, pp. 66-73, 2014.

5. E. Bartocci, L. Bortolussi, L. Nenzi, G. Sanguinetti, "On the robustness of temporal properties for stochastic models," in Proc. of *HSB 2013: the 2nd Intern. Workshop on Hybrid Systems and Biology*, Taormina, Italy, vol. 125(1), pp. 3-19, 2013.

6. E. Bartocci, L. Bortolussi, L. Nenzi, "A temporal logic approach to modular design of synthetic biological circuits," in Proceedings of *CMSB 2013: the 11th International Conference on Computational Methods in Systems Biology*, Austria, Springer-Verlag, Lecture Notes in Computer Science, vol. 8130, pp. 164-178, 2013.

Presentations

Invited talks:

1. *"Reinforcement Learning in Quantitative Formal Methods"*, University of Trieste, Trieste, Italy, January 2016;
2. *"Qualitative and Quantitative Monitoring of Spatio-Temporal Properties"*, Saarland University, Saarbrüchen, Germany, May 2015.
3. *"A temporal logic approach to modular design of synthetic biological circuits"*, ISTI, Pisa, Italy, May 2013.

Conference and Workshop talks:

4. *"Qualitative and Quantitative Monitoring of Spatio-Temporal Properties"*, 15th International Conference on Runtime Verification,Vienna, Austria, September 2015.
5. *"Studying Emergent Behaviours in Morphogenesis using Signal Spatio-Temporal Logic"*, 4th International Workshop on Hybrid Systems and Biology,Madrid, Spain, September 2015.
6. *"Specifying and Monitoring Properties of Stochastic Spatio-Temporal Systems in Signal Temporal Logic"*, 8th International Conference on Performance Evaluation Methodologies and Tools,Bratislava, Slovakia, December 2014.
7. *" On the Robustness of Temporal Properties for Stochastic Models"*, 2nd International Workshop on Hybrid Systems and Biology, Taormina, Italy, September 2013.

Other talks:

8. *"Qualitative and Quantitative Monitoring of Spatio-Temporal Properties"*, QUANTICOL plenary meeting, Lucca, Italy, December 2015.
9. *"Specifying and Monitoring Properties of Stochastic Spatio-Temporal Systems in SSTL"*, QUANTICOL plenary meeting, Grenoble, France, February 2015.
10. *"Specifying and Monitoring Properties of Stochastic Spatio-Temporal Systems in Signal Temporal Logic"*, Lucca, Italy, November 2014.
11. *"Verification of stochastic and spatial behaviours of complex systems"*, 11th Summer School on Modelling and Verification of Parallel Processes, Nantes, France, July 2014.

12. *"SSTL: The Signal Spatio-Temporal Logic,"*, QUANTICOL scientific meeting, Lucca, Italy, June 2014.
13. *"Spatio-Temporal logics for CAS"*, Thesis Proposal, Lucca, Italy, February 2014.
14. *"Modelling bike sharing in StoKlaim"*, QUANTICOL Space Workshop, Informatics Forum, Edinburgh, October 2013.
15. *"Signal Temporal Logic: a good logic for quantitative analysis"*, QUANTICOL pre kick-off meeting, Lucca, Italy, February 2013.
16. *"A logic-based approach to determine the connection between modules and their behavioral properties"*, Informatics Forum, Edinburgh, October 2012.

Abstract

Models of *complex systems*, composed of many heterogeneous interacting components, are challenging to analyse, due to the size and complexity of the network of interactions among the individual entities. The analysis becomes even more challenging when the *spatio-temporal* aspects of the system are to be taken into account. In this thesis, we propose a framework of efficient techniques to validate and analyse the behaviour of complex systems with spatio-temporal dynamics, both in the *stochastic* and *deterministic* cases. In particular, we define *Signal Spatio-Temporal Logic* (SSTL), a spatial extension of *Signal Temporal Logic* (STL). SSTL presents two new operators: the *bounded somewhere* and the *bounded surround*, that can be used to specify metric and topological properties in a discrete space. Given an SSTL formula, we design efficient *monitoring algorithms* to check its validity and compute its *satisfaction* (robustness) score over a spatio-temporal trace. To deal with stochastic systems, we define a *stochastic* version of the *quantitative semantics* of STL that we extended later to SSTL. We then combine it with *machine learning* techniques to define efficient *parameter estimation* and *system design* procedures. The specification and validation of SSTL formulae have been implemented in a Java tool, jSSTL. Finally, the expressivity of SSTL and the efficiency of the algorithms developed in this work are showed on interested and challenging case studies, including an *epidemic spreading* model of a waterborne disease, a *pattern formation* example for *reaction-diffusion* systems and a *french flag* model of the morphogen Bicoid.

Chapter 1

Introduction

1.1 Motivation

There is an increasing interest in the introduction of smart solutions in the world around us. A huge number of computational devices is interacting in an open and changing environment, with humans and nature in the loop that form an intrinsic part of the system. For many of these systems, the spatial and temporal dimensions are strictly correlated and influence each other. This is the case of many *Collective Adaptive Systems* (CAS), like the guidance of crowd movement in emergency situations or the improvement of the performance of bike sharing systems in smart cities (FGM12), and of many *Cyber-Physical Systems*, like pacemaker devices controlling the rhythm of heart beat. Others interesting scenarios where the space plays a crucial role can be found in the biological world, e.g., the formation of patterns from biochemical processes acting at the cellular level, a process known as *morphogenesis*. Some evident examples of these patterns can be observed in the stripes of a zebra, the spots on a leopard, the filament structure of the cyanobacteria Anabaena or the square pattern of the sulfur bacteria T. rosea.

All these examples can captured under the general umbrella of *Complex systems* (FS11). Complex systems are systems that comprise a large number of heterogeneous components forming a complex interaction

network. As we have seen from the examples, they are studied in many different disciplines and their efficiency and functionality is becoming increasingly relevant in our society. In the literature, there does not exist a unique, clear and widely acknowledged definition of them (LLW12), in the following we discuss their main characteristics.

They usually consist of a *large* number of different entities. Hence, we are considering big systems. The interactions between the components are *non-linear*, which reflects in *emergent behaviours*, not directly derivable from the knowledge of the individual parts. A complex system can exhibit a *hierarchical self-organisation* under selective pressure, meaning that it can autonomously show emergent behaviour without an external control. Many of the scientific and technological challenges we are currently facing deal with the complexity arising from the non-linear interactions between a large number of heterogeneous components. Examples span from biological systems (from bacteria up in the ladder of life), to ecological interactions, to socio-technical systems (in which digital sensors and devices interact with the natural and social environment).

They are rarely completely deterministic. Often they show *stochastic* or *hybrid* dynamics. For example, molecules inside cells perform random movements (*random walk*) and the reactions among them may occur when the probability of collision is high enough. Hence, the number of molecules of each species at each time instant is therefore a random process. When the number of molecules of each species involved is large, so that many reactions happen in any small interval of time, stochastic effects tend to average out and can be neglected. However, if the concentration of the molecules (of at least some of the species) is low the stochasticity plays an important role and must be taken into account (ESS02).

Complex systems often show an *in-homogenous spatial distribution*. Many complex systems involve a large number of heterogeneous spatial entities that are located and can move in a physical space. Hence, the spatial information is crucial to properly understand their emerging dynamics. For instance, in some eukaryotic cells, the motility is realised by the spatial polarisation of small-G-protease signalling proteins (JMEK07). Epidemic spreading at the national or worldwide scale depends crucially

on the asymmetric /inhomogenuous population distribution and on the mobility patterns and available travel routes (TBP+12).

1.2 Approach

Our ability to understand, control and design complex systems requires refined mathematical and computational tools.

This thesis focuses on the design and control models of such systems, described mathematically by formalisms, such as *Ordinary Differential Equations* (ODE), Partial Differential Equations (PDE), *Continuous Time Markov Chain* (CTMC) (Dur12), patch-CTMC, and *Stochastic Hybrid System* (SHA) (BLB05). Controlling a system requires us to be able to analyse its dynamics and in particular to verify if it satisfies or not specific behaviours. Hence, we have to be able to describe such behaviours, and to monitor whether, to which extent and how robustly, they are satisfied by a system. With the design of a system, known also as the *system design problem*, we mean the capacity to drive the dynamics of the model. This often leads to the study of the parameter space of those models, with the aim of finding the values needed to show specific desired behaviours; for this reason it is also called *parameter synthesis*.

A possible way to tackle the problem is to use formal methods and in particular *logic-based* languages like *temporal logic* (TL) (Pnu77). Temporal logic is a modal logic with a well-defined syntax and semantics to specify in a precise and concise way emergent behaviours. It has specific operators, called *temporal operators*, to describe properties of time-dependent events. This means that the specific behaviour is described by a logic formula, that we call *property*. There are many kind of temporal logics, depending on the type of system that has to be analysed, e.g., with continuous or discrete state space, and on the model of time, e.g., continuous or discrete time, linear or branching time, and so on. Our systems are principally described by mathematical models for which, usually, we do not have exact result but just numerical traces derived by simulations of the model. The type of behaviours that we want to described then are well specified by logics with *future linear temporal* modalities (Hen98)

as STL, MTL and LTL (BHHK03; CDKM11); it means that the logics can encode properties about the future of paths/trajectories.

Given a model and a property, we can use then automatic techniques as *Model Checking* (BK08; CJGP99) (MC) to verify whether the property is satisfied. In case of model checking for linear temporal formulae the verification is done iteratively checking *all* possible trajectories/behaviours of the model. To deal with stochastic systems *Probabilistic Model Checking* (KNP04; BHHK03; BCHG$^+$97) (PMC) is a well-established verification technique that can be used to compute the probability that a property, expressed in temporal logic, may be satisfied by a given stochastic process. In this case, the verification is done computing the probability measure of behaviours that satisfy the property. This methodology produces the exact solution, up to a prescribed numerical error, as it operates directly on the structure of the stochastic model. Despite the success and the importance of PMC, this technique suffers serious computational limitations, either due to *state space explosion* of the systems (an exponential blow up of the state-space) or to the difficulty in analytically checking formulae expressed in specific logics, like *Metric Temporal Logic* (MTL) (BHHK03; CDKM11).

The type of systems that we are considering are very large and complex, hence the standard model checking procedures are not feasible. In this case, simulation and testing are the preferred validation methods. This is the area of the *run-time verification* (BP09), where an individual simulation trace of the system is checked against a formula, using an automatic verification procedure, called *monitoring*. This permits to directly verify properties over deterministic models, for which we have a unique solution (usually the numerical integration of the ODE system), but it is not sufficient to deal with stochastic dynamics. In this latter case, *statistical model checking* (SMC) (JCL$^+$09b; YKNP04; YS06) can be used to estimate the satisfiability distribution of a given temporal logic formula in a stochastic model, with usually a guarantee of asymptotic correctness. This technique consists of three main steps. First, the model is simulated for finitely many runs. Second, the satisfaction of the property is verified for each run (trajectory) with a monitoring procedure. Finally, statistical

analysis are used to approximate the probability distributions of the satisfaction or to test whether the simulations provide a statistical evidence for the satisfaction of the property. The idea is that the simulations of a stochastic system are produced according to the distribution defined by the system, and then they can be used to get estimates of the probability measure on executions. Another interesting consideration about the monitoring approach is that it does not necessarily need a mathematical model but just observable traces of some process, even real observations. So this procedure can be applied also to the analysis of black box systems for which we do not have a model but only information about their behaviour.

All these described techniques provide only qualitative measures of the *satisfiability* (yes/no answer). However, this notion of satisfiability may be not enough to determine the capacity of a system to maintain a particular emergent behaviour. For example, the model can be unaffected by the uncertainty of the perturbations due to its stochastic nature. A similar issue can also arise in deterministic dynamical systems, which may be subject to extrinsic noise or uncertainty in the parameters. To address this question in the deterministic case, researchers in the verification community have proposed several notions of *robustness* (DM10; FP09; RBFS08), providing suitable definitions of distance between a trajectory of a system and the behavioural property of interest, expressed in terms of a temporal logic formula. These effectively endow the logic of interest with a *quantitative semantics*, allowing us to capture not only whether a property is satisfied but also *how much* it is satisfied. A similar notion of robustness for stochastic models would clearly be desirable but, to our knowledge, was not formalised before our work. An interesting logic that comes with a quantitative semantics is *Signal Temporal Logic* (STL) (MN04; DFM13). STL is a temporal logic very suitable to specify behaviours of real-valued time series generated during the simulation of a dynamical system. It extends the dense-time semantics of *Metric Interval Temporal Logic* (AFH96) (MITL), parameterising it with a set of numerical predicates playing the role of atomic propositions.

The literature of applications of formal methods (temporal logic in

particular) to complex systems has been mainly focused on well-mixed systems, where the spatial nature of interactions is abstracted for the sake of simplicity. However, we have seen that there are many examples of complex systems where the spatial aspect is crucial to properly understand their emerging dynamics. From the point of view of modelling languages and simulation, some relevant work in *spatial modelling* has been done in recent years (BHMU11; FH14). Some work has been done also in the area of *spatial logic* (APHvB07), yet focussing more on theoretically investigation, expressivity and decidability, often in continuous space. Less attention has been placed on more practical aspects, especially in the validation procedure. In particular, model checking routines have a much more recent history. Relevant works are those on spatial logics for process algebra with locations as (NKL+07; CG00), or spatial logic for rewrite theories (BM12). In (CLLM14), *Spatial Logic for Closure Spaces* (SLCS) is proposed for a discrete and topological notion of space, based on closure spaces (Gal99). An extension of the SLCS with temporal aspects, as "snapshot" models, can be found in (CGG+15). It extends the logic with the temporal modality of the branching logic *Computation Tree Logic*, CTL. However, the algorithms to check snapshot models have high computational cost and are susceptible to state-space explosion problems because the spatial formulae have to be recomputed at every state. Furthermore, the logic does not have a stochastic and a quantitative semantics.

The only linear-time spatio-temporal logic with monitoring procedure for checking spatio-temporal properties of spatially located differential equations that we are aware of is *Spatial-Temporal Logic* (SpaTeL) (GBB14; GSC+09; HJK+15), in which spatial properties are expressed using ideas from image processing, namely *quad trees* (FB74). This allows one to capture very complex spatial structures, but at the price of a complex formulation of spatial properties, which are in practice only learned from some template image. They also use a measure of robustness as a fitness function to guide the parameter synthesis process for a deterministic reaction diffusion system using *Particle Swarm Optimisation* (PSO) algorithms. However, the authors do not consider stochastic systems and

their optimisation algorithms for parameter synthesis generally do not provide any asymptotically guarantee for reaching the global optimum.

The design of a spatial logic is strictly related to the description of *space* in which the dynamics takes place. Space can be *logical*, i.e., a set of locations, it can be *discrete*, i.e., a grid or a more general graph, or it can be *continuous*, for instance the Euclidian space. The first type of systems that we investigate in this thesis have a discrete space, structured as a weighted graph. These models describe the number of entities/agents in each node. The agents can move from one node to a connected one. The reason why we focus our attention on discrete space is that many applications, like bike sharing systems or metapopulation epidemic models (MBR+12), are naturally framed in a discrete spatial structure. Moreover, in many circumstances continuous space is abstracted as a grid or as a mesh. This is the case, for instance, of many numerical methods that simulate the spatio-temporal dynamics using *Partial Differential Equations* (PDE). Hence, this class of models can be dealt checking properties on such discretisation.

1.3 Contributions

In this thesis, we design an original formal framework that offers suitable techniques to analyse the behaviour of both deterministic and stochastic complex systems, verifying their spatio-temporal properties.

First, we extend the robustness degree of STL formulae in a probabilistic setting (BBNS13; BBNS15). In particular, we provide a simulation-based method to formally define a notion of *robust satisfiability distribution*. This distribution is an important measure to understand how the behaviour specified by the temporal formula is affected by the stochasticity of the system. In particular, we consider two indicators of this distribution: the *average robustness* and the *conditional average robustness* of a formula being true or false. We present then three examples: the *Schlögl* system (GCPDI05), a simple set of biochemical reactions exhibiting a bistable behaviour, the *Incoherent type 1 Feed-forward loops* (I1-FFL) (Alo07), a frequent motif in gene regulatory systems, and the *Repressilator* (EL00;

BP08; BP10), a synthetic biological clock implemented as a gene regulatory network. With this examples we illustrate how to approximate the distribution of the robustness degree and its indicators, and we show that they provide valuable informations that are not captured by the satisfaction probability alone.

Secondly, we exploit the average robustness to address the *system design problem* (BBNS13; BBNS15). Given a stochastic model and a specific desired behaviour, described as a STL formula ϕ, the goal is to optimise (few) control parameters of such model in order to maximise the average robustness of ϕ. In this way, the model with the optimise parameters corresponds to the model that "better" (more robustly) shows the required specification. The methodology combines the formal method approach with a *machine learning* optimisation technique. Such optimisation is carried out using the *Gaussian Process Upper Confidence Bound* (GP-UCB) algorithm (SKKS12), coming from *active learning*. This state-of-the-art optimisation algorithm emulates the true function from just few samples and performs very well in a simulation based scenario with noisy evaluations. We consider then three different system design problems on the three case studies used to study the average robustness.

Third, we introduce *Signal Spatio-Temporal Logic*, SSTL, (NB14; NBC+15), a spatial extension of STL. We consider a discrete representation of space, modelled as a weighted graph, with populations of interacting components evolving in each location (node) and with entities migrating from one location to another one (via edges).

SSTL integrates the temporal modalities of STL with two new spatial operators: the *bounded somewhere* $\diamondsuit_{[d_1,d_2]}$ and the *bounded surround* $\mathcal{S}_{[d_1,d_2]}$. A third modality, the *everywhere* operator $\boxdot_{[d_1,d_2]}$, can be derived; $\diamondsuit_{[d_1,d_2]}\varphi$ holds when there exists a location at a distance between d_1 and d_2 from the current location where the property φ is satisfied, $\boxdot_{[d_1,d_2]}\varphi$ holds if all the locations with a distance between d_1 and d_2 from the current one satisfy φ. They permit to describe properties such as " from a bike sharing station, in a radius of 100 meters, there are more than 30 bikes", or "in all the positions around my location, at a distance less the 1 km there are no infected individuals". The surround operator

$\varphi_1 \mathcal{S}_{[d_1,d_2]} \varphi_2$, inspired by the spatial until modality defined in (CLLM14), expresses the topological notion of being surrounded by a φ_2-region, while being in a φ_1-region. This means that there exists a region A such that all its elements satisfy φ_1 and all the locations that surround A, i.e., that are directly connected with a location in A, satisfy φ_2. Furthermore, the locations have to satisfy the metric constraints. It permits for example to identify regions in space with high concentration of a certain protein surrounded by an high concentration of another one.

We provide a Boolean and a quantitative semantics for SSTL, extending the monitoring procedure of STL to compute the satisfaction over Boolean and real-valued spatio-temporal signals, and designing efficient algorithms for the new spatial operators. The major challenge is to monitor the surround operator for the quantitative semantics, for which we propose a novel fixed point algorithm, discussing its correctness and computational cost. This spatial monitoring requires a different algorithm from those developed for timed modalities, as space is bi-directional, thus it makes sense to observe both *reaching* and *being reached*; classical path-based model checking does not coincide with spatial model checking also because loops in space are not relevant in the definition of *surrounded* operators. We provide also a stochastic version of SSTL, leveraging the monitoring algorithms within a stochastic model checking routine. Then, we test the logic on two case studies: a model of the *spreading of cholera*, a waterborne disease, considering the effect of rivers in the diffusion of the epidemics (BAM$^+$08; MBR$^+$12) and a *pattern formation* in a *Turing reaction-diffusion* system modelling a process of *morphogenesis* (Tur52b).

Finally, we integrate SSTL within the statistical machine learning framework previously described to perform system design of spatio-temporal properties (BBM$^+$15). Furthermore, we extend, in a similar way, the smoothed model checking technique developed in (BMS16), that considers the problem of computing the satisfaction probability of a property for stochastic models with uncertainty parameters. Hence, we present a detailed spatio-temporal analysis of a developmental model of segmentation in Drosophila, known as the *French Flag pattern*. This analysis per-

mits novel insights as to how the various model parameters interact to give rise to the patterning behaviour.

To support qualitative and quantitative monitoring of SSTL properties, we have developed a Java tool, jSSTL. A Java library and an Eclipse plugin are available on-line at `http://quanticol.sourceforge.net/`. The source codes can be found at `https://bitbucket.org/LauraNenzi/jsstl`. The system design methodology, instead, for temporal models has been first implemented in a pseudo tool in `MATLAB` for the optimisation part and in Java for the modelling and simulation part, then it has been integrated in the tool U-check (BMS15), available at `https://github.com/dmilios/U-check`. U-check is a Java toolbox for formal analysis of stochastic systems, and it can be used also to estimate parameters from qualitative observations and to perform the smoothed model checking. The system design analysis for spatio-temporal models integrates jSSTL within U-check.

1.4 Structure of The Thesis

After this introduction (Chapter 1), the thesis is divided in two parts.

Part I presents the background material and contains a literature review. It is structured in three chapters.

In Chapter 2, we summarise the modelling approach of complex systems. First, we consider temporal modelling, without a characterisation of the space, in particular we illustrate stochastic, deterministic and hybrid dynamics for population models. Then, we examine systems with a discrete space described as a weighted graph, defining the patch-population models and their spatio-temporal dynamics.

In Chapter 3, we present the logic-based approach to specify the behaviour of complex systems. In particular, we describe *Signal Temporal Logic* (MN04; MNP08), STL, and its monitoring algorithms. Then, we give an overview of the existing spatial and spatio-temporal logics.

Chapter 4 summarises a number of statistical methods that we exploit in our work: *Statistical Model Checking,* (JCL+09b; YKNP04; YS06),

Gaussian Processes - Upper Confidence Bound Optimisation (RW06; SKKS12) and *Smoothed Model Checking* (BMS16).

Part II discusses our contribution and is organised in 4 chapters.

In Chapter 5, we define a novel notion of *robustness* for temporal properties of *stochastic* models and its application in the context of the *system design problem* (BBNS13; BBNS15), then we apply the new framework to a number of case studies.

In Chapter 6, we present *Signal Spatio-Temporal Logic*, SSTL, (NB14; NBC+15), a spatial extension of *Signal Temporal Logic* with two spatial modalities: the *bounded somewhere* operator $\diamondsuit_{[d_1,d_2]}$ and the *bounded surround* operator $\mathcal{S}_{[d_1,d_2]}$. We introduce the type of signals that the logic specifies, we define the syntax and the semantics of SSTL and we describe its monitoring procedure. In particular, we illustrate in detail the monitoring algorithms for the surround operator for the Boolean and the quantitative semantics and we prove their correctness. We, then, present a number of case studies to show the logic at work.

In Chapter 7, we extend the methodology presented in Chapter 5 to SSTL (BBM+15). Furthermore, we extend, in a similar way, the smoothed model checking technique developed in (BMS16), and described in Chapter 4. The entire framework can then be used to analyse and design systems with stochastic spatio-temporal dynamics. In particular, we apply the techniques to study a french-flag model of the Drosophila's Bicoid morphogen.

Chapter 8 describes jSSTL, a Java tool that we implemented, which makes possible to specify and verify SSTL properties over spatio-temporal traces. Furthermore, in this chapter, we briefly introduce U-check, a Java-tool for the analysis of stochastic complex systems with parametric uncertanty.

In the last chapter, Chapter 9, we report the concluding remarks and discuss future directions.

Part I

Background and literature

Chapter 2

Modelling Approaches

Modelling complex systems dynamics requires us to commit to some choices: time can be continuous or discrete, we can analyse the states of each individual agent or we can count the number of agents in each state, we can have a stochastic, a deterministic or an hybrid dynamics, and, we can examine or not the physical space of the system, again in a continuous or a discrete way. Different choices need different modelling approaches. Here, we report a selection of these possibilities and the connected modelling approaches that we will use later, in the next chapters, to construct our models. In this thesis, we considered only models with continuous time and aggregate states, i.e., we are interested in studying the behaviour of populations; in particular, we are interested in computing the number/density of agents in each state at each time. In the first section, we consider temporal modelling: we illustrate stochastic, deterministic and hybrid dynamics for population models and we introduce the Skorokhod metric. In the second section, instead, we discuss spatio-temporal modelling. In particular, we describe how to represent the space and the models related to this description.

2.1 Temporal Modelling

In this section, first we define a language to formally described population models, then we present three different dynamics: stochastic, deterministic, and hybrid, briefly introducing the simulation algorithms used to sample traces of stochastic processes. Finally, we define the Skorokhod metric, which endows the space of trajectories with a natural topology.

Population Model. A *population model* intuitively is a system in which a large number of different agents or components interact together and take, through local transitions, a number of different states. The transitions can be seen as descriptions of events changing the global state of the system. There are many example of population processes, like social systems, biochemical networks, ecological systems and computer networks.

Definition 2.1 (Population model) *A population model $\mathcal{M}$ is a tuple $\mathcal{M} = (\mathbb{S}, \boldsymbol{X}, \mathcal{T})$, where:*
– $\mathbb{S} = \{1, ..., n\}$ is the set of states of the agents in the population.
– $\boldsymbol{X} = (X_1, \cdots, X_n)$ is the state vector, describing the state of the population model. The variable $X_i \in \mathbb{R}_{\geq 0}$ represents the density, concentration, or number (in which case it takes integer values) of components in the i^{th} state. The domain of $\boldsymbol{X}$, i.e., the state space of the system, is a subset of $\mathbb{R}^n$ and is denoted by $\mathbb{D}$.
– $\mathcal{T} = \{\tau_1, ..., \tau_m\}$ is the set of global transitions of the form $\tau = (a, \mathbf{v}, f)$ where
• a is the label of the transition,
• $\mathbf{v} \in \mathbb{R}^n$ is the update vector, giving the net change of each counting variable due to the transition,
• $f : \mathbb{D} \to \mathbb{R}_{\geqslant 0}$ is the rate function, giving the rate of the transition as a function of the global state of the system.

Each transition τ can be seen as a rule of the form

$$q_1 s_{i_1} + \cdots + q_k s_{i_k} \to r_1 s_{j_1} + \cdots + r_h s_{i_h}$$

where $s_{i_a}, s_{j_b} \in \mathbb{S}$ are states of the system, and q_i, r_j are the stoichiometric coefficients, i.e., the amount of components/entities consumed or produced by the transition. The update vector $\mathbf{v}$ condenses the stoichiomet-

ric information as $\mathbf{v} = \sum_{b \leq h} r_b \mathbf{e}_{j_b} - \sum_{a \leq k} q_a \mathbf{e}_{i_a}$, where $\mathbf{e}_j$ is a vector equal to one in position j and zero elsewhere.

The dynamical evolution of these models can be described in different ways: we can interpret them stochastically as a *Markov chains* or deterministically as a system of *Ordinary Differential Equations* (ODEs) or hybrid as a *Stochastic Hybrid Automata* (SHA).

2.1.1 Stochastic Dynamics

Stochastic processes are useful mathematical constructs to describe the random evolution of a system in time. The following definition formalises the intuitive concept of random evolution.

Definition 1 *Let $(\Omega, \mathcal{A}, P)$ be a probability space*[1] *and $\mathbb{D} \subseteq \mathbb{R}^n$. A continuous time stochastic process with values in $\mathbb{D}$ is a collection of $\mathbb{D}$-valued random variables $\mathbf{X}(t)$, indexed by $t \in [0, \infty)$ and defined on the same probability space $(\Omega, \mathcal{A}, P)$.*

In this thesis, we will restrict our attention to the case where the sample space $\mathbb{D}$ is a subset of the m dimensional Euclidean space $\mathbb{R}^m$. We will be particularly concerned with the trajectory-based view of stochastic processes, restricting to those processes whose trajectories are *cadlag* functions. A *cadlag function* $g : [0, \infty) \to \mathbb{D}$ is a right continuous function having left limits for any $t \in [0, \infty)$, i.e., $g(t) = g(t^+)$ and $g(t^-)$ exists for all t. Call $\mathcal{D}([0, \infty), \mathbb{D})$ the space of cadlag functions with values in $\mathbb{D}$.

We adopt the following *notational conventions*: by $\mathbf{x}$ we denote an element of $\mathcal{D}([0, \infty), \mathbb{D})$, with $\mathbf{x}(t)$ representing the value of the cadlag function at time t. The stochastic process can be represented as a family of vectors:

$$(\mathbf{X}(t))_{t \in [0,\infty)} = (X_1(t), ..., X_n(t))_{t \in [0,\infty)}$$

where $\mathbf{X}(t)$ denotes the $\mathbb{D}$-valued random variable at time t, the state of the system at time t. In the case of population processes, each variable X_i counts the number of entities in the i^{th} state at time t. We denote by $\mathbf{X}$ the stochastic process seen as a random variable over $\mathcal{D}([0, \infty), \mathbb{D})$.

[1] Here Ω is the sample space, $\mathcal{A}$ is a sigma-algebra, and P a probability measure. See (Bil12) for an introduction of measure and probability theory.

The most important class of stochastic processes we will consider are *Continuous Time Markov Chains* (CTMCs) (Dur12) that describe population processes (PCTMCs). A *Continuous Time Markov Chain* is a stochastic process with a countable state space $\mathbb{D}$ that evolves in continuous time [2] ($T = \mathbb{R}_{\geqslant 0}$) and that has the *memoryless* property:

$$P(\mathbf{X}(t_{n+1}) = \mathbf{d}_{n+1} | \mathbf{X}(t_0) = \mathbf{d}_0, ..., \mathbf{X}(t_n) = \mathbf{d}_n) = P(\mathbf{X}(t_{n+1}) = \mathbf{d}_{n+1} | \mathbf{X}(t_n) = \mathbf{d}_n)$$

where $\mathbf{d}_0, ...\mathbf{d}_{n+1} \in \mathbb{D}$ and $t_0 < t_1 < ... < t_{n+1}$ is a increasing sequence of times. This means that the system has no memory of where it was in the past, hence the evolution of the system depends only on its present state.

Given a population model $\mathcal{M} = (\mathbb{S}, \mathbf{X}, \mathcal{T})$, the set of transitions $\mathcal{T} = \{\tau_1, ..., \tau_m\}$ specifies the dynamics of the system; they can be seen as the description of events changing the state of the system.

From the set of transitions $\mathcal{T}$, we can easily derive (BHLM13) the formal representation of a CTMC in terms of its infinitesimal generator matrix as:

$$\mathbf{Q}(\mathbf{d_i}, \mathbf{d_j}) = \sum_{\tau \in \mathcal{T} | \mathbf{v}_\tau = \mathbf{d}_j - \mathbf{d}_i} f_\tau(\mathbf{d}_i), \quad i \neq j \tag{2.1}$$

For each $\mathbf{d}_i, \mathbf{d}_j \in \mathbb{D}$, with $i \neq j$, $\lambda_{i,j} = \mathbf{Q}(\mathbf{d_i}, \mathbf{d_j})$ represents the rate of an exponential distribution $T_{i,j} \sim Exp(\lambda_{i,j})$, namely the distribution of a random variable modelling the time needed to go from state $\mathbf{d}_i$ to state $\mathbf{d}_j$. The diagonal elements of the matrix are equal to $\mathbf{Q}(\mathbf{d_i}, \mathbf{d_j}) = -\sum_{i \neq j} \mathbf{Q}(\mathbf{d_i}, \mathbf{d_j})$ and represent the opposite of the *exit* rate from the state $\mathbf{d}_i$. For all i, j s.t. $\nexists \mathbf{v}_\tau = \mathbf{d}_i - \mathbf{d}_j$, $\mathbf{Q}(\mathbf{d_i}, \mathbf{d_j}) = 0$. The state space of the PCTMC is $\mathbb{N}^n$ (or a proper subset, if any conservation law is in force). For more details about PCTMC, see for instance (BHLM13).

Such PCTMC can be simulated with standard algorithms, like *Stochastic Simulation Alghorithm* (SSA) (Gil77), or Gibson-Bruck method (GB00), or τ-leaping (CGP06). We briefly describe these algorithms bellow.

[2] Furthermore, we are assuming that the probability of the transition is independent of n (thus considering a time-homogeneous Markov Chain).

SSA. The SSA exploits the decomposition of a CTMC into the *jump chain* and the *holding time*. The *jump chain* specifies the probability of choosing a certain transition which is equal to $P(\tau|\mathbf{x}) = \frac{f_\tau(\mathbf{x})}{f(\mathbf{x})}$, where $f(\mathbf{x}) = \sum_{\tau\epsilon\mathcal{T}} f_\tau(\mathbf{x})$ is the sum of all rates; $f(\mathbf{x})$ is the rate of an exponential distribution, independent from the jump chain, modelling the time spent in state $\mathbf{x}$, and called *holding time*.

Let $\mathbf{x}$ be the current state at the current time t. At each time step, the algorithm first computes the rates $f_\tau(\mathbf{x})$ and $f(\mathbf{x})$, then it chooses the next transition $\hat{\tau}$, according to the jump chain distribution, and the time used for the transition $t_{\hat{\tau}}$, according to an exponential distribution with parameter $f(\mathbf{x})$, $\text{Exp}(f(\mathbf{x}))$. Finally, it updates the current state to $\mathbf{x} + \mathbf{v}_{\hat{\tau}}$ and the current time to $t + t_\tau$, and terminates typically when a final time is reached.

Gibson-Bruck This algorithm improves the previous method by storing not only the rates but also the firing time of each transition. An exponential distribution with rate λ, indeed, can be seen as the firing time of an event with probability to happen in a time between $[t, t + dt]$ equal to λdt. The idea is that, at each time step, a single transition happens during a time $t \in [0, s]$ and this information is used to update the firing time of the other transitions with a rule that we explain now. The rates $f_\tau(\mathbf{x}(t))$, seen as functions of a trajectory of the system, are time-dependent. We can see the race condition between transitions as a race condition between independent time-inhomogeneuous exponential random variable, coupled through the history $\mathbf{x}(t)$. Now, the cumulative rate of an exponential random variable with inhomogeneuous rate $f_\tau(\mathbf{X}(s))$ is $\Lambda_\tau(s) = \int_0^s f_\tau(\mathbf{X}(t))dt$. At time 0, let λ_0 be the rate of τ. We suppose that it does not change in time (until other transitions happens), then $\Lambda_\tau(s) = \int_0^s \lambda_0 dt = s\lambda_0$ and the firing time is $s_0 = \frac{1}{\lambda_0}\xi \sim \text{Exp}(\lambda_0)$ where ξ is an exponential random variable with rate 1. Now, suppose that, at time t_0, an event τ' happens and this changes the rate of τ in λ_1, then the firing time of τ can be found solving $\int_0^{t_0} \lambda_0 dt + \int_{t_0}^{s_1} \lambda_1 dt = s_1\lambda_1 - t_0\lambda_1 + t_0\lambda_0 = \xi = \lambda_0 s_0$, i.e.,

$$s_1 = \frac{\lambda_0}{\lambda_1}(s_0 - t_0) + t_0.$$

The rule can easily been generalised by induction to n intermediate jumps.

Let us describe now the algorithm. Let $\mathbf{x}_i$ be the state at time t_i. The algorithm is initialised by computing the rates $\lambda_0^\tau = f_\tau(\mathbf{x}(0))$ and the firing time s_0^τ according to an exponential distribution with parameter the rate λ_0^τ, $\text{Exp}(\lambda_0^\tau)$, for each $\tau \in \mathcal{T}$. Then, at each time step t_i the algorithm executes the fastest transition $\hat{\tau} = \arg\min_{\tau \in \mathcal{T}} s_i^\tau$. It updates the current state to $\mathbf{x}_{i+1} = \mathbf{x}_i + \mathbf{v}_{\hat{\tau}}$ and the current time to $t_{i+1} = t_i + s_i^{\hat{\tau}}$. Then, it updates the firing times s_{i+1}^τ of all the transitions except $\hat{\tau}$, following the described rule

$$s_{i+1}^\tau = \frac{\lambda_n^\tau}{\lambda_{n+1}^\tau}(s_i^{\hat{\tau}} - t_i) + t_i.$$

For $\hat{\tau}$, instead, it samples a new firing time $s_{i+1}^{\hat{\tau}} = t_i + \rho$, where ρ is generated according to an exponential distribution with parameter $\lambda_i^{\hat{\tau}}$, i.e., the rate of $\hat{\tau}$. Furthermore, to speed up the execution, the algorithm uses also a dependency graph for the states and a priority queue for the firing times.

τ-leaping. Another way to simple describe a PCTMC is in terms of Poisson processes. The process that counts how many times a transition η has fired up to time t is a time-inhomogeneuous Poisson process with cumulative rate $\int_0^\tau f_\eta(\mathbf{X}(s))ds$, where $f_\eta(\mathbf{X}(s))$ is the inhomogeneuous rate of the transition η. Let us call these processes $Y_\eta(t)$. It can be proved (Kur81) that the processes $Y_\eta(t)$ are independent and that the PCTMC can be described by the following equation in terms of Y_η:

$$\mathbf{X}(t) = \mathbf{X}(0) + \sum_{\eta \in \mathcal{T}} \mathbf{v}_\eta Y_\eta(t) = \mathbf{X}(0) + \sum_{\eta \in \mathcal{T}} \mathbf{v}_\eta y_\eta\Big(\int_0^\tau f_\eta(\mathbf{X}(s))ds\Big)$$

where $\mathbf{v}_\eta$ is the update vector of the transition η and y_η are the independent Poisson random variable associated with the Poisson process Y_η.

If τ is sufficiently small, we can assume the rate $f_\eta(\mathbf{X}(\tau))$ to be constant in $[0, \tau]$, which implies $\int_0^\tau f_\eta(\mathbf{X}(s))ds = \tau\lambda_\eta$, hence

$$\mathbf{X}(t) = \mathbf{X}(0) + \sum_{\eta \in \mathcal{T}} \mathbf{v}_\eta y_\eta(\tau\lambda_\eta)$$

This is at the heart of the τ-leaping talgorithm. Let $\mathbf{x}$ be the current state at the current time t. At each time step, the algorithm, first, chooses τ; then, for each transition, it samples a value m_η from the Poisson r.v. $y_\eta(\tau\lambda_\eta)$. Finally, it updates the current state to $\mathbf{x} + \sum_{\eta \in \mathcal{T}} \mathbf{v}_\eta m_\eta$ and the current time to $t + \tau$.

The challenge in this method is the choice of τ: if it is too small the computational cost to sample $y_\eta(\tau\lambda_\eta)$ become very high, if it too large the rate will not be constant in $[t, t + \tau]$.

2.1.2 Deterministic Dynamics

The deterministic dynamics is such that, given the initial conditions and the environment influencing the model (external signals), the response is uniquely determined: given the same input, the model always produces the same output. This class of models describes well systems that involve large numbers of agents so that the fluctuations of their number are (relatively) negligible. This dynamics is described by a set of (usually non-linear) Ordinary Differential Equations (ODEs).

From a Population model $\mathcal{M} = (\mathbb{S}, \mathbf{X}, \mathcal{T})$, we can construct a set of ODEs, assuming variables $\mathbf{X}$ to be continuous and interpreting each rate as a flow, thus obtaining the vector field

$$F(\mathbf{X}) = \sum_{\tau \in \mathcal{T}} \mathbf{v}_\tau \mathbf{f}_\tau(\mathbf{X}), \tag{2.2}$$

defining the ODEs $\frac{d\mathbf{X}}{dt} = F(\mathbf{X})$.

It is possible to prove that, with a suitable rescaling of the variables (i.e., dividing then by the system size), the dynamics of the CTMC for large populations converge to the solution of this ODE. A formal proof of this statement, known as mean-field or fluid approximation, can be found in (Kur70; BHLM13).

We give now a simple example of a population model and its stochastic and deterministic semantics.

Example 2.1 (SIR model) *We illustrate the definition of population models by a simple epidemic scenario involving N individuals. Each individual can be in three different states: susceptible to infection (S), infected (I), and recovered and immune to infection (R). The set of agent's states is then $\mathbb{S} = \{S, I, R\}$ and the variables are $\mathbf{X} = (X_S, X_I, X_R)$, where $X_i \in \{0, \cdots, N\}$. The state space $\mathbb{D}$ of the system is a subset of $[0, N]^3$. There are 3 different transitions:*

- *a susceptible individual can be infected by getting in contact with an infected individual,*

  $$inf : I + S \to 2I$$

  with $v_{inf} = (-1, 1, 0)$ and rate function $f_{inf}(\mathbf{X}) = k_{inf} \cdot X_S \cdot X_I$,

- *an infected individual can recover,*

  $$rec : I \to R$$

  with $v_{rec} = (0, -1, 1)$ and rate function $f_{rec}(\mathbf{X}) = k_{rec} \cdot X_I$,

- *a recovered individual can lose its immunity,*

  $$loss : R \to S$$

  with $v_{loss} = (1, 0, -1)$ and rate function $f_{loss}(\mathbf{X}) = k_{loss} \cdot X_R$.

The population model is then $\mathcal{M} = (\{S, I, R\}, \mathbf{X}, \mathcal{T})$ with

$$\mathcal{T} = \{(inf, v_{inf}, f_{inf}), (rec, v_{rec}, f_{rec}), (loss, v_{loss}, f_{loc})\}.$$

The stochastic process for this model can be represented by the family of vectors $(\mathbf{X}(t))_{t \in \mathbb{R}_{\geqslant 0}} = (X_S(t), X_I(t), X_R(t))_{t \in \mathbb{R}_{\geqslant 0}}$ where each $X_i(t) \in \mathbb{N}$ counts the number of entities in the state i at time t. We can derive the infinitesimal generator matrix Q from (2.1). For example, if we have 2 individuals, the state space $\mathcal{D}$ has six different elements, equal to the number of combinations to have the individuals in one of the three states. For instance, $d_i = (1, 1, 0)$ corresponds to the state of the system where an individual is susceptible and the other one is infected and $d_j = (0, 2, 0)$ corresponds to the state of the system where both the individuals are infected, then $d_j - d_i = (-1, 1, 0) = v_{inf}$ and

$$Q(i, j) = f_{inf}(d_i) = k_{inf} \cdot X_S \cdot X_I = k_{inf}.$$

The fluid approximation of this model, instead, is obtained applying (2.2) and assuming $\mathbf{X} \in \mathbb{R}^3$*, and it corresponds to the system of ODEs:*

$$\begin{cases} \frac{dX_S}{dt} = -X_I \cdot X_s \cdot k_{inf} + X_R k_{loss}, \\ \frac{dX_I}{dt} = X_I \cdot X_s \cdot k_{inf} - X_I k_{rec}, \\ \frac{dX_R}{dt} = X_I k_{rec} - X_R \cdot k_{loss}. \end{cases}$$

2.1.3 Hybrid Dynamics

Hybrid dynamics are dynamics where a part of its processes are treated as stochastically and a part deterministically. Deterministic dynamics are justified when all classes of agents are present at high concentrations, stochastic dynamics, instead, well describes systems with small populations. However, there are situation where some class of agents are large and others are very small. For example, genetic networks in a single cell: genes are normally present at very low copy numbers in cells, and their state can be usefully described as small finite state machines (BP13), i.e., as entities with a small number of internal states (e.g., free state, bound by a repressor molecule, etc). On the other hand, gene products (mRNAs and proteins) can have very high counts, so that modelling the genetic network as a PCTMC may incur significant computational costs. In these cases, a better strategy is to approximate only some variables as continuous, keeping discrete the others. This reflects in the hybrid dynamics: some transitions will be converted into flows (generally those modifying only continuous variables), while the others will remain stochastic discrete events. This gives rise to a model that can be expressed in terms of a class of Stochastic Hybrid Automata (SHA, (BP13)) known as Piecewise-Deterministic Markov Processes (Dav93).

More specifically, from a Population model $\mathcal{M} = (\mathbb{S}, \mathbf{X}, \mathcal{T})$, we design the SHA, keeping some variables of $\mathbf{X}$ discrete and assuming the others to be continuous, interpreting the rates of these second as flows. The SHA so obtained have discrete modes identified by the value of discrete variables. In between discrete transitions, the system evolves following the solution of a set of differential equations, whose vector field is mode-dependent (via the value of discrete variables). Discrete jumps happen at

exponentially distributed random times, at a non-constant rate that can depend on the continuous variables. After each jump, the value of the discrete variables can change. Also continuous variables can be updated, see (BP13) for further details.

SHA can also be defined by assuming a stochastic continuous dynamics within each mode (OMS13; ORS10). In this case, the system evolves in mode q by following a trajectory which is a solution of a *stochastic differential equation* (SDE) (Øks03) of the form

$$\frac{d\mathbf{X}}{dt} = F_q(\mathbf{X})dt + \sigma(\mathbf{X})d\mathbf{W},$$

where $\sigma(\mathbf{X})$ is a mode and state dependent Lipschitz continuous $n \times r$ *diffusion matrix*, $F_q(\mathbf{X})$ is the mode dependent *drift* (the deterministic part), and $d\mathbf{W}$ is an $r \times 1$ vector of uncorrelated Wiener processes (white noise). The dynamics then can be seen as a sequence of discrete jumps (the mode), interleaved by periods of continuous evolution along a trajectory of the SDE. Solutions of SDE can be approximately simulated using the standard *Euler-Maruyama* algorithm (RR08; Øks03). This method fixes a time step h and iteratively computes $\mathbf{x}(t+h) = \mathbf{x}(t) + F_q(\mathbf{x}(t))h + \sigma(\mathbf{x}(t))\mathcal{N}(0, hI_r)$, where $\mathcal{N}(0, hI_r)$ is a $r-$dimensional Gaussian random variable with mean 0 and covariance matrix hI_r.

2.1.4 Topology of The Space of Trajectories

The space $\mathcal{D}([0,\infty), \mathbb{D})$ can be given the structure of a metric space by the *Skorokhod metric*. The Skorokhod metric is first defined on compact time intervals $[0,T]$ and then extended over the whole positive time axis $[0,\infty)$. Consider the uniform metric on the space $\mathcal{D}([0,T], \mathbb{D})$, i.e., $d_U(\mathbf{x}', \mathbf{x}) = \sup_{0\le t\le T} \|\mathbf{x}'(t) - \mathbf{x}(t)\|$. The uniform metric endows the space of cadlag functions with a topology, however it is easy to see that this is too restrictive for most purposes. Consider, for example, the cadlag function $\mathbf{x}$ defined to be 0 for $t < t_0$, $t_0 > 0$ and 1 for $t \ge t_0$, and the sequence of cadlag functions $\mathbf{x}_n$ defined to be 0 for $t < t_0 + \frac{1}{n}$ and 1 otherwise. Then, for every $t \neq 0$, it is possible to find an N s.t. $\mathbf{x}(t) = \mathbf{x}_n(t)\ \forall n > N$, i.e., the sequence converges point-wise almost everywhere. However, under the

uniform topology $d_U(\mathbf{x}, \mathbf{x}_n) = 1\ \forall n$, so the sequence does not converge as a sequence of functions. This is a general fact: if we have a sequence $\mathbf{x}_n$ of cadlag functions, then they will converge to $\mathbf{x}$ in the uniform norm if and only if the discontinuous jumps of $\mathbf{x}_n$ happen precisely at the same times as those of $\mathbf{x}$ (for $n \geq n_0$).

The idea behind the Skorokhod metric is to relax the uniform metric by allowing a small difference in these jump times by resynchronising them. Informally, if the uniform metric allows one to wiggle space a bit, the Skorokhod metric allows us also to wiggle time. To formalise this statement, let $\omega(t) : [0, T] \to [0, T]$ be a time-wiggle function, i.e., a strictly increasing continuous function. Call $\mathcal{I}_T$ the set of such functions. Then, the Skorokhod distance between $\mathbf{x}, \mathbf{y} \in \mathcal{D}([0, T], \mathbb{D})$ is

$$d_T(\mathbf{x}, \mathbf{y}) = \inf_{\omega \in \mathcal{I}_T} \max\{ \sup_{t \in [0,T]} \|\omega(t) - t\|, \sup_{t \in [0,T]} \|\mathbf{x}(t) - \mathbf{y}(\omega(t))\| \}. \tag{2.3}$$

In our example above, one can simply choose the time wiggle function $\omega(t) = t + \frac{1}{n}$, so that the second term in the r.h.s. of equation (2.3) is always zero. The Skorokhod distance therefore evaluates to $\frac{1}{n}$, and the sequence is seen to converge. The metric d_T is extended to a metric on $\mathcal{D}([0, \infty), \mathbb{D})$ by discounting large times as follows:

$$d(\mathbf{x}, \mathbf{y}) = \sum_{K \in \mathbb{N}} 2^{-K} \min\{1, d_K(\mathbf{x}, \mathbf{y})\}.$$

The Skorokhod metric defines a topology for which $\mathcal{D}([0, \infty), \mathbb{D})$ is complete and separable, i.e., it is a Polish space[3]. See (Bil99) for a detailed introduction to the metric and its properties.

2.2 Spatio-Temporal Modelling

In this section, we consider spatially distributed systems. The main issue in the context of spatial modelling is the representation of space. It can be continuous or discrete, we can consider a metric in the space or not

[3]In fact, completeness requires one to work with an equivalent metric, but this is not relevant for this thesis (Bil99).

and we can have one, two or a three dimensional space. For an exhaustive classification of space and movement modelling techniques, we refer the reader to (ea31). Since most of our case studies naturally framed in a discrete spatial structure, we decided to consider a discrete space, taking into account discretisations of continuous spaces as a grid or as a mesh. This is the case, for instance, in many numerical methods to simulate the spatio-temporal dynamics of Partial Differential Equations (PDE). First, we formally describe how to represent the space then we define the models related to this description and their dynamics.

2.2.1 Discrete Space

If the space is discrete, it consists of a countable (often finite) number of locations that are linked in some way. We can think to it as a graph with the locations as nodes and the links as edges. The regular discrete space is a special case of the irregular one. For example, a 2D-grid can be seen as an undirected graph where each node has a fixed vertex degree equal to four.

In particular, we will consider discrete models of space that can be represented as a finite weighted undirected graph.

Definition 2.2 *A* **(positive) weighted undirected graph** *is a tuple* $G = (L, E, w)$*, where:*

- L *is the finite set of locations (nodes),* $L \neq \varnothing$
- $E \subseteq L \times L$ *is a symmetric relation, namely the set of connections (edges),*
- $w : E \to \mathbb{R}_{>0}$ *is the function that returns the cost/weight of each edge.*

The space is also equipped with a metrics.

Definition 2.3 *The* **weighted distance** *is defined as*

$$d(\ell, \ell') := \min\{\sum_{e \in S} w(e) | S \text{ is a path between } \ell \text{ and } \ell'\}.$$

This means that the weighted distance is a metrics that returns the cost of the shortest path, for each pair of nodes of the graph; where the shortest path is the path that minimises the sum of costs.

Remark 2.1 *If we give an order to the locations, $L = \{\ell_1, ..., \ell_i, ...\}$, then the weighted distance can be seen as a matrix $(d)_{i,j \in E^*}$, where $d_{i,j}$ is the distance from ℓ_i to ℓ_j.*

Furthermore, we denote by E^* the set containing all the pairs of connected locations, i.e., the transitive closure of E, and by $L^{\ell}_{[d_1,d_2]}$ the set of locations ℓ' at a distance between d_1 and d_2 from ℓ, formally

$$L^{\ell}_{[d_1,d_2]} := \{\forall \ell' \in L \mid d_1 \leq d(\ell, \ell') \leq d_2, \text{ with } d_1, d_2 \geq 0\}.$$

This means that edges of the graph are equipped with a positive weight, giving a metric structure to the space, in terms of shortest path distances. The weight will often represent the distance between two nodes. This is the case, for instance, when the graph is a discretisation of continuous space. However, the notion of weight is more general, and may be used to encode different kinds of information. As an example, in a model where nodes are locations in the city and edges represent streets, the weight could represent the average travelling time, which can be different between two paths with the same physical length but different levels of congestion or different number of traffic lights.

Finally, another relevant notion for this work is that of *external boundary* of a set of nodes A, i.e., the set of nodes directly connected with an element of A but not part of it.

Definition 2.4 *Given a subset of locations $A \subseteq L$, we define the boundary of A as:*

$$B^+(A) := \{\ell \in L \mid \ell \notin A \wedge \exists \ell' \in A \text{ s.t. } (\ell', \ell) \in E\}.$$

All these terms naturally extend from graphs to directed graphs.

2.2.2 Pacth-Based Population model

In this subsection, we adapt the definition and the dynamics of population models, described in the previous section, to systems embedded in a discrete space. The definition of a spatial population model is strictly related to the choice of the space in which the model is embedded. As said in the previous subsection, we will mainly work with discrete spaces, in

particular with weighted graphs. Each node represents a different spatial location (*patch*), each patch contains a population of agents, described by indicating the number/density of individuals in each state as a classic population process, as explained in Section 2.1.

Definition 2.5 (Patch-based population model.) *A Patch-based population model is a tuple* $(\mathcal{M}, G, \mathcal{V})$ *where:*

- $\mathcal{M} = (\mathbb{S}, \mathbf{X}, \mathcal{T})$ *is a population model, satisfying Definition 2.1,*
- $G = (L, E, w)$ *is a weighted graph,*
- $\mathcal{V} = \{\nu_1, ..., \nu_k\}$ *is the set of inter-patch transitions, i.e., the transitions that describe the migration of entities between patches, each transition is of the form* $\nu_l = (a_l, s, g_l)$*, where:*
  - a_l *is the label of the transition,*
  - $s \in \mathbb{S}$ *is the state of the entity that migrates*
  - $g_l : \mathbb{D} \times L \times L \to \mathbb{R}_{\geqslant 0}$ *is the rate function, where* $\mathbb{D}$ *is the state space of the system;* $g_l(\mathbf{X}, \ell_i, \ell_j)$ *is the rate for the migration of a component in state* s *from location* ℓ_i *to location* ℓ_j *when the global state of the system is* $\mathbf{X}$*.*

We can then refer to the description of such a process by a family of vectors

$$\mathbf{X}(t, \ell)_{t \in T, \ell \in L} = (X_1(t, \ell), X_2(t, \ell), ..., X_n(t, \ell))_{t \in T, \ell \in L},$$

indexed by time and space.

In Figure 2.1, taken from (ea31), we can see a graphical representation of discrete space models. The graphs consider two classes of agents A and B, the first with two states A_1 and A_2 and the second with three, B_1, B_2, and B_3. Figure 2.1(a) describes a model with discrete state space; indeed, we can see that individual tokens are grouped into stacks over the locations/nodes. Figure 2.1(b), instead, considers continuous state space, i.e., it describes the states by concentrations. This is represented in the graph by a column with a real-valued height for each state in each location.

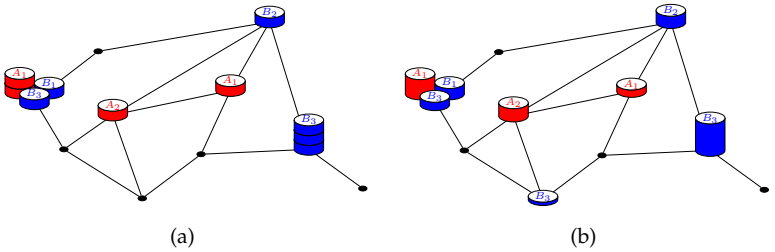


Figure 2.1: Representation of discrete space models with discrete (a) and continuous (b) state variables. The figure is taken from (ea31).

We treat the dynamics of these systems by considering one (time-dependent) variable $X_{i,\ell}$ for each state-location pair, i.e., the variables represent the number/density of each agent state in each location. The dynamics then, given by intra-patch interactions and inter-patch migration of agents, results in a PCTMC for the discrete state space and in a ODE system for the continuos one. However, the number of variables is the product between the number of locations and the number of states, hence, the computational cost becomes quickly very high, especially for the simulation of stochastic systems. In case of the deterministic dynamics, the ODEs system is often the result of the discretisation of a *Partial Differential Equation* (PDEs) system, used to treat continuous spaces, according to a *Finite Difference* scheme. For example, a square, seen as continuous 2-dimensional space can be discretised in a $K \times K$rectangular grid, where each cell represents a location (node), we will have then a ODE for each variable and each location of the system.

To see a nice and simple example of patch-based population models we refer to the first case study present in Section 6.5, a model of cholera outbreak.

We remark that, the fact that the space is finite permits, as in the temporal case, to see the stochastic spatio-temporal trajectories as a random variable over the space of cadlag functions $\mathcal{D}([0,\infty],\mathbb{D})$, where in this case $\mathbb{D} = \mathbb{D}_1 \oplus \cdots \oplus \mathbb{D}_m$, with $m = |L|$, the number of locations.

Chapter 3

Logics and Monitoring

In the first part of this chapter, we present a brief introduction on temporal logics and verification techniques for complex systems. In the second part, we introduce the *Signal Temporal Logic* (MN04; MNP08), STL, a suitable logic to specify and verify temporal dynamics of complex systems. In particular, we describe the STL syntax and semantics and monitoring algorithms to verify STL properties. Finally, in the last section, we give an overview of the existing spatial and spatio-temporal logics.

3.1 How to Specify and Verify Behaviours of Complex Systems

As described in Chapter 2, the dynamics of our systems is given, usually, by PCTMC or patch PCTM (stochastic dynamics) or by a set of ODEs (deterministic dynamics). We will analyse these models by means of simulation. For ODEs models, we have a single trajectory of the system, usually given by the approximate solution of the equation's (by numerical integration). With the PCTMC, methods as Gillespie's Stochastic Simulation Algorithm (Gil77), SSA, are used to compute a number of samples/trajectories of the systems. The behaviour that we want to specify then has to be related to these trajectories/simulations of the models.

Formally, given a population model $\mathcal{M} = (\mathbb{S}, \mathbf{X}, \mathcal{T})$, with state vector

$\mathbf{X} = (X_1, ...X_n) \in \mathbb{D}$ (defined in Chapter 2), a *trace/trajectory/path* of the system is a function $\xi : \mathbb{T} \to \mathbb{D}$ from the time domain $\mathbb{T} \subseteq \mathbb{R}_{\geq 0}$, to the state space $\mathbb{D} = D_1 \times \cdots \times D_n$. If we call $\mathcal{D}$ the set of all possible path (the trajectory space) over $\mathbb{D}$, we can define a linear-time *property* φ as a subset L_φ of $\mathcal{D}$. The subsets L_φ can be defined syntactically using different formalisms such as logical formulae, regular expressions or automata that accept them.

One of these formalisms is temporal logic (Pnu77); It provides a very elegant framework to specify in a compact and formal way temporal behaviours. It is a modal logic that has specific operators (temporal operators) to describe properties of *time-dependent* events. There are many kinds of temporal logics, we focus on *linear dense-time* temporal logics because we analyse the trajectories space (i.e., sequence of observations) and we model systems with a continuous dynamics.

Given a model and a property, we then need then automatic techniques to check whether the property is satisfied. For the type of systems that we want to analyse (very large and complex), we can use a *monitoring* approach. A *monitoring* procedure is a technique to verify if a *single* trajectory ξ, satisfies or not a given property ϕ [1], i.e., $\xi \vDash \phi$.

Usually, a monitoring algorithm produces a *Boolean* answer, yes/no, depending if the observed trace satisfies or not the property. This means that different trajectories can all satisfy the same property. For example, let us consider the trace $\xi : \mathbb{T} \to \mathbb{R}$, and the property $(\xi(0) > k)$, i.e., we want to verify if the trace at time zero is larger than a constant k, $\xi \vDash (\xi(0) > k)$; a Boolean monitoring will produce the same answer if $\xi(0) = k + \epsilon$ or if $\xi(0) >> k$. To address this problem, recently some researchers have proposed several notions of *robustness* (DM10; FP09; RBFS08), providing suitable definitions of distance between a trajectory of a system and the behavioural property of interest. These new notions can be used to endow the logic with a *quantitative semantics*, allowing to give a measure of the satisfaction of the desired specification. This value can be a key indicator and add important informations about the dynamics of our

[1] Note that this means also that it does not need a mathematical model but just observable traces of some process, even real observations.

complex systems.

To deal with stochastic dynamics, where we have a trajectory space and not just a single trace, as in the deterministic case, we can use techniques such as *statistical model checking* (SMC) (YKNP04). This method will be explained in detail in the next chapter where we summarise some statistical methods exploited in this thesis.

3.2 Signal Temporal Logic

Among the myriads of temporal logics available, we decided to work with *Signal Temporal Logic* (STL) (MN04; DM10). STL is a temporal logic suitable to characterise behavioural patterns in real-valued time series, generated during the simulation of dynamical systems. It extends the dense-time semantics of the *Metric Interval Temporal Logic* (MITL) (AFH96) with a set of parametrised numerical predicates playing the role of atomic propositions and it is interpreted on real-valued signals. It comes with a Boolean and a quantitative semantics and efficient monitoring algorithms (MN04; DFM13) to check its properties. Furthermore, two interesting tools have been developed for this logic: AMT (NM07) for the Boolean semantics and Breach (Don10) for the quantitative one.

Signals. A *signal* s is a function $\mathbf{s} : \mathbb{T} \to D$, where $\mathbb{T}$, the time domain, is a real-valued interval $[0, T] \subseteq \mathbb{R}_{\geq 0}$, for some $T > 0$, and D is a subset of $\mathbb{R}^* = \mathbb{R} \bigcup \{+\infty, -\infty\}$. Signals with $D = \mathbb{B} = \{0, 1\}$ are called *Boolean signals*, while signals whit $D = \mathbb{R}^*$ are called real-valued or *quantitative* signals.

Given a trace $\xi : \mathbb{T} \to \mathbb{D}$, we can define $\mathbf{x}(t) \coloneqq (x_1(t), \cdots, x_n(t))$, where each $x_i = \xi_{|D_i} : \mathbb{T} \to D_i$, for $i = 1, ..., n$, is the projection on the i^{th} coordinate/variable of ξ. Note that these projections have the form of quantitative signals. They are called the *primary signals* of the trace. We can then see the trace as a set of primary signals. This means that STL can specify properties of temporal traces. For simplicity, from now we denote the trace as $\mathbf{x}$.

Definition 3.1 (STL syntax) *The syntax of STL is given by*

$$\varphi := \top \mid \mu \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \mathcal{U}_{[a,b]} \varphi_2,$$

where $\top$ is a true formula, conjunction and negation are the standard Boolean connectives, $[a, b]$ is a dense-time interval with $a < b$ and $\mathcal{U}_{[a,b]}$ is the until operator. The atomic predicate $\mu : \mathbb{R}^n \to \mathbb{B}$ is defined as $\mu(\mathbf{d}) := (y(\mathbf{d}) \geqslant 0)$, $\mathbf{d} \in \mathbb{R}^n$, where $y : \mathbb{R}^n \to \mathbb{R}$ is a real-valued function. The predicate μ can be lifted to an operation between signals, transforming a real valued signal into a Boolean one, i.e., to a mapping $\mu : \mathcal{D}([0, \infty), \mathbb{R}^n) \to \mathcal{D}([0, \infty), \mathbb{B})$, by $\mu(\mathbf{x})(t) = (y(\mathbf{x}(t)) \geqslant 0)$; $y(\mathbf{x}(t))$ is known as the secondary signal.

The (bounded) until operator $\varphi_1 \mathcal{U}_{[a,b]} \varphi_2$ requires φ_1 to hold from now until, in a time between a and b time units, φ_2 becomes true. The *eventually* operator $\mathcal{F}_{[a,b]}$ and the *always* operator $\mathcal{G}_{[a,b]}$ can be defined as usual: $\mathcal{F}_{[a,b]}\varphi := \top\mathcal{U}_{[a,b)}\varphi$, $\mathcal{G}_{[a,b]}\varphi := \neg\mathcal{F}_{[a,b]}\neg\varphi$. We introduce now the Boolean and the quantitative semantics for STL as in (MN04; DM10).

Definition 3.2 (STL Boolean Semantics) *The Boolean satisfaction relation $\vDash$ for an STL formula φ on a temporal trace $\mathbf{x}$ is defined recursively by:*

$$\begin{aligned}
&(\mathbf{x}, t) \vDash \top \\
&(\mathbf{x}, t) \vDash \mu && \Leftrightarrow \quad \mu(\mathbf{x}(t)) = 1 \\
&(\mathbf{x}, t) \vDash \neg\varphi && \Leftrightarrow \quad (\mathbf{x}, t) \nvDash \varphi \\
&(\mathbf{x}, t) \vDash \varphi_1 \wedge \varphi_2 && \Leftrightarrow \quad (\mathbf{x}, t) \vDash \varphi_1 \text{ and } (\mathbf{x}, t) \vDash \varphi_2 \\
&(\mathbf{x}, t) \vDash \varphi_1 \mathcal{U}_{[a,b]}\varphi_2 && \Leftrightarrow \quad \exists t' \in [t + a, t + b] : (\mathbf{x}, t') \vDash \varphi_2 \wedge \forall t'' \in [t, t'), (\mathbf{x}, t'') \vDash \varphi_1
\end{aligned}$$

A trace $\mathbf{x}$ satisfies φ, denoted by $\mathbf{x} \vDash \varphi$, if and only if $(\mathbf{x}, 0) \vDash \varphi$.

This because we are working with *future* temporal modalities whose truth value now depends on the future states of the trace.

Definition 3.3 (STL Quantitative Semantics) *The* quantitative satisfaction function *ρ returns a value $\rho(\varphi, \mathbf{x}, t) \in \tilde{\mathbb{R}}^2$ quantifying the robustness degree (or satisfaction degree) of the property φ by the signal $\mathbf{x}$ at time t with respect to*

[2] $\tilde{\mathbb{R}} = \mathbb{R} \cup \{-\infty, +\infty\}$.

perturbations. It is defined recursively as follows:

$$
\begin{array}{lcl}
\rho(\top, \mathbf{x}, t) & = & \top \\
\rho(\mu, \mathbf{x}, t) & = & y(\mathbf{x}(t)) \qquad \textit{where } \mu \equiv y(\mathbf{x}(t)) \geqslant 0 \\
\rho(\neg\varphi, \mathbf{x}, t) & = & -\rho(\varphi, \mathbf{x}, t) \\
\rho(\varphi_1 \wedge \varphi_2, \mathbf{x}, t) & = & \min(\rho(\varphi_1, \mathbf{x}, t), \rho(\varphi_2, \mathbf{x}, t)) \\
\rho(\varphi_1 \, \mathcal{U}_{[a,b]} \varphi_2, \mathbf{x}, t) & = & \sup\limits_{t' \in t + [a,b]} (\min(\rho(\varphi_2, \mathbf{x}, t'), \inf\limits_{t'' \in [t,t')} (\rho(\varphi_1, \mathbf{x}, t'))))
\end{array}
$$

Moreover, we let $\rho(\varphi, \mathbf{x}) \coloneqq \rho(\varphi, \mathbf{x}, 0)$.

The sign of $\rho(\varphi, \mathbf{x})$ provides the link with the standard Boolean semantics of (MN04): $\rho(\varphi, \mathbf{x}) > 0$ only if $\mathbf{x} \vDash \varphi$, while $\rho(\varphi, \mathbf{x}) < 0$ only if $\mathbf{x} \nvDash \varphi$. The case $\rho(\varphi, \mathbf{x}) = 0$, instead, is a borderline case, and the truth of φ cannot be assessed from the robustness degree alone. Furthermore, ρ does not preserve logical equivalence, for instance $\phi \vee \neg\phi = \top$, but $\rho(\phi \vee \neg\phi, \mathbf{x}, t) = \max(\rho(\phi, \mathbf{x}, t), -\rho(\phi, \mathbf{x}, t)) \neq \top = \rho(\top, \mathbf{x}, t)$.

The absolute value of $\rho(\varphi, \mathbf{x})$, instead, can be interpreted as a measure of the robustness of the satisfaction with respect to noise in signal $\mathbf{x}$, measured in terms of the induced perturbation in the secondary signal. This means that if $\rho(\varphi, \mathbf{x}, t) = r$ then for every signal $\mathbf{x}'$ for which every secondary signal satisfies $\max_t |y_j(t) - y'_j(t)| < r$, we have that $\mathbf{x}(t) \vDash \varphi$ if and only if $\mathbf{x}'(t) \vDash \varphi$.

Remark 3.1 *We stress that the choice of the secondary signals $y : \mathbb{R}^n \to \mathbb{R}$ is an integral part of the definition of the STL formula, reflecting the intuition of the modeller and encoding the behaviour of interest. Different choices of secondary signals result in formulae expressing different behavioural properties, hence naturally in different robustness measures.*

The robustness degree of Definition 3.3 has to be interpreted as a weight of "how much" a given model (with fixed initial conditions and parameters) satisfies a STL formula. More precisely, its absolute value represents the max distance of the signal $\mathbf{x}$ under consideration from the set of trajectories satisfying/violating the formula (FP09).

In this sense, this measure is different from the more common sensitivity-based notions of robustness, like those discussed in (KCRS11), measur-

ing the size of a region in the parameter space in which the system behaviour is roughly constant. However, sensitivity analysis and its related techniques can be applied as well in combination with the robustness degree of Definition 3.3 as a discriminative function for interesting behavioural patterns to investigate. The definition of *robustness* or *robust satisfaction* considered here was first proposed by Fainekos and Pappas in (FP09) and later extended by Donzé et al. in (DM10). This is different from the *satisfaction/violation degree* or *quantitative satisfaction* defined by Rizk et al. in (RBFS08). In the first case the *robustness* corresponds to the distance of a signal from a set of signals satisfying the same formula (the minimal perturbation value that can violate the specification), while in the second case it provides a distance between a formula and a set of formulae satisfying the same signal (RBFS08).

3.2.1 Monitoring

A monitoring procedure is an algorithm for deciding whether a given trace $\mathbf{x}$ satisfies a property φ, $\mathbf{x} \vDash \varphi$, i.e., whether $\mathbf{x} \in L_\varphi$. The algorithm, for an STL formula φ, works with a bottom-up approach on the syntax tree of φ, iteratively computing the temporal signals of each subformula. Each node of the tree represents a subformula, the leaf are the atomic propositions and the root represents the whole formula. Given a trace $\mathbf{x}(t)$, the algorithm starts computing the Boolean and/or the quantitative signals of all the atomic propositions, then it goes up on the tree computing the Boolean and/or the quantitative signals of a node using the signals of its child. There exist then a specific algorithm to compute the signal for each operator $*$ of the logic.

Let's illustrate this in a sketch of the algorithm for the quantitative signal, Algorithm 1. The algorithm, given as input a trace $\mathbf{x}$ and a formula φ, returns the quantitative signal $\text{RobustSignal}(\varphi, \mathbf{x}(t)) := \rho(\varphi, \mathbf{x}, t)$ of the trace $\mathbf{x}$ for the formula φ. We recall that if $\text{RobustSignal}(\varphi, \mathbf{x}(0)) > 0$ then $\mathbf{x} \vDash \varphi$. As trace, the monitoring in (DFM13) considers a piecewise-linear function obtained by linear interpolation of a sequence of time-stamped values (usually the solution of the numerical integration), also

the output RobustSignal$(\varphi, \mathbf{x})$ is a piecewise-linear function.

Algorithm 1 RobustSignal$(\varphi, \mathbf{x})$

case $\varphi = \top$
  return $y = \bar{\top}$ (a constant signal true)
case $\varphi = y(\mathbf{x}(t)) \geq 0$
  return y
case $\varphi = *\varphi_1$
  y =RobustSignal$(\varphi_1, \mathbf{x})$
  return Compute $(*, y)$
case $\varphi = \varphi_1 * \varphi_2$
  y_1 =RobustSignal$(\varphi_1, \mathbf{x})$
  y_2 =RobustSignal$(\varphi_2, \mathbf{x})$
  return Compute $(*, y_1, y_2)$

The sketch of the algorithm for the Boolean signal is almost the same, replacing the atomic proposition case with a Boolean signal instead of the secondary signal y.

In the running example below, we present informally how some STL operators can specify dynamic behaviours and how they can be monitored. To see in detail the monitoring algorithms we refer to (MN04) for the Boolean semantics and to (DFM13) for the quantitative one.

Example 3.1 (SIR properties) *Let us consider the SIR population model presented in the example 2.1, Chapter 2, with the deterministic semantics. It corresponds to the system of ODEs:*

$$\begin{cases} \frac{dX_S}{dt} = -X_I \cdot X_s \cdot k_{inf} + X_R k_{loss}, \\ \frac{dX_I}{dt} = X_I \cdot X_s \cdot k_{inf} - X_I k_{rec}, \\ \frac{dX_R}{dt} = X_I k_{rec} - X_R \cdot k_{loss}. \end{cases}$$

where, the variables X_S, X_I, and X_R represent the concentration of susceptible, infected and recovered individuals. Given the initial condition of the system, $\mathbf{X}(0) = (0.9, 0.1, 0)$, we can derive the solution of the system via numerical integration. We normalise the values s.t. $X_S + X_I + X_R = 1$. Let us denote the trace of the system as $\mathbf{x}(t) = (x_S(t), x_I(t), x_R(t))$. A plot of the trace can be found in Figure 3.1.

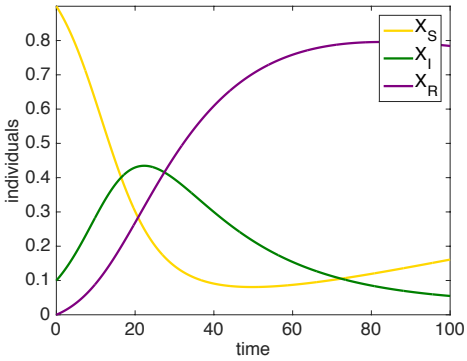


Figure 3.1: A simulation of a deterministic epidemic model.

For example, we want to study if, in the presence of an infection, after a certain time, the concentration of recovered individuals remains high for at least 30 time units. This behaviour can be specified by the formula

$$\varphi_{SIR} : (x_I > 0.05) \wedge \mathcal{G}_{[60,90]} (x_R > 0.7), \tag{3.1}$$

that means "at time 0, the concentration of infected individuals is more than 0.05 and, after 60 time units, the concentration of recovered individuals remains more than 0.7 for at least 90-60=30 time units" .

To see if the trace $\mathbf{x}(t) = (x_S(t), x_I(t), x_R(t))$ satisfies the formula φ_{SIR}, the monitoring algorithm has, first, to determine the parse tree of the formula and then to compute the Boolean or the quantitative signals, from the atomic propositions to the root, climbing up the tree. The Boolean and quantitative satisfaction correspond then to the value of the Boolean and quantitative signals of φ_{SIR} at time zero. The procedure is illustrated in Figure 3.2.

In Figure 3.3 we plot the Boolean and the quantitative signals for each subformula. The atomic predicates of the formula are: $\mu_1 : x_I - 0.5 > 0$ and $\mu_2 : x_R - 0.7 > 0$. From these we can derive the secondary signals $y_1(\mathbf{x}(t)) = x_I(t) - 0.5$ and $y_2(\mathbf{x}(t)) = x_R(t) - 0.7$, where $x_S(t)$, $x_I(t)$, and $x_R(t)$ are the primary signals, and $\mathbf{x} = (x_S, x_I, x_R)$ corresponds to a trajectory of the system.

We discuss just the computation of the quantitative signal, considering the Boolean signal equal to one only when the quantitative signal is positive, i.e., $s_\phi(t) := 1$ if $\rho_\phi(t) > 0$. However, there exist also a specific monitoring procedure for the Boolean semantics (MN04) that uses the interval covering of a signal; we will discuss and exploit it in our algorithms in Chapter 6.

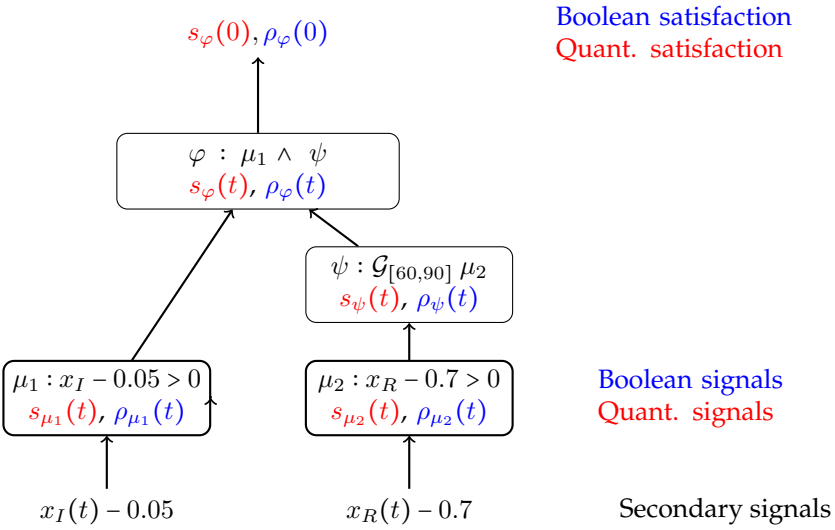


Figure 3.2: The monitoring procedure of the formula φ.

First, the quantitative signals of the atomic propositions correspond exactly to the secondary signals, i.e., $\rho_{\mu_i}(t) = y_i(t)$. *Then, from the signal* $\rho_{\mu_2}(t)$, *we can compute the signal* ρ_ψ *of the subformula* $\psi : \mathcal{G}_{[60,90]}\mu_2$. *The temporal modalities of STL are future temporal modalities, i.e., given a time* t, *they verify whether a property is satisfied at a certain time in a future dense-time interval* $t \oplus [a, b] = [t + a, t + b]$. *For this reason the monitoring procedure has to go backwards, i.e., to shift the signal backwards. Let* $[0, T]$ *be the time domain of the signal* μ_2. *Then, time domain of* ψ *will be* $[0, T - 90]$.

For a fixed trace, for compactness of notation, we denote the quantitative signal of a formula ψ *as* $\rho_\psi := \rho(\psi, x)$. *From the quantitative semantics, we have that*

$$\rho_\psi(t) = \min_{t' \in [t+60, t+90]} (\rho_{\mu_2}(t)), \quad \text{for } t \in [0, 10].$$

Finally, $\rho_{\phi_{SIR}}(t) = \rho_{\mu_1 \wedge \psi}(t) = \min(\rho_{\mu_1}(t), \rho_\psi(t))$, *for* $t \in [0, 10]$.

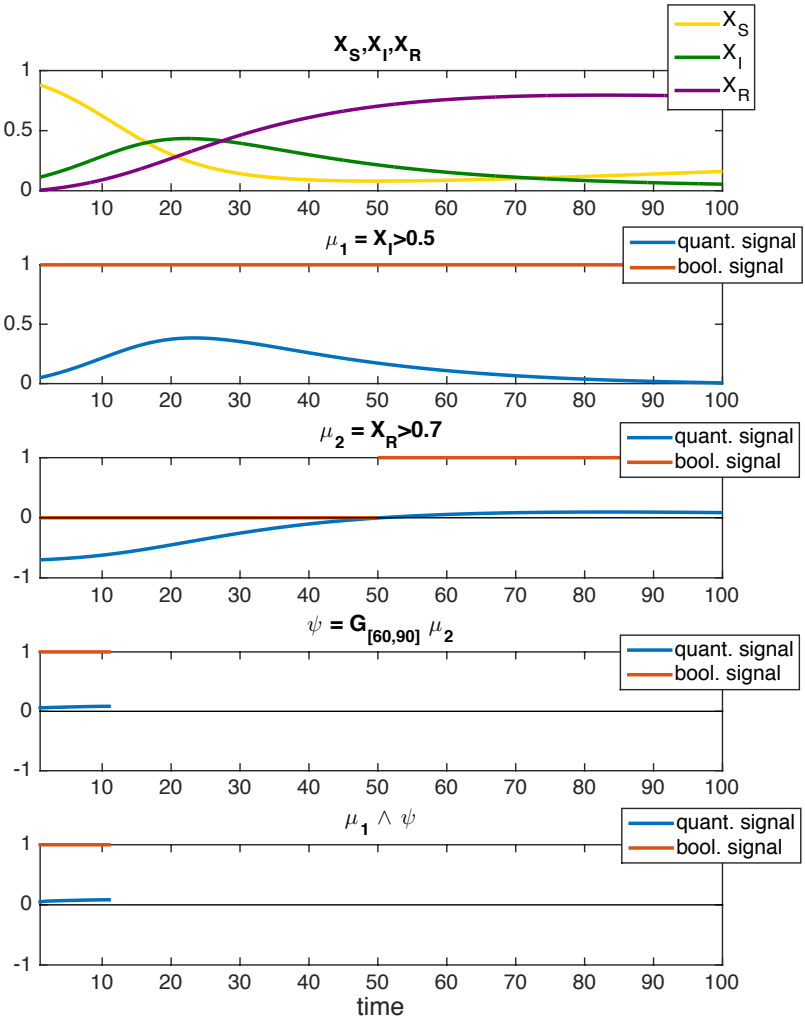


Figure 3.3: The Boolean and the quantitative signals for each subformulae of the formula 3.1

3.3 A Literature Review on Spatial Logics

In this section, we give an overview of the existing logics dealing with, in some way, spatial aspects of systems. In general, logics and in particular their semantics are usually related to the class of model on which they are interpreted. As an extreme example, spatial properties cannot be interpreted in a model with no spatial aspects. Another example is the correspondence between logic operators and the constructs in Process Calculus like in the *Ambient Calculus* (CG00). Hence, we classify logics depending on the type of models they have to be interpreted on. The first two subsections describe logics based on topological or closure space models. The third subsection illustrates SpaTel (HJK+15), a spatio-temporal logic in which spatial properties are expressed using ideas from image processing, namely *quad trees* (FB74). The fourth subsection presents two examples of logics for *process algebras with locations* (Cas01). In the last section, we report some others spatial logics.

3.3.1 Topo-Logics

Here we present some investigations about spatial operators in the context of topological space and the S4 modal logic. The material comes from the Handbook of Spatial Logics (APHvB07), in particular Chapters 5 and 9, and from the technical report (CML08).

Given a set X, a *topology* on X is a collection $\mathcal{T} \subset \mathcal{P}(X)$ of subsets of X (called *open sets*) s.t.: $\varnothing, X \in \mathcal{T}$ and $\mathcal{T}$ is closed under arbitrary unions and under finite intersections. The pair (X,$\mathcal{T}$) is called a topological space.

A *topological model* $\mathcal{M}$ is a topological space provided with a *valuation* function $\mathcal{V}$ from the set of atomic propositions to $\mathcal{P}(X)$ [3]. The interpretation of a formula ϕ is given over the single point $x \in X$, i.e., $\mathcal{M}, x \vDash \phi$.

The first spatial interpretation with this type of model consists in the description of open/closed properties using the S_4 modal operators $\Box$ and $\Diamond$. Given a formula ϕ that holds in the subset S of X, $\Box\phi$ means that ϕ holds in the *interior* of S (the largest open set contained in S) and $\Diamond\phi$

[3] Given an atomic proposition p, $\mathcal{V}(p)$ are the set of $x \in X$ for which p holds.

means that ϕ holds in the *closure* of S (the smallest closed set containing S). In other words,

$$\mathcal{M}, x \vDash \Box\phi \iff \exists O \in \mathcal{T} \text{ s.t. } x \in O \text{ and } \mathcal{M}, y \vDash \phi \quad \forall y \in O$$

$$\mathcal{M}, x \vDash \Diamond\phi \iff \forall O \in \mathcal{T} \text{ s.t. } x \in O \text{ implies that } \exists y \in O \text{ s.t. } \mathcal{M}, y \vDash \phi$$

From these we can derive other connectives such as the *boundary* operators $\mathcal{B}\phi = \Diamond\phi \wedge \neg \Box \phi$.

Extending the logic with quantifiers (the *existential* operator E and the *universal* operator U) permits the specification of global properties, i.e., to speak about the behaviour of a subset of points of the space. In detail,

$$\mathcal{M}, x \vDash E\phi \iff \exists y \in X \text{ s.t. } \mathcal{M}, y \vDash \phi$$

$$\mathcal{M}, x \vDash U\phi \iff \forall y \in X, \quad \mathcal{M}, y \vDash \phi$$

To add time in the model we can extend the logic with the temporal Until operator.

There are many other possible spatial properties in this context. As we will explain below, these type of properties are interesting only for continuous topologies; since as for now we are working only with discrete space, we do not address this topic in detail.

3.3.2 Spatial Logic for Closure Spaces (SLCS)

SLCS, Spatial Logic for Closure Spaces (CLLM14), is a spatial logic that extends the topological semantics of modal logics to closure spaces (Gal99).

The finest topology of a discrete space is the discrete topology where each element is both open and closed. The use of S_4 modal operators as spatial operators in case of this topology becomes useless, as the closure and the internal of a subset will always coincide with itself. Hence, the semantic becomes:

$$\mathcal{M}, x \vDash \Box\phi \iff \mathcal{M}, x \vDash \phi$$

$$\mathcal{M}, x \vDash \Diamond\phi \iff \mathcal{M}, x \vDash U\phi.$$

A possible solution to specify open/closed properties in a discrete space is to use the *closure spaces*. Given a set X, a closure space is a pair $(X, \mathcal{C}l)$

where $\mathcal{C}l : \mathcal{P}(X) \to \mathcal{P}(X)$ is a function s.t., for all $A, B \in \mathcal{P}(X)$, it satisfies the properties: $\mathcal{C}l(\varnothing) = \varnothing$, $\mathcal{C}l(A) \supset A$ and $\mathcal{C}l(A \cup B) = \mathcal{C}l(A) \cup \mathcal{C}l(B)$. A particularity of these spaces is the lack of the *idempotent* property of closure, i.e., $\mathcal{C}l(\mathcal{C}l(A)) \neq \mathcal{C}l(A)$. Hence, if X is the set of nodes of a graph, the closure operator can be seen as the operation that maps a set of nodes A to the set of all nodes that belong to A or that are one step away from an element of A. An important class of Closure space is the class of graphs.

SLCS is equipped with a *closure* modality, $\Diamond$, turning the closure operator $\mathcal{C}l$ into a logical operator. It means that a point x satisfies a formula $\Diamond\varphi$ iff x is an element of the closure of the set of points satisfying φ. The second spatial modality of SLCS is a *spatially* interpreted *until* operator $\mathcal{U}$. Intuitively, this operator describes a situation in which it is not possible to escape an area of points satisfying a certain property, unless by passing through at least one point that satisfies another given formula. This means that a point x satisfies $\varphi_1 \mathcal{U} \varphi_2$ if there is no possibility to "leave" a φ_1-region unless passing by a point that satisfies φ_2. As in the topological case, from these operators, we can also derived some other spatial operators like the *interior* $\Box\phi$ (describing the element in the interior region), the *boundary* $\partial\phi$ (describing the element in the boundary), the *interior boundary* $\partial^{-}\phi$, (describing the elements that satisfy ϕ and that are in the boundary), and the *closure boundary* $\partial^{+}\phi$, (describing the element that satisfy ϕ and that are in the boundary). Furthermore, we can derive other operators that describe *rechability* and *global* or *possible satisfaction* properties. For more details about this logic, we refer to (CLLM14). A first application of that work in the context of smart transportation can be found in (CGL^{+}14).

A temporal extension of SLCS, *Spatio-Temporal Logic of Closure Spaces* (STLCS), combining the logic with the temporal modality of the branching logic CTL, *Computation Tree Logic*, can be found in (CGG^{+}15). STLCS is interpreted on a variant of Kripke models, where valuations are interpreted at points of a closure space. The temporal aspect is treated as a "snapshot" model. The problem with this type of algorithm (to check snapshot models) is that they have very high computational cost. They are susceptible to state-space explosion problems because the spatial for-

mulas have to be recomputed at every state. Furthermore, the logic does not have a stochastic or a quantitative semantics.

3.3.3 Spatel

Spatial-Temporal Logic (HJK+15) (SpaTeL) is the unification of Signal Temporal Logic (MN04) (STL) and Tree-Spatial-Superposition-Logic (TSSL) introduced in (AGBB14) to classify and detect spatial patterns. TSSL reasons over *quad trees* (FB74), spatial data structures that are constructed by recursively partitioning the 2-dimensional space into uniform quadrants. TSSL is derived from Linear Spatial-Superposition-Logic (LSSL) (GSC+09), where the notion of superposition provides a way to describe statistically the distribution of discrete states in a particular partition of the space and the spatial operators correspond to *zooming in and out* of particular areas. This allows one to capture very complex spatial structures, but at the price of a complex formulation of spatial properties, which are in practice only learned from some template image. In (GSC+09), the authors show also that, by nesting these operators, they are able to specify self-similar and fractal-like structures, that generally characterise the patterns emerging in nature. SpaTeL is equipped with a qualitative (yes/no answer) and a quantitative semantics that provides a measure or robustness of how much the property is satisfied or violated. In (HJK+15) this measure of robustness is used as a fitness function to guide the parameter synthesis process for a deterministic reaction diffusion system using particle swarm optimisation (PSO) algorithms. However, the authors do not consider stochastic reaction-diffusion systems and PSO techniques generally do not provide any asymptotically guarantee for reaching the global optimum.

3.3.4 Spatial Logics for Process Algebras

In the following, we present in detail `MoSL` and the Ambient Logic. They are two examples of spatial logics specifically defined to describe properties of process algebra models (Cas01). Process algebrae or process calculus are high-level languages to describe concurrent systems (collec-

tion of agent or processes that can interact, communicate and synchronise). They usually come with specific semantics to translate the high-level model in mathematical models such as CTMC systems.

Mobile Stochastic Language. MoSL(NKL+07) is a logic designed to specify properties of Klaim models (NLK+). StoKlaim is a stochastic extension of Klaim (NFP98) (by the addition of exponential negative distribution rates to each action).

Klaim is a programming language based on process algebra with a Linda-like communication model. As Linda, it has an asynchronous communication, shared memory through tuple spaces and pattern matching. In addition to Linda, it has a number of explicit localities where the processes and the tuple spaces are positioned. Linda primitives are extended by adding the address of the tuple space on which they operate. A Klaim system, called a *net*, is a set of nodes, each of which is identified by a *physical locality* that can be seen as the address of the network. In each node there is a tuple space, a set of processes (running in parallel) and an *allocation environment*. We can refer to the nodes using the physical or the *logical* localities; the latter have only a local meaning (i.e., it depends on where the process is running) and the allocation environment is used to map the logical localities to the physical ones. The principal actions in Klaim are: **out** (to put a tuple), **in** (to take a tuple), **read** (to read a tuple without removing them), **eval** (to deposit and start a process), **newloc** (to create a locality).

MoSL is a *real-time temporal* logic. It is both *action-* and *state*-based and it is *probabilistic*, i.e., it is possible to express the probability that a particular event happens. The atomic propositions are built using a variant of the MoMo (NL04) *consumption* and *production* operators. A *consumption formula* $P@\ell \rightarrow \Phi$ is satisfied if there exists in the network a process P running at site ℓ and the "remaining" network satisfies Φ. We can express also the consumption of tuples using the formula $< F > @\ell \rightarrow \Phi$, where F is a template and the formula is satisfied if there exists in ℓ a tuple that matches the template. A *production formula* $P@\ell \leftarrow \Phi$ is satisfied if the network satisfies Φ whenever a process P is executed at site ℓ. We can

express also the production of tuples using the formula $< f > @\ell \leftarrow \Phi$, where f is a tuple, and the formula is satisfied if the network satisfies Φ whenever a tuple f is stored in ℓ.

MoSL is inspired by aCSL (HKMkS00) (an action-based version of CSL (ASSB96)), hence it has two classes of formulae: *state formulae* and *path formulae*. Path formulae are based on the aCTL *until* operator, $\Phi_\Delta \mathcal{U}_\Omega \Psi$, where Φ and Ψ are state formulae and Δ and Ω are a set of actions. In MoSL, rather than sets of actions, Δ and Ω indicate a set of *action specifiers* $\{\xi_1, .., \xi_n\}$. An action specifier can be seen as a template for actions; for example, the action specifier $\ell_1 : O(< f >, \ell_2)$ is satisfied by any action executed at site ℓ_1 which uploads a tuple $< f >$ to the site ℓ_2. The until operator presents also a time constraint. A path satisfies $\Phi_\Delta \mathcal{U}_\Omega^{\leq t} \Psi$ whenever, within t time units, eventually a state which satisfies Ψ is reached, via a path such that each state satisfies Φ, each executed actions satisfies Δ and the last action satisfies Ω. The state formulae include the atomic propositions, the classic formulae in propositional logic and the probabilistic formulae $P_{\bowtie p}(\varphi)$ and $S_{\bowtie p}(\varphi)$, where $\bowtie \in \{<, >, <=, >=\}$ and $p \in [0, 1]$; the last formula, $S_{\bowtie p}(\varphi)$, corresponds to a steady-state probability property, i.e., for $t \to \infty$. A *numerical model-checker* has been implemented in SAM (CL10) using the model-checker algorithm proposed in (NKL+07). Furthermore, SAM includes also a *statistical model-checker*.

Ambient Logic. Ambient Logic (CG00) was introduced to describe properties of models specified in the Ambient Calculus language (CG98). We chose this logic as an example of a category of logics, designed to express properties of concurrent systems, in particular for mobile processes and mobile ambients. The choice is justified because it gives a good overview of the existing spatial operators in this context. Some of the spatial modalities that we will describe have been directly taken from (e.g., the π-Calculus, (Cai04), (CC03), (LV08))) or are similar to (e.g., MoSL) other process calculi with localities; furthermore, some operators can be applied also to process algebras without locations (CM98).

The Ambient Calculus is a process algebra with localities, designed to study the mobility of concurrent systems. The localities are hierarchically

organized, the resources can be shared privately, i.e., only the processes that know a private/secret location could access to it. The space here can be seen as a tree, where each subtree represents the hierarchy of its root location. Therefore, we do not have a physical structure of space nor we have any information about the relationship between locations that are not nested.

We describe now some of the ambient logic operators, for more details, for example on the derived connectives, we refer to (CG00). Operators can be divided in five classes: those in which the system is designated to a specific location, those in which the system is composed of more the one subsystems, those in which the system restricts the utilization of the resources to a certain subsystems, the spatial modalities that specify the states that may be reach "further away" and the quantified variables. Given a location n and the formulae ϕ, ψ:

- The *location* operator $n[\phi]$ means that ϕ holds at location n. It is satisfied by each process P located in n that satisfies ϕ, i.e., by each $R = n[P]$ s.t. $P \vDash \phi$.

  The *placement* operator $\phi @ n$ is satisfied by each process P that if located in n satisfies ϕ, i.e., by each P s.t. $n[P] \vDash \phi$.

- The *composition* operator $\phi \mid \psi$ means that ϕ and ψ hold contiguously. The formula is satisfied by contiguous processes of the form $P \mid Q$ (where $\mid$ is the classic process algebra parallel operator), i.e., when $\exists R = P \mid Q$, $P \vDash \phi$, and $Q \vDash \psi$.

  The *guarantee* operator $\triangleright$ is used to ensure that a process satisfies a property also when it is a subcomponent. In detail, a process $P \vDash \phi \triangleright \psi$ iff for each Q s.t. $Q \vDash \phi$ we have $Q \mid P \vDash \psi$.

- The *hiding* operator $\diagdown$ is used to restrict a name. $P \vDash \phi \diagdown n$ iff there exists a process P s.t. $(\nu n)P \vDash \phi$, where $(\nu n)P$ is the Ambient Calculus *restriction* operator and means that n is not know outside P, i.e., outside the scope of νn.

The *revelation* operator ® is instead used to reveal a private name. $P \vDash n®\phi$ iff it exists a process Q s.t. $P = (\nu n)Q$ and $Q \vDash \phi$.

- The *somewhere* ($\Diamond$) and *everywhere* ($\Box$) modalities. The "exploration" is done only along the sublocation of a process, i.e., these operators exploit the hierarchical organization of the system. For example $P \vDash \Diamond\phi$ if ϕ holds at some sublocation Q of P.

- The quantified variables range only over name variables x and cannot occur free in the formula but only together with other constructs. A process P satisfies a *universal* quantification $\forall x.\phi$ iff for all names $m \in \Delta$, where Δ is the set of all names, we have that P satisfies $\phi\{x \leftarrow m\}$. For example, if $P \vDash \forall x.(x[T])$ iff $P \vDash m[T]$, $\forall m \in \Delta$, where T means true. The *existential* quantifier $\exists x.\phi$ works in a similar way.

The combination of these operators permits to describe very elaborate properties, but having so many operators could be a problem. Furthermore, all these operators are strictly related to a specific process algebra.

3.3.5 Other Spatial Logics

Other works that present spatial logics as extension of classic temporal logics, have been developed for: networks of processes (RS85), (graph) rewrite theories (BM12), (Mes08), bigraphs (CMS07), and data structure as graphs (CGG02) and heaps (BDL12). Below we discuss the work in (RS85) in more detail.

A Multiprocess Network Logic. An example of extension of classical temporal logics is given by Reif in (RS85). The logic is applied over a directed graph model $\mathcal{M}$. The nodes are processes. With each process is associated an $\omega-$sequence of states and with each state is associated a set of atomic propositions true in that state. In addition to the Boolean and temporal operators there are three spatial modalities: *somewhere*, *everywhere* and symbols called *links* drawn from the set of label, L attached to the edge of the graph. The time is discrete. These spatial operators

permit to relate properties at time $t \in \mathbb{N}$ of a given process P with properties of other processes linked to it at the same time t (two processes are linked if there exists a path that connects them). For example, defining $s_{P,i}$ the i-th state of the process P, calling ϕ a property and L^* the set of all paths between all nodes, the semantic is:

$$s_{P,i} \vDash somewhere\ \phi \iff \exists \alpha \in L^* \text{ s.t. } Q = E(\alpha, P) \text{ and } Q \vDash \phi$$

where $E : L^* \times P \to P$ is a partial function that maps a process P to the process $E(\alpha, P)$ where α is the path that connects the two processes.

Combining the temporal and the spatial operator, we can relate properties of different states of different processes. For example it is easy to describe the provision for broadcasting messages:

hereafter everywhere (M-sent → *everywhere eventually* M-received)

where M-sent and M-received are atomic propositions indicating that a message is sent or received in a state of a process.

Chapter 4

Statistical Methods

In this Chapter, we summarise a number of statistical methods that we exploit in our work: *Statistical Model Checking*, (JCL+09b; YKNP04; YS06), *Gaussian Processes - Upper Confidence Bound Optimisation* (RW06; SKKS12) and *Smoothed Model Checking* (BMS16).

4.1 Statistical Model Checking

The class of algorithms that goes under the name of *Statistical Model Checking* (SMC) is successfully used in the formal methods community to estimate the probability of the satisfaction of a set of linear temporal logic formulae ϕ over a stochastic process. The idea is that, given a PCTMC (or an equivalent stochastic model), with fixed parameters $\boldsymbol{\theta}$, a simulation algorithm is used to sample trajectories of the process, e.g., SSA (Gil77), described in Chapter 2. For each sample, we run a suitable verification algorithm to validate ϕ. In this way, we generate a Bernoulli random variable X_ϕ (equal to 1 iff ϕ is true). Statistical analysis are then performed on X_ϕ to compute the probability distributions of the satisfaction of $P(X_\phi = \texttt{tt})$ or to test if $P(X_\phi = \texttt{tt}) > q$, with a chosen confidence level α. There are two different statistical approaches: frequentist and Bayesian inference. We describe below the latest, for the frequentist approach we refer the interested reader to (JCL+09b; YKNP04; YS06).

The Bayesian SMC (JCL+09a) exploits prior distribution to estimate $P(X_\phi = \texttt{tt})$. We can encode our uncertain knowledge about $P(X_\phi = \texttt{tt})$ by assuming that $p = P(\phi = 1)$ is given by a random variable, whose density function is called the *prior* density. The choice of the prior usually depends on the beliefs and previous experiences about the system. A possible choice is the Beta prior distribution $Beta(p|a, b)$, where the parameters a and b are usually set to 1 and regularise the estimate when a truth value is rarely observed. The Beta distribution is, indeed, the conjugate prior to the Bernoulli distribution. This relationship gives rise to closed-form solutions to the posterior density over p (Bis06). Let n be the total number of observations, h the number of true observations and D_ϕ a sample data then the posterior

$$P(p|D_\phi) = \frac{P(D_\phi|p)P(p)}{P(D_\phi)} = Beta(p, a + h, b + n - h).$$

We can then use the predictive distribution: $P(X_\phi = \texttt{tt}|D_\phi) = E[p|D_\phi] = \frac{h+a}{h+a+n-h+b}$ as prediction of the true probability $P(X_\phi = \texttt{tt})$.

4.2 GP - Upper Confidence Bound Optimisation

In this thesis, we exploit *Gaussian Process* (GP) regression (RW06) to efficiently estimate an unknown objective. Function approximation is a central task in machine learning and statistics. The general regression task can be formulated as follows (Bis06): given a set of input-output pairs $(\boldsymbol{\theta}_i, y_i)$, $i = 1, \ldots, N$ (*training data*), with $\boldsymbol{\theta}_i \in \mathbb{R}^d$ and $y_i \in \mathbb{R}$, determine a function $f: \mathbb{R}^d \to \mathbb{R}$ s.t. $f(\boldsymbol{\theta}_i)$ is optimally close to the target values y_i (usually in terms of minimising a suitable loss function). Several methods exist for addressing this task; we adopt a Bayesian perspective: we specify a prior distribution over a suitable function space, and condition on the observed values to obtain a posterior estimation of the function value at all possible input points. GPs are flexible non-parametric distributions over spaces of functions which can be used as prior distributions in a Bayesian framework, where the input-output pairs represent noisy observations of the unknown function. This enables a natural quantifi-

cation of the uncertainty of the estimated function at every new input value; this uncertainty will play a central role in the optimal design strategy we propose in Chapter 5. We describe below in detail the definition of GP and the optimisation technique.

Gaussian Process. Formally, a GP over a subset $\mathcal{X} \subseteq \mathbb{R}^d$ is a collection of random variables indexed by $\boldsymbol{\theta} \in \mathcal{X}$, any finite number of which have a joint Gaussian distribution, and it is completely defined by its *mean* $\mu: \mathcal{X} \to \mathbb{R}$ and its *covariance* $k: \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ functions. We denote a sample from a GP with mean function μ and covariance function k as

$$f \sim \mathcal{GP}(\mu, k).$$

Gaussian Process Regression. Denote again as $(\boldsymbol{\theta}_i, y_i), \quad i = 1, .., N$ our observations, and let $p(y_i|f(\boldsymbol{\theta}_i))$ denote observation error model. From the previous definition we have

$$f \sim GP(\mu, K) \leftrightarrow f = (f(\boldsymbol{\theta}_1), ..., f(\boldsymbol{\theta}_N)) \sim N(\boldsymbol{\mu}, K),$$

where $\boldsymbol{\mu}$ is the vector of evaluations of μ at every $\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_N$, and K is the matrix obtained evaluating the covariance matrix at all pairs of training inputs. A case of particular interest arises when the observation error model $p(y_i|f(\boldsymbol{\theta}_i)) = \mathcal{N}(0, \sigma^2)$ is Gaussian itself. In this case, the inference can be computed analytically to yield a closed form for the predictive posterior.

In the regression procedure, the combination of the prior Gaussian process $GP(\mu, k)$ and the training set of points $(\boldsymbol{\theta}_i, y_i), i = 1, .., N$ (where in our case y_i is the noisy evaluation of the unknown function computed at $\boldsymbol{\theta}_i$) leads to a posterior distribution $GP(\mu_{new}, k_{new})$ over the space of functions, which represents our knowledge of the values of f at a further input point $\boldsymbol{\theta}_*$. In particular, let be $\mathbf{k}_* = k(\boldsymbol{\theta}_*, \boldsymbol{\theta}_1), \ldots, k(\boldsymbol{\theta}_*, \boldsymbol{\theta}_N)$ the vector obtained evaluating the covariance matrix at the new input and at all training input, and as $k_{**} = K(\boldsymbol{\theta}_*, \boldsymbol{\theta}_*)$ the prior variance at the new input

point, we have that

$$\begin{aligned} \mu_{new}(f(\boldsymbol{\theta}_*)) &= \mathbf{k}_*^T \left(K + \sigma^2 I\right)^{-1} \mathbf{y} \\ k_{new}(f(\boldsymbol{\theta}_*) &= k_{**} - \mathbf{k}_*^T \left(K + \sigma^2 I\right)^{-1} \mathbf{k}_*. \end{aligned} \tag{4.1}$$

As it is clear from the equations for μ_{new} and k_{new}, the choice of the covariance function $k(x, x')$ is fundamental in the modelling decision, as it defines the behaviour of the posterior GP and thus the type of functions that emulate the wanted unknown function f. In this work, we will exploit the popular Radial Basis Function (RBF)

$$k(x, x') = \gamma \exp[-\frac{|x - x'|^2}{\delta^2}]$$

where the amplitude γ and the lengthscale δ are two hyper-parameters that control how much and how fast a sampled function changes in time. The success of this particular covariance lays in the fact that this kernel enjoys a universality property (Ste02) that ensures the existence of an optimal emulation of our unknown function, and moreover it identifies with probability one infinitely differentiable functions. For a good review of Gaussian Process Regression see (RW06) .

The input-output pairs in a regression task are often different features of experimentally observed data points. In our work, the output points correspond to true functional evaluations of an unknown (and analytically intractable) function of the inputs. In this case, the regression task is often given the special name of *emulation* in the statistics literature.

In our analysis, we can only obtain a sampling approximation through statistical methods such as SMC (described in the previous section). This means that our function evaluations are noisy; by virtue of the Central limit theorem we can assume that, provided sufficient samples, the noise will be approximately Gaussian. [1] This therefore enables us to obtain an analytical estimate of the posterior process (RW06).

GP - UCB algorithm. We now address the task of maximising the unknown function. Let $\mu_N(\boldsymbol{\theta})$ and $k_N(\boldsymbol{\theta})$ be the mean and variance of the

[1] Note that here we approximate as a GP a fitness score as a function of parameters. We are not imposing any (Gaussian) approximation of the process itself.

GP emulator at a given point in input space $\boldsymbol{\theta}$ (recall that the marginal at any point will be Gaussian). At each iteration of the algorithm, the solution is searched among the set of observations $(\boldsymbol{\theta}_1, y_1), ..., (\boldsymbol{\theta}_N, y_N)$, and an intuitive approach to the optimisation problem would be to maximise the posterior mean $\mu_N(\boldsymbol{\theta})$, exploring the region (i.e., choosing the next input) near the point $\boldsymbol{\theta}_{N+1} = \arg\max_{\boldsymbol{\theta}}[\mu_N(\boldsymbol{\theta})]$. This procedure, though, is evidently vulnerable to local optima: the emulated function is estimated based on relatively few function evaluations, so that, while the emulator typically provides a good approximation of the true function near the sampled points, regions of parameter space far from the sampled points may contain the true maximum undetected. Using the language of reinforcement learning, maximising the emulated function would privilege exploitation (i.e., using currently available information) at the expense of exploration. Obviously, given sufficient computational power, one may consider sampling many parameter points so as to have sufficient coverage of the whole region of interest; this strategy is however bound to fail in even moderate dimensions due to the curse of dimensionality. An elegant solution to the above conundrum can be obtained by also considering the *uncertainty* of the emulated function (which is also computed analytically in GP regression): intuitively, one should explore regions where the maximum *could plausibly be*, i.e., regions in parameter space where there is substantial posterior probability mass for the function to take a high value. This is done by the GP-Upper Confidence Bound (GP-UCB) algorithm (SKKS12); indeed, it takes into account also the posterior covariance $k_N(\boldsymbol{\theta})$, that eventually allows the investigation of new regions (far from the current input points) where the global maximum may be hidden; this means that it maximises an upper quantile of the distribution (e.g., the 95% quantile). The rule for selecting the next input point to be added to the set of observations is

$$\boldsymbol{\theta}_{N+1} = \text{argmax}_{\boldsymbol{\theta}} \left(\mu_N(\boldsymbol{\theta}) + \beta_N \sqrt{k_N(\boldsymbol{\theta})} \right) \tag{4.2}$$

where β_N is a constant that depends on the iteration in such a way that it ensures convergence. This allows us to recast the optimisation problem balancing the exploitation of promising regions (where the emulation has

high values) with the exploration of new regions (where the emulation is very uncertain, and hence high values may be hidden). The algorithm stops whenever no further improvements can be made (i.e., the evaluations y_i over the new inputs resemble the ones comprising the current set of observations), or we reach a prefixed maximum number of iterations. The convergence of the algorithm to the global optimum was theoretically proven in (SKKS12).

4.3 Smoothed Model Checking

The standard Statistical Model Checking (SMC) techniques, described in Section 4.1, are usually applied to models with fixed parameters, their application over a range of parameter values requires then a prohibitive number of simulations. The *Smoothed Model Checking* (BMS16) is a novel SMC technique that permits to verify simultaneously a property, for a range of parameters values of an uncertain stochastic model, i.e., a model for which the parameters values are not all known exactly. It relies on the characterisation of the satisfaction probability of MITL formula ϕ for these kind of models. The method leverages the smoothness of the truth probability as function of the model parameters, under some conditions, and transfers the information across nearby parameter values. From this information, it designs an analytical approximation of such probability. This allows the computation of the approximate probability for all values of the uncertain parameters, through the evaluation of the satisfaction on few parameter values only.

Let us now describe the procedure in detail. An uncertain CTMC, $\mathcal{M}_{\boldsymbol{\theta}}$, is a family of CTMCs, defined in Chapter 2, whose transition rates, $\boldsymbol{\tau}(\mathbf{X}, \boldsymbol{\theta})$, depend also on a set of parameters $\boldsymbol{\theta} \in \boldsymbol{K} \subseteq \mathbb{R}^d$. Given an uncertain CTMC $\mathcal{M}_{\boldsymbol{\theta}}$, the satisfaction function of a MITL formula ϕ is a function $f : \boldsymbol{K} \to [0, 1]$, such that :

$$f(\boldsymbol{\theta}) \equiv p(\phi = \mathbf{true} | \mathcal{M}_{\boldsymbol{\theta}}), \tag{4.3}$$

i.e., $f(\boldsymbol{\theta})$ is the probability that ϕ is true conditionally on the model having parameters $\boldsymbol{\theta}$. The goal of the smoothed model checking algorithm is

to statistically estimate the satisfaction function $f(\boldsymbol{\theta})$, evaluating such function (via monitoring of the formula on samples) only for a small set of parameter values, the *training data* set, $(\hat{\boldsymbol{\theta}}, \hat{\mathbf{f}}) = \{(\boldsymbol{\theta}_i, f(\boldsymbol{\theta}_i)), i = 1, \cdots, N\}$.

It has been proven in (BMS16) that, if the transition rates $\boldsymbol{\tau}(\mathbf{X}, \boldsymbol{\theta})$ depend smoothly on the parameters $\boldsymbol{\theta}$ and polynomially on the state of the system $\mathbf{X}$, then$f(\boldsymbol{\theta}) \in C^{\infty}(\boldsymbol{K})$, meaning that the satisfaction function of ϕ is a smooth function of the parameters. The smoothness of $f(\boldsymbol{\theta})$ permits to exploit the Gaussian Process (GP) regression, described in the previous section, to efficiently emulate the unknown function.

The Gaussian Process (GP) regression adopts a Bayesian perspective. Given a Gaussian prior distribution and a training set $(\hat{\boldsymbol{\theta}}, \hat{\mathbf{f}})$, we can evaluate the mean and the covariance at a new input value $\boldsymbol{\theta}_*$, that corresponds to the gaussian distribution $p(f(\boldsymbol{\theta}_*), f(\boldsymbol{\theta}_1), \cdots, f(\boldsymbol{\theta}_N))$.

Combining this distribution with the *likelihood* functions $p(\hat{f}_i | f(\boldsymbol{\theta}_i)), i = 1, \cdots, N$ and applying the Bayes' theorem, we can obtain the joint posterior:

$$p(f(\boldsymbol{\theta}_*), f(\boldsymbol{\theta}_1), \cdots, f(\boldsymbol{\theta}_N) | \hat{\mathbf{f}}) := \frac{1}{C} p(f(\boldsymbol{\theta}_*), f(\boldsymbol{\theta}_1), \cdots, f(\boldsymbol{\theta}_N)) \prod_i p(\hat{f}_i | f(\boldsymbol{\theta}_i)),$$

where C is a normalisation constant. The desired posterior $p(f(\boldsymbol{\theta}_*) | \hat{\mathbf{f}})$ is then obtained marginalising the joint posterior:

$$p(f(\boldsymbol{\theta}_*) | \hat{\mathbf{f}}) = \int \prod_{i=1}^{N} df(\boldsymbol{\theta}_i) p(f(\boldsymbol{\theta}_*), f(\boldsymbol{\theta}_1), \cdots, f(\boldsymbol{\theta}_N) | \hat{\mathbf{f}}). \tag{4.4}$$

The *likelihood* function $p(\hat{f}_i | f(\boldsymbol{\theta}_i))$ describes how the probability of the actual observations $\hat{\boldsymbol{\theta}}$ depends on the unknown true value at that input points. In the smoothed model checking, the observations $f(\boldsymbol{\theta}_1), \cdots, f(\boldsymbol{\theta}_N)$ are computed using a Binomial random variable $B(m, f(\boldsymbol{\theta}_i))$, where m is the number of observations of each parameters set of the training set. It means that the algorithm generates and monitors m samples for each parameters set $\boldsymbol{\theta}_i$ of the training set, m usually is a small number.

Finally, the procedure computes an approximation of the GP posterior distribution $p(f(\boldsymbol{\theta}_*) | \hat{\mathbf{f}})$, using and Expectation Propagation (EP) ap-

proach (Min13), a variational procedure producing an analytical approximation of the integral (4.4),(RW06). In this way, we obtain an analytical approximation of the satisfaction function. This implies that the satisfaction probability can be estimated at any point in the parameter space with no additional cost. Furthermore, evaluating the statistics of the induces posterior distribution we obtain also a confidence interval.

The power of this method is that usually fewer samples are required to achieve a given level of accuracy, at any single point, than just by using standard SMC. In the experiments of (BMS16), it has been possible to accurately approximate the satisfaction function over a wide range of parameters using less than 10% of the simulation runs required to obtain the same result with exhaustive parameter exploration by classic SMC. This resulted in a decrease of the total analysis time nearly by 90%.

Part II

Contribution

Chapter 5

System Design via Robustness Maximisation

In this chapter, we present a novel notion of *robustness* for temporal properties of stochastic models and its application in the context of the *system design problem*, then we apply the new framework to a number of case studies. The work has been published in (BBNS13; BBNS15).

5.1 Robustness of Models

A classical question in formal modelling is how to compute the probability that a behaviour, expressed in terms of a certain temporal logic formula, may occur in a given stochastic process, modelled, for example, as a Continuous Time Markov Chain or as a Stochastic Hybrid Automaton (see Chapter 2), with fixed parameters. *Probabilistic Model Checking* (BCHG+97; BHHK03) is a well-established verification technique that provides an answer to such a question, i.e., the probability of a property being true. However, especially when we deal with stochastic models, the notion of satisfiability may not be enough to determine the capacity of a system to maintain a particular emergent behaviour, unaffected by the uncertainty of the perturbations due to its stochastic nature or by possible small changes in the model parameters.

In case of deterministic dynamical systems, which may be subject to extrinsic noise or uncertainty in the parameter, researchers from the verification community have proposed several notions of *temporal logic based robustness* (DM10; FP09; RBFS08). We described this notion in Chapter 3, showing how it provides a measure of distance between a trajectory of a system and the behavioural property of interest, expressed in terms of a temporal logic formula. This effectively endows the logic of interest with a quantitative semantics, as we have seen for *Signal Temporal Logic*, allowing us to capture not only whether a property is satisfied but also *how much* it is satisfied. Below, we formalise a similar notion of robustness for stochastic models.

5.2 Stochastic Semantics of STL

Here, starting from the quantitative semantics of the *Signal Temporal Logic*, STL, described in Chapter 3, we present a formal definition of the notion of robustness for stochastic systems, showing that this naturally leads to a distribution of robustness scores.

Consider a STL formula ϕ, with predicates interpreted over state variables of a stochastic process $(\mathbf{X}(t))_{t\in T} = (X_1(t), ..., X_n(t))_{t\in T}$, where each vector $\mathbf{X}(t)$ corresponds to the state of the system at time t. We assume $\mathbf{X}(t) \in \mathbb{D}$, where $\mathbb{D}$ is the *state space* of the system. For simplicity, we indicate the stochastic model with $\mathbf{X}(t)$.

As described in Chapter 2, $\mathbf{X}(t)$ can be seen also as a random variable $\mathbf{X}$ on the space $\mathbb{D}$-valued *cadlag functions* $\mathcal{D}([0,\infty), \mathbb{D})$, which in this section we denote by $\mathcal{D}$, assuming the domain $\mathbb{D}$ to be fixed. It means that the set of trajectories $\mathbf{x}$ of the stochastic process $\mathbf{X}$ is represented by the set $\mathcal{D}$.

The Boolean semantics of ϕ is readily extended to stochastic models as customary, by measuring the probability of the set of trajectories that satisfy the formula

$$P(\phi) = \mathbb{P}\{\mathbf{x} \in \mathcal{D} \mid \mathbf{x} \vDash \phi\}.$$

BooleanThis is true for CTMC while for SHA or SDE more refined constructions are needed, see for instance (Dav93; Bil99). Furthermore, the set of trajectories that satisfy/falsify a formula is a measurable set (ZPC10), so that we can safely talk about its probability.

In order to extend this definition to the robustness score, it is convenient to think of the set of trajectories that satisfy ϕ as a measurable function $I_\phi : \mathcal{D} \to \{0,1\}$, such that $I_\phi(\mathbf{x}) = 1$ if and only if $\mathbf{x} \vDash \phi$. Then, we can define the random variable $I_\phi(\mathbf{X})$ on $\{0,1\}$ induced by $\mathbf{X}$ via I_ϕ as the Bernoulli random variable which is equal to 1 with probability $P(\phi)$. We can equivalently write:

$$\mathbb{P}(I_\phi(\mathbf{X}) = 1) = \mathbb{P}(\{\mathbf{x} \in \mathcal{D} \mid I_\phi(\mathbf{x}) = 1\}) = \mathbb{P}(I_\phi^{-1}(1)).$$

We can extend the robustness score to stochastic models in a similar way: given a trajectory $\mathbf{x}$, we can compute its robustness score, according to the quantitative semantics (Def. 3.3, Chapter 3), and interpret $\rho(\phi, \mathbf{x}, 0)$ as a functional $\mathtt{R}_\phi : \mathcal{D} \to \mathbb{R}$. To propagate the distribution of $\mathbf{X}$ to $\mathbb{R}$, we need then to show that $\mathtt{R}_\phi$ is measurable.

Theorem 5.1 *For any STL formula ϕ, with atomic predicates defined by continuous functions, the functional $\mathtt{R}_\phi : \mathcal{D} \to \mathbb{R}$ is measurable, with respect to the Borel σ-algebra of the topology of $\mathcal{D}$ induced by the Skorokhod metric*

In order to prove theorem 5.1, we need some preliminary results about measurability of the functions involved in the definition of the robustness score ρ.

We recall that the space of cadlag functions $\mathcal{D}([0,\infty), \mathbb{D})$ is given the structure of a metric space by the *Skorokhod metric*, also described in Chapter 2. Furthermore, the Skorohod metric defines a topology for which $\mathcal{D}([0,\infty), \mathbb{D})$ is complete and separable, i.e., it is a Polish space.

We first note here that a sequence of functions $\mathbf{x}^n \in \mathcal{D}([0,T], \mathbb{D})$ converges to $\mathbf{x} \in \mathcal{D}([0,T], \mathbb{D})$ if and only if there is a sequence of time-wiggle functions $\omega^n \in \mathcal{I}_T$ satisfying $\sup_{t\in[0,T]} \|\omega^n(t) - t\| \to 0$ and $\sup_{t\in[0,T]} \|\mathbf{x}^n(t) - \mathbf{x}(\omega^n(t))\| \to 0$.

We also recall an important property of cadlag functions, which essentially allows us to approximate any cadlag function with a step function.

Lemma 5.1 *Let* $\mathbf{x} \in \mathcal{D}([0,T],\mathbb{D})$. *For each* $\varepsilon > 0$, *there exists a finite grid* $\mathbb{T} = \{0 = t_0 < t_1 < t_2 < \ldots < t_k = T\}$ *such that for each* $i = 0, \ldots, k-1$

$$\sup_{t,s \in [t_i, t_{i+1})} \|\mathbf{x}(t) - \mathbf{x}(s)\| < \varepsilon$$

Proof: See (Bil99). ∎

As we always deal with time bounded signals, we start by considering ρ, for a given formula ϕ, as a transducer of real-valued signals, i.e., as a functional from $\mathcal{D}([0,T],\mathbb{D})$ to itself. To be more precise, we need to take into account that a STL formula looks T_ϕ time units into the future[1], hence $\hat{\mathtt{R}}_\phi : \mathcal{D}([0,T],\mathbb{D}) \to \mathcal{D}([0,T-T_\phi],\mathbb{D})$ (where $\hat{\mathtt{R}}$ denotes the functional between cadlag function spaces associated with the robustness score ρ).

Lemma 5.2 *Consider the space* $\mathcal{D}([0,T],\mathbb{D})$ *of cadlag functions. The following functionals are measurable, with respect to the Borel* σ*-algebra induced by the Skorokhod topology (or the product* σ*-algebra).*

a) Forward time shifts: $\delta_a : \mathcal{D}([0,T],\mathbb{D}) \to \mathcal{D}([0,T-\hat{T}],\mathbb{D})$, *defined by* $\delta_a(\mathbf{x})(t) = \mathbf{x}(t+a)$, *for* $\hat{T} > a$. *Similarly for backward time shifts.*

b) Pointwise maximum (and minimum): $\max : \mathcal{D}([0,T],\mathbb{D}) \times \mathcal{D}([0,T],\mathbb{D}) \to \mathcal{D}([0,T],\mathbb{D})$.

c) Maximum (and minimum) over a dense interval in the future: $SUP_{[T_1,T_2]} : \mathcal{D}([0,T),\mathbb{D}) \to \mathcal{D}([0,T-T_2),\mathbb{D})$, *defined*[2] *as*

$$SUP_{[T_1,T_2]}(\mathbf{x})(t) = \sup_{\tau \in t \oplus [T_1,T_2]} x(\tau),$$

for $0 \le T_1 < T_2 < T$ *fixed.*

Proof: To prove points a) and b), we rely on the fact that the Borel σ-algebra in $\mathcal{D}([0,T],\mathbb{D})$ is generated by the following collection of sets (forming a π-system): $\mathcal{A}_T = \{\pi^{-1}_{t_1,\ldots,t_k}(\mathbf{h}) \mid k \in \mathbb{N}, \mathbf{h} \in \mathbb{D}^k, 0 \le t_1 < \ldots < t_k \le T\}$, where $\pi_{t_1,\ldots,t_k}$ is the projection on $\mathbb{D}^k$, which is measurable (Bil99).

[1] T_ϕ is defined recursively by $T_\mu = 0$, $T_{\phi_1 \wedge \phi_2} = \max\{T_{\phi_1}, T_{\phi_2}\}$, $T_{\neg\phi} = T_\phi$, and $T_{\phi_1 \mathcal{U}_{[T_1,T_2]} \phi_2} = \max\{T_{\phi_1}, T_{\phi_2}\} + T_2$.

[2] Recall that $\oplus$ is the Minkowski sum of two sets, $A \oplus B = \{a + b \mid a \in A, b \in B\}$

a) We just need to prove that $\delta_a^{-1}(\pi^{-1}_{t_1,\dots,t_k}(\mathbf{h}))$ is measurable for each $\pi^{-1}_{t_1,\dots,t_k}(\mathbf{h}) \in \mathcal{A}_{T-\hat{T}}$. But $\delta_a^{-1}(\pi^{-1}_{t_1,\dots,t_k}(\mathbf{h})) = \pi^{-1}_{t_1+a,\dots,t_k+a}(\mathbf{h})$.

b) Denote with $\mathbf{b}$ a Boolean tuple of length k and with $\bar{\mathbf{b}}$ its element-wise Boolean complement. Define $\pi^{-1}_{t_1,\dots,t_k,\mathbf{b}}(\mathbf{h})$ to be the set

$$\left[\bigcap_{i:\mathbf{b}[i]\ true} \pi^{-1}_{t_i}(\{h_i\})\right] \cap \left[\bigcap_{i:\mathbf{b}[i]\ false} \pi^{-1}_{t_i}((-\infty, h_i))\right]$$

which is measurable (by measurability of finite dimensional projections). It holds that

$$\max^{-1}(\pi^{-1}_{t_1,\dots,t_k}(\mathbf{h})) = \bigcup_{\mathbf{b}} \pi^{-1}_{t_1,\dots,t_k,\mathbf{b}}(\mathbf{h}) \times \pi^{-1}_{t_1,\dots,t_k,\bar{\mathbf{b}}}(\mathbf{h}),$$

where the union is taken over all possible Boolean tuples of length k. Hence, the set $\max^{-1}(\pi^{-1}_{t_1,\dots,t_k}(\mathbf{h}))$ is measurable in the product σ-algebra.

c) We prove this only for the maximum, as the result for the minimum follows similarly.

For each n, define the finite grid $\mathbb{T}_n = \{T_1, T_1 + \delta_n, \dots, T_2\}$, where $\delta_n = \frac{T_2 - T_1}{n}$. Furthermore, let

$$SUP^n_{[T_1,T_2]}(\mathbf{x})(t) = \max_{\tau \in t \oplus \mathbb{T}_n} x(\tau).$$

Fix $\mathbf{x} \in \mathcal{D}([0,T), \mathbb{D})$, and call $g^n(t) = SUP^n_{[T_1,T_2]}(\mathbf{x})(t)$ and $g(t) = SUP_{[T_1,T_2]}(\mathbf{x})(t)$. We will prove that $g^n \to g$ in the Skorokhod metrics. By additionally showing the measurability of $SUP^n_{[T_1,T_2]}$ for each n, we can rely on a classic result for measurable functions, i.e., that the pointwise limit of a sequence of measurable functions is measurable (Bil12), to prove the measurability of $SUP_{[T_1,T_2]}$.

Measurability of $SUP^n_{[T_1,T_2]}$. Call $\mathbf{x}_i(t) = \mathbf{x}(t + i \cdot \delta_n)$ and observe that $SUP^n_{[T_1,T_2]}(\mathbf{x})(t) = \max\{\mathbf{x}_0(t), \dots, \mathbf{x}_n(t)\}$. The measurability of $SUP^n_{[T_1,T_2]}$ follows from points a) and b) above (extending point b to the maximum of n functions is straighforward).

Convergence of g^n to g. Fix $\varepsilon > 0$ sufficiently small and use Lemma 5.1 for $\mathbf{x}$ on $[0,T]$ to find a grid $\mathbb{T}_{\mathbf{x},\varepsilon} = \{t_0, t_1, \ldots, t_h\}$ in $[0,T]$ satisfying the condition in the Lemma for the given ε. Fix a closed interval $I \subseteq [0,T]$ and consider any finite set $\hat{\mathbb{T}}_I$ that contains one point $\hat{t} \in [t_j, t_{j+1}) \cap I$ for each $[t_j, t_{j+1}) \cap I \neq \varnothing$. By splitting the supremum in each $[t_j, t_{j+1})$, Lemma 5.1 implies that

$$\sup_{\tau \in I} \mathbf{x}(\tau) \leq \max_{\hat{t} \in \hat{\mathbb{T}}} \mathbf{x}(\hat{t}) + \varepsilon. \tag{5.1}$$

Now let $\delta_{\mathbb{T}_{\mathbf{x},\varepsilon}} = \min_{t_j \in \mathbb{T}_{\mathbf{x},\varepsilon}} (t_{j+1} - t_j)$ be the smallest step size of $\mathbb{T}_{\mathbf{x},\varepsilon}$ and choose n_0 such that $\delta_{n_0} = \frac{T_2 - T_1}{n_0} < \delta_{\mathbb{T}_{\mathbf{x},\varepsilon}}/2$. It then follows that $t \oplus \mathbb{T}_n$ contains at least one point in each of the intervals $[t_j, t_{j+1})$ of $\mathbb{T}_{\mathbf{x},\varepsilon}$, hence (5.1) implies (uniformly in t) that

$$g(t) - \varepsilon < g^n(t) < g(t).$$

Concluding, we found n_0 such that, for all $n > n_0$, $d_T(g^n, g) < \varepsilon$, which implies the convergence of g^n to g in $\mathcal{D}([0,T], \mathbb{D})$. ∎

Lemma 5.3 *Let ϕ a STL formula. The functional $\hat{\mathtt{R}}_\phi$ associated with it, $\hat{\mathtt{R}}_\phi : \mathcal{D}([0,T], \mathbb{D}) \to \mathcal{D}([0, T - T_\phi], \mathbb{D})$, is measurable.*

Proof: We proceed by structural induction on the formula ϕ.

Atomic predicate μ. Let μ be defined by the function $y(\mathbf{x}[t])$, required to be at least continuous. As the pointwise extension to $\mathcal{D}([0,T], \mathbb{D})$ of a continuous function is a continuous functional, $\hat{\mathtt{R}}_\mu$ is measurable.

Negation $\neg\phi$. $\hat{\mathtt{R}}_{\neg\phi} = -\hat{\mathtt{R}}_\phi$ is measurable by inductive hypothesis (and continuity of the function $-x$).

Conjunction $\phi_1 \wedge \phi_2$. $\hat{\mathtt{R}}_{\phi_1 \wedge \phi_2} = \min\{\hat{\mathtt{R}}_{\phi_1}, \hat{\mathtt{R}}_{\phi_2}\}$, which is measurable in virtue of Lemma 5.2 b) and of the fact that measurability is preserved when composing measurable functions.

Eventually $\mathcal{F}_{[T_1,T_2]}\phi$. $\hat{\mathtt{R}}_{\mathcal{F}_{[T_1,T_2]}\phi} = \sup_{[T_1,T_2]}(\hat{\mathtt{R}}_\phi)$ is measurable in virtue of the measurability of $\hat{\mathtt{R}}_\phi$ (structural induction) and of Lemma 5.2 c).

Globally $\mathcal{G}_{[T_1,T_2]}\phi$**.** $\hat{\mathtt{R}}_{\mathcal{G}_{[T_1,T_2]}\phi} = \inf_{[T_1,T_2]}(\hat{\mathtt{R}}_\phi)$ is also measurable for the same reason above.

Until $\phi_1 \mathcal{U}_{[T_1,T_2]}\phi_2$**.** By definition,

$$\begin{aligned}\hat{\mathtt{R}}_\phi(\mathbf{x})(t) = \hat{\mathtt{R}}_\phi(\mathbf{x})(t) &= \hat{\mathtt{R}}_{\phi_1 \mathcal{U}_{[T_1,T_2]}\phi_2}(\mathbf{x})(t) \\ &= \sup_{t' \in t \oplus [T_1,T_2]} \{\min\{\hat{\mathtt{R}}_{\phi_2}(\mathbf{x})(t'), \inf_{\tau \in [t,t']} \hat{\mathtt{R}}_{\phi_2}(\mathbf{x})(\tau)\}\}.\end{aligned}$$

Measurability follows from a technical argument similar to the one of Lemma 5.2 c), combined with the measurability of $\hat{\mathtt{R}}_{\phi_1}(\mathbf{x})$ and $\hat{\mathtt{R}}_{\phi_2}(\mathbf{x})$ by inductive hypothesis. More specifically, we need to construct a sequence of convergent approximations $R^n(\mathbf{x})(t)$ of

$$\hat{\mathtt{R}}_{\phi_1 \mathcal{U}_{[T_1,T_2]}\phi_2}(\mathbf{x})(t),$$

computed on a discrete grid that shifts with t and t'. The discrete grids, like in Lemma 5.2 c), have to be independent from the specific $\mathbf{x}$, while the convergence must be uniform in t, but it can depend on x (we need to prove only pointwise convergence).

More specifically, let $\mathbb{T}^n = \{T_1, T_1 + \delta_n, \ldots, T_2\}$ for $\delta_n = (T_2 - T_1)/n$, and $\mathbb{T}_1^n = \{0, \delta_n^1, \ldots, T_2\}$, for $\delta_n^1 = T_2/n_1 \leq \delta_n$. Then the finite approximation of $\hat{\mathtt{R}}_\phi(\mathbf{x})(t)$ is

$$\mathtt{R}^{\mathtt{n}}(\mathbf{x})(\mathtt{t}) = \max_{\mathtt{t}' \in \mathtt{A}_{\mathtt{n}}} \{\min\{\hat{\mathtt{R}}_{\phi_2}(\mathbf{x})(\mathtt{t}'), \min_{\tau \in \mathtt{B}_{\mathtt{n}}} \hat{\mathtt{R}}_{\phi_2}(\mathbf{x})(\tau)\}\},$$

with $A_n = t \oplus T^n$ and $B_n = (t \oplus \mathbb{T}_1^n \cup t \ominus \mathbb{T}_1^n) \cap [t, t']$. The definition of B_n keeps into account the fact that both t and t' can vary, and it is needed to ensure that there is an n_0 such that, for $n \geq n_0$ we find for any t, t' points of B_n in a fixed but arbitrary finite partition of $[0, T]$. The measurability of $\mathtt{R}^{\mathtt{n}}$ as a functional follows from similar arguments than those in Lemma 5.2 c).

Now, fix $\varepsilon > 0$ and an element $\mathbf{x}$ in $\mathcal{D}([0,T], \mathbb{D})$. Construct a finite partitioning $\mathbb{T}_{\mathbf{x},\varepsilon}$ of $[0,T]$ such that Lemma 5.1 is satisfied both for $\hat{\mathtt{R}}_{\phi_1}(\mathbf{x})$ and $\hat{\mathtt{R}}_{\phi_2}(\mathbf{x})$ for $\varepsilon/2$. Then it is easy to check that, if n is such that δ_n is smaller than half the step $\delta_{\mathbb{T}_{\mathbf{x},\varepsilon}} = \min_{t_j \in \mathbb{T}_{\mathbf{x},\varepsilon}}(t_{j+1} - t_j)$ of

$\mathbb{T}_{\mathbf{x},\varepsilon}$, then A_n and B_n contain points of each interval of $\mathbb{T}_{\mathbf{x},\varepsilon}$, so that the minimum approximates the infimum with an error bounded by $\varepsilon/2$ for each t and t', and the maximum cumulates another $\varepsilon/2$ approximation error with respect to the supremum. It follows that the distance between $\mathtt{R}^{\mathtt{n}}(\mathbf{x})(\mathtt{t})$ and $\hat{\mathtt{R}}_\phi(\mathbf{x})(t)$ is no more than ε, uniformly in t, showing the convergence of the approximation in the Skorokhod metric. ∎

We can finally prove Theorem 5.1. Recall that $\mathtt{R}_\phi : \mathcal{D}([\mathtt{0},\mathtt{T}],\mathbb{D}) \to \mathbb{R}$ is defined by $\mathtt{R}_\phi(\mathbf{x}) = \rho(\phi,\mathbf{x},\mathtt{0})$. Given a formula ϕ, let $\mathcal{D} = \mathcal{D}([0,T],\mathbb{D})$ for some $T > T_\phi$.

Proof of Theorem 5.1. The theorem follows form the fact that $\mathtt{R}_\phi = \pi_0 \circ \hat{\mathtt{R}}_\phi$, i.e., it is the composition of the measurable functional $\hat{\mathtt{R}}_\phi$ (Lemma 5.3) with the measurable projection π_0. ∎

In virtue of Theorem 5.1, $\mathtt{R}_\phi$ induces a real-valued random variable $R_\phi = \mathtt{R}_\phi(\mathbf{X})$ with probability distribution given by

$$\mathbb{P}\left(\mathtt{R}_\phi(\mathbf{X}) \in [\mathtt{a},\mathtt{b}]\right) = \mathbb{P}\left(\mathbf{X} \in \{\mathbf{x} \in \mathcal{D} \mid \rho(\phi,\mathbf{x},0) \in [a,b]\}\right)$$

In other words, if we apply the definition of robustness to a stochastic model, we obtain a distribution of robustness degrees. This distribution tells us much more than the standard probabilistic semantics, because it tells us "how much" a formula is true.

In particular, here we will be interested in some statistics of this distribution, specifically the average robustness degree, and the average robustness conditional on a formula being true or false. The first quantity gives a measure of how strongly a formula is satisfied on average. The larger this number, the more robust is satisfaction. Most of the times, this number will be correlated with the satisfaction probability, yet we can have a large average satisfaction score even for a small probability of satisfaction. Better indicators of the intensity of satisfaction and dissatisfaction are the conditional averages, $\mathbb{E}(R_\phi \mid R_\phi > 0)$ and $\mathbb{E}(R_\phi \mid R_\phi < 0)$. These are related to the average by the equation

$$\mathbb{E}(R_\phi) = P(\phi)\mathbb{E}(R_\phi \mid R_\phi > 0) + (1 - P(\phi))\mathbb{E}(R_\phi \mid R_\phi < 0)$$

which holds provided $\mathbb{P}(R_\phi = 0)$ is zero.

In the next sections, we investigate to what extent these three synthetic indices are good descriptors of the robustness distribution, and how they can be exploited to do parameter synthesis for stochastic models. Before that, we discuss the continuity of $\mathtt{R}_\phi$.

Continuity of $\mathtt{R}_\phi$ An interesting question about the robustness score $\mathtt{R}_\phi$ of a formula ϕ is whether it is continuous as a functional on the space of trajectories $\mathcal{D}$. It turns out that $\mathtt{R}_\phi$ is not continuous on $\mathcal{D}$, because $\mathcal{D}$ contains trajectories with discontinuous jumps, and the notion of metric convergence in $\mathcal{D}$ allows one to align close jumps (in time) between two trajectories. On the other hand, in the definition of the robustness score ρ, there is no such flexibility on the time bounds of the formula. This discrepancy results in the lack of continuity. We formalise and prove this in the proposition below.

Proposition 5.1 *Let ϕ be a STL formula. The functional $\hat{\mathtt{R}}_\phi : \mathcal{D}([0,T],\mathbb{D}) \to \mathcal{D}([0,T-T_\phi],\mathbb{D})$ is not continuous with respect to the Skorokhod topology.*

Proof: We provide a counterexample to continuity, by exhibiting a formula ϕ and a trajectory $\mathbf{x} \in \mathcal{D}([0,T],\mathbb{D})$ such that $\hat{\mathtt{R}}_\phi$ is not continuous in $\mathbf{x}$. Fix $\phi = \mathcal{F}_{[1,2]} X \geq 0$, where X is the only system variable. The trajectory $\mathbf{x}(t)$ we consider is equal to 1 for $t \in [0,1) \cup [2,T)$, and equal to 0 for $t \in [1,2)$. Let now $\mathbf{x}^n(t)$ be equal to 1 for $t \in [0,1) \cup [2+\varepsilon_n, T)$ and to 0 for $t \in [1, 2+\varepsilon_n)$, where $\varepsilon_n > 0$ is a sequence such that $\varepsilon_n \to 0$. We have that $\hat{\mathtt{R}}_\phi(\mathbf{x}) = \mathbf{y}$, with $\mathbf{y}$ the constant function 1, while $\hat{\mathtt{R}}_\phi(\mathbf{x}^n) = \mathbf{y}^n$, where $\mathbf{y}^n(t) = 0$ for $t \in [0, \varepsilon_n)$ and $\mathbf{y}^n(t) = 1$ for $t \geq \varepsilon_n$. It is easy to check that $\mathbf{x}^n \to \mathbf{x}$ in the Skorokhod topology, using the sequence of time wiggle functions ω^n such that $\omega^n(2) = 2 + \varepsilon_n$ and ω^n linear elsewhere. On the other hand $\mathbf{y}^n$ does not converge to $\mathbf{y}$, which proves the claim, as continuous functions send convergent sequences to convergent sequences. ∎

Continuity, however, is a desirable feature, as it guarantees that small perturbations in system trajectories will result in small perturbations in the robustness score. Hence, a more precise characterisation of the continuity properties of $\mathtt{R}_\phi$ on subspaces of $\mathcal{D}$ would be valuable, yet nontrivial. It is quite easy to assess that $\mathtt{R}_\phi$ is continuous on the subspace $\mathcal{C} \subset$

$\mathcal{D}$ of *continuous* trajectories, as in $\mathcal{C}$ the metric on $\mathcal{D}$ reduces to the standard supremum norm, for which $\mathtt{R}_\phi$ is known to be continuous (DFM13). We conjecture that $\mathtt{R}_\phi$ will be continuous also for most of the trajectories with jumps. The argument is as follows. Consider a simple formula $\phi = \mathcal{F}_{[T_1,T_2]}X \geq 0$. Then problems may arise for all those trajectories for which two discontinuous jumps happen at exactly T_2, T_1, or $T_2 - T_1$ time instants apart. We believe, although we still do not have a formal proof, that the functional $\mathtt{R}_\phi$ is continuous on all other trajectories. This soon implies that $\mathtt{R}_\phi$ is almost surely continuous with respect to any probability measure on $\mathcal{D}$ induced by a Continuous Time Markov Chain (or by other nicely behaved stochastic processes, like Feller processes (Kal10)). By standard arguments about weak convergence of probability measures on $\mathcal{D}$ (Bil99), this will guarantee that small perturbations of any stochastic model will results in small perturbations of the distribution R_ϕ of the robustness score.

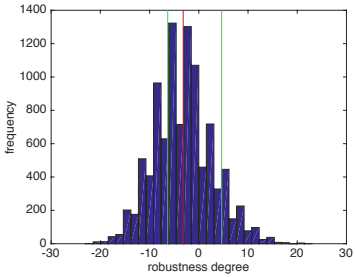
The alternative would be to modify the definition of the robustness score ρ to enforce continuity of $\mathtt{R}_\phi$. This would require a notion of space-time robustness. In (DM10), the authors consider a definition which is based on the localisation of zeros (of the atomic predicates or of the robustness function). Unfortunately, this degree can be computed easily for piecewise linear continuous signals, but it is undecidable in general, even for continuous functions (Col08; Ric97)). A possible alternative can be that of "blurring" the boundaries of the time intervals by a proper use of integrals. Investigating such a direction, however, is out of the scope of this work.

Example 5.1 (SIR stochastic property) *We give a simple example of the computation of the average robustness. For more interesting case studies, we remind to Section 5.4. Let's consider again the SIR population model presented in the Example 2.1. In this case, we want to study the stochastic dynamics of the system. The stochastic process for this epidemic model can be represented by the family of vectors* $(\mathbf{X}(t))_{t\in\mathbb{R}\geqslant 0} = (X_S(t), X_I(t), X_R(t))_{t\in\mathbb{R}\geqslant 0}$ *where each* $X_i(t) \in \mathbb{N}$ *counts the number of indivuals in the state* i *at time* t*, and* N *is the number of the whole population. Then, we consider the property*

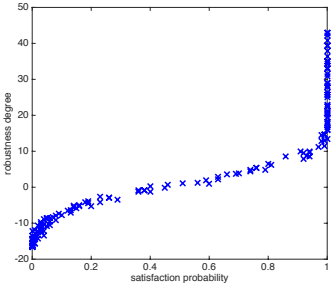
$$\varphi_{peak} : \mathcal{G}_{[35,45]}(x_I > 30). \tag{5.2}$$

The meaning of the formula is "the number of infected individuals is always more than 30 between 35 and 45 time units".

In Figure 5.1(a) left, we plot the distribution of the average robustness for the formula 5.2, for 10000 runs. The red line corresponds to the average robustness, equal to -3.1704 with error ± 0.1899 *at* 95% *confidence level. The green lines represent the conditional averages robustness of the formula* φ *in (5.4) being true or false. The satisfaction probability, indeed is* $p = 0.2582$*, error* ± 0.0138 *at* 95% *confidence level. In Figure 5.1(b), we show an example of the relationship between the average robustness and the satisfaction probability. We plot the average robustness against the satisfaction probability, varying the parameter* k_{rec} *in the interval [0.01, 0.08] with a 0.005 time step. We can see that when the probability is equal to 0 or 1 the information is the same for many parameters values; instead, we have more information with the average robustness. If we consider only the windows of parameters values where the probability* $p \in]0, 1[$ *then the dependency seems to be linear, with a Pearson correlation coefficient equal to 0.9898.*



(a) Robustness distribution



(b) Satisfaction probability vs robustness degree

Figure 5.1: Robustness distribution for Formula 5.2 for 10000 runs. The average robustness (red line) is -0.0151, the conditional averages of robustness are -6.3145 and 4.6445 (green lines) (left). Satisfaction probability versus average robustness varying parameter k_{rec} between 0.01 and 0.08 in steps of 0.03 units (right).

5.3 System Design

We now discuss an application of the robust semantics to the *system design problem*. Usually, the system design problem consists in tuning properly the uncertain parameters of a model in order to reproduce the behavioural properties observed in the experimental data. For this reason, we can refer to this problem also as the *parameter synthesis problem*. In particular, the problem we want to tackle is the following:

> Given a population model, M, depending on a set of parameters $\boldsymbol{\theta} \in \boldsymbol{K} \subseteq \mathbb{R}^d$, and a specification ϕ given by a STL formula, find the parameter combination $\boldsymbol{\theta}^*$ such that the system satisfies ϕ as *robustly* as possible.

We will tackle this problem by:

1. Rephrasing it as a (non-linear, non-convex) *optimisation problem*.
2. Evaluating the function to optimise, the average robustness.
3. Solving the optimisation problem.

Let's see in detail each point.
(1) Rephrasing it as a (non-linear, non-convex) optimisation problem. An optimisation problem is the problem of finding the best parameters that maximise or minimise a certain function. Rephrasing the problem means then to find which is the function that has to be maximise/minimise in a such a way that the model satisfies ϕ as robustly as possible. According to the quantitative semantics of STL, the robustness value $\rho(\varphi, \mathbf{x}) = \rho(\varphi, \mathbf{x}, 0)$ expresses the level of satisfaction of ϕ by a trajectory $\mathbf{x}$ (i.e., the satisfaction at time zero). We are then interested in maximising the average robustness:

$$E(R_\phi) = \int \rho(\phi, \mathbf{x}) p(\mathbf{x}) d\mathbf{x} \tag{5.3}$$

where $p(\mathbf{x})$ is the probability density of trajectory $\mathbf{x}$. So, the system design corresponds to find the parameter configuration $\boldsymbol{\theta}^*$ that maximises the function $f : \boldsymbol{K} \to \mathbb{R}$ s.t. $f(\boldsymbol{\theta}) := E(R_\phi)[\boldsymbol{\theta}]$.

(2) Evaluating the function to optimise. $E(R_\phi)$ is evaluated at a few parameter values, with a fixed number of runs, usually 100. In particular, the algorithm is initialised with a random grid of points, $\boldsymbol{\theta}_i, i = 1, .., N$ (the parameter values), for each of which $E[R_\phi]$ is approximately evaluated via statistical means. The input-output pairs $(\boldsymbol{\theta}_i, y_i), i = 1, .., N$ is the *training data* set, where $y_i \in \mathbb{R}$ are the evaluations/observations.

(3) Solving the optimisation problem. The optimisation problem is solved using an optimisation strategy for *reinforcement learning*.This methodology is an extension of (BS13; BS15) and is based on a statistical emulation of the unknown function via Gaussian processes regression (RW06) and a Gaussian Process - Upper Confidence Bound optimisation algorithm, GP-UCB (SKKS12), described in Chapter 4. An efficient estimation of the unknown objective function (i.e., the average robustness as a function of the parameters) is a key ingredient for the design problem. Function approximation is a central task in machine learning and statistics. The general regression task can be formulated as follows (Bis06): given a set of input-output pairs $(\boldsymbol{\theta}, y_i), i = 1, .., N$ (*training data*), with $\boldsymbol{\theta} \in \mathbb{R}^d$ and $y_i \in \mathbb{R}$, determine a function $f: \mathbb{R}^d \to \mathbb{R}$ s.t. $f(\boldsymbol{\theta})$ is optimally close to the target values y_i. Several methods exist for addressing this task; here we consider Gaussian Process (GP) regression, a popular Bayesian methodology (RW06) described in detail in Chapter 4. Indeed, if the statistical analysis is performed over a sufficiently large number of simulations of the process (few tens typically suffice), we can assume that the estimation noise is approximately Gaussian. For this reason, we can treat the unknown function to be maximised as a random function (sampled from a suitable prior stochastic process, namely a Gaussian Process (GP)), and the numerical estimations as function evaluations. Using the evaluations of the previous point as a training set, the GP is then used to emulate the unknown function, i.e., to make predictions regarding the $E[R_\phi]$ value at different parts of the search space. We calculate the GP posterior for a set of test points; that involves calculating an estimate of the expected robustness and its associated variance. These observations are then used to obtain a posterior prediction at new input points, which are chosen according to an efficient optimisation procedure called GP - Upper Con-

fidence Bound (GP-UCB) algorithm. The GP optimisation algorithm dictates that the point that maximises the upper quantile of the GP posterior is added to the training set, after being evaluated for its associated robustness via statistical estimation. A high value for the upper quantile at any point in the parameter space indicates the possibility of an undiscovered maximum nearby. This feature allows us to direct the search towards areas of the parameter space that appear to be more promising. More details about Gaussian process and GP-UCB can be found in Chapter 4. This process is repeated for a number of iterations, and the training set is progressively updated with new potential maxima until no further improvements can be made or we reach a prefixed maximum number of iterations. The output of the algorithm are then the parameter values corresponding to the maximum y_i among those in the training set, i.e., $\boldsymbol{\theta}^* = \boldsymbol{\theta}_i$. For a smooth objective function, the algorithm is proved to converge to the global optimum in (SKKS12).

Note that the samples also allow us to estimate the (sample) variance in the average robustness at every sampled parameter value; this information can also be included leading us to a heteroschedastic (i.e., with non identical noise) regression problem (which is however still analytically tractable), see (BS13; BS15) for more details.

To better understand how the technique works in practice, we refer the reader to the next section which describes its application to case studies step by step.

In certain cases, such as bistable systems, a large average robustness may not be the appropriate objective; in fact, for highly unbalanced robustness scores, a formula can have a high average robustness without having a high probability of being true. Therefore, one would like to modify the design problem to incorporate an additional constraint that the satisfaction probability p of the formula be bounded below by a fixed q. This considerably complicates the problem: we are not aware of provably convergent non-convex constrained optimisation algorithms. Nevertheless, the problem can be approximately solved using penalty terms to encode for probability constraints. More specifically, assuming we want to enforce the satisfaction probability to be at least q, we add a

penalty term of the form $\alpha(q - p)$, if $p < q$, and 0 otherwise, where $\alpha < 0$ controls the penalty intensity. A sufficiently high value of α (which can be chosen manually with a few trial runs) will ensure that the optimisation will satisfy the probability constraint.

5.4 Case Studies

In this section, we report a number of case studies to investigate experimentally the notion of robust semantics of STL formulae for stochastic models and its exploitation in the system design problem. We consider three case studies: the Schlögl system (GCPDI05), a simple set of biochemical reactions exhibiting a bistable behaviour, the Incoherent type 1 Feed-forward loops (I1-FFL) (Alo07), a frequent motif in gene regulatory systems, and the Repressilator (EL00; BP08; BP10), a synthetic biological clock implemented as a gene regulatory network. More specifically, we consider CTMC models of the Schlög system and the I1-FFL, and a hybrid model of the Repressilator. The models, the optimisation algorithm and the experiments are implemented in `Matlab` exploiting the `Breach` toolbox for the quantitative monitoring and a dedicated Java implementation, combining standard Monte Carlo simulation (by the Gillespie algorithm (Gil77)) and Bayesian statistical model checking (JCL$^+$09b) for the satisfaction probability. All the experiments were run on a Macbook Pro, OS X 10.9.5, Intel Core i5 processor with 2.6 GHz, 8GB 1600 MHz memory. Recently, the whole procedure has been implemented in a Java toolbox, U-Check (BMS15), briefly described in Chapter 8, together with other innovative techniques to analyse stochastic systems.

5.4.1 Schlögl System

The Schlögl model is a simple biochemical network with four reactions, listed in Table 5.1. The rates of the reactions are computed according to the mass action principle for stochastic models (Gil77). Species A and B are considered to be present in large quantities, hence they are assumed to be constant and the system will be represented by only one variable,

Reaction	rate constant	init pop
$A + 2X \to 3X$	$k_1 = 3 \cdot 10^{-7}$	$X(0) = 247$
$3X \to A + 2X$	$k_2 = 1 \cdot 10^{-4}$	$A(0) = 10^5$
$B \to X$	$k_3 = 1 \cdot 10^{-3}$	$B(0) = 2 \cdot 10^5$
$X \to B$	$k_4 = 3.5$	

Table 5.1: Biochemical reactions of the Schlögl model. Parameters are taken from (DG08a).

the concentration of the species X. The characteristic of this system is to have, for certain parameter values, like the one shown in Table 5.1, a bistable behaviour.

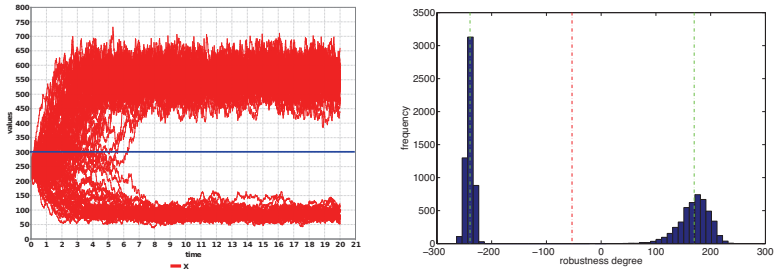
More specifically, the reaction rate ODE system has two stable steady states, and for this model the trajectories of the stochastic system starting from a fixed initial state, $X(0) = x_0$, can end up in one attractor or the other. The probability of choosing one stable state or the other depends on the position of x_0 relative to the basin of attraction of the two equilibria. If we start close to its boundary, the bistable behaviour becomes evident, see Figure 5.2(a).

We now consider the property of eventually ending up in one basin of attraction, using the following STL formula to express it

$$\varphi : \mathcal{F}_{[0,T_1]}\mathcal{G}_{[0,T_2]}(X - k_t \geq 0), \quad k_t = 300. \tag{5.4}$$

The exact meaning of the formula is: *after at most T_1 time units, the concentration of the species X stabilises to a value which remains above $k_t = 300$ for at least T_2 time units*. From the atomic predicate $\mu(X) = X - k_t \geq 0$, we can derive the secondary signal $y(x(t)) = x(t) - k_t$, where $x(t)$ is the primary signal corresponding to a trajectory of the system (the variation in the concentration of X over the time).

As Figure 5.2(a) shows, if the model is in the large equilibrium, then this property is true, and false in the other case. If we estimate the probability of the formula statistically, for model parameters as in Table 5.1 and formula parameters $T_1 = 10$ and $T_2 = 15$, then we obtain the value $p = 0.4583$ (10000 runs, error ± 0.02 at 95% confidence level and execution

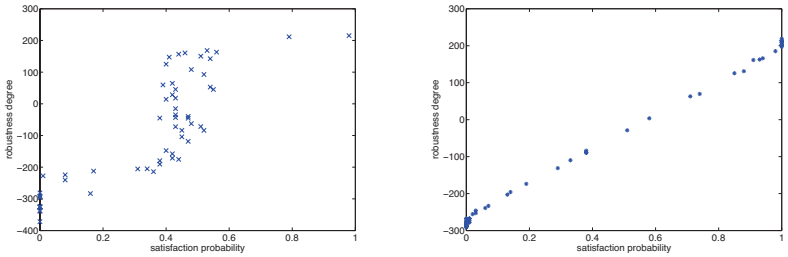


(a) CTMC simulation

(b) Distribution of the robustness degree

Figure 5.2: Simulation of the Schlögl model (100 runs), for parameters as in Table 5.1. The blue straight line is the value $X = 300$ (left). The distribution of robustness degree for the STL formula 5.4 with $T_1 = 10$ and $T_2 = 15$ time units (10000 runs). The average robustness is -53.15 (vertical red line), the conditional averages of robustness are 169.89 and -239.52 (vertical green lines), and satisfaction probability is 0.4552 (right).

time of 99.49 seconds). This raw number cannot be used to retrieve any information on the bistability of the system. Indeed, a system stabilising just above $x(t) = k_t = 300$, and such that roughly 55% of its trajectories cross such threshold "frequently", can satisfy the same formula with the same probability. The bimodal behaviour of the system becomes evident, instead, if we look at the distribution of the robustness degree of the formula, see Figure 5.2(b). The figure shows also the conditional robustness averages of the formula φ in (5.4) being true or false that are 169.89 and -239.52, respectively. These two indicators estimate how robustly the system remains in the basin of attraction of each steady state. Hence, the robustness degree carries additional amount of information w.r.t. the satisfaction probability of a STL formula. We stress that we are not comparing the robustness degree with the probability distribution of the CTMC $X(t)$: both the satisfaction probability of φ and its robustness are (unidimensional) quantities derived from $X(t)$, which are easier to compute and visualise. In Figure 5.3, we varied the threshold level k_t in the formula (Figure 5.3(a)), and the rate constant k_3 (Figure 5.3(b)), and



(a) Varying the threshold k_t.

(b) Varying the parameter k_3.

Figure 5.3: Satisfaction probability versus average robustness degree for varying (left) the threshold k_t in the STL formula (5.4) and (right) the parameter k_3. k_3 was varied between 100 and 300 in steps of 10 units, while the threshold k_t was varied between 50 and 600 in steps of 10.

then we plot the satisfaction probability versus the average robustness degree, estimating them statistically from 10000 runs for each parameter combination. As we can see these two quantities seem to be correlated. By varying the threshold, the Pearson's correlation coefficient between satisfaction probability and robustness degree is 0.8386, while the dependency, by visual examination of the data, seems to follow a sigmoid shaped curve. In the second case, instead, the correlation between satisfaction probability and average robustness degree is 0.9718, with an evident linear trend.

System Design. We set up the experiment as follows. We ask to maximise the robustness degree of the formula 5.4 optimising the parameter k_3. We varied k_3 uniformly in $[50, 1000]$, fixing all other parameters to the values of Table 5.1. We ran the GP-UCB optimisation algorithm by first estimating the robustness degree for 15 points sampled randomly and uniformly from the parameter space with 100 runs for each sample, and then using the GP-UCB strategy to estimate the maximum of the upper bound function in a grid of 200 points. If, in this grid, a point is found with a larger value than those of the observation points, we compute

the robustness also for this new point, and add it to the observations (thus changing the GP approximation). Termination happens when no improvement can be made after three grid resampling. Further integration of local maximisation can further improve the method.
In the experiment, repeated 10 times, we used a GP with radial basis kernel (Bis06), with length scale fixed to 0.5 (after standardisation of the parameter range to $[-1, 1]$). The amplitude of the kernel was adaptively set to 60% of the difference between the max and the mean value of the robustness for the initial observations. The observation noise was experimentally fixed to 1, by monitoring the average standard deviation at different random parameter combinations.
The median of the results are shown in Table 5.2. As we can see, the result of the optimisation suggests that the more robust system satisfying the specification (i.e., remaining as much as possible above the threshold 300 for a sufficiently long amount of time) is the one obtained for $k_3 = 1000$. This is in agreement with the fact that, in this case, the robustness function is easily seen to be a monotonic and increasing function of the parameter. This property is helpful to test the method, given that we know the analytic solution. Furthermore, GP regression using a gaussian kernel often results in an accurate reconstruction of a monotonic function, due to the kernel shape (RW06), resulting in a challenging problem for our method. In Table 5.2 we report also the optimisation time, 23.156 seconds, which is almost entirely spent in evaluating the likelihood (22.565 seconds), i.e., in running the simulations of the system. In Figure 5.4(a), we plot the emulation of the robustness function obtained in the last iteration of one of the 10 experiments. We can see also from here that the optimum value of k_3 (the value that maximise the emulation function and the robustness degree) is 1000. The result is confirmed by the computation of the approximated robustness distribution, Figure 5.4(b): for $k_3 = 1000$, the system becomes monostable, and X stably remains above 550 units (corresponding to an average robustness score above 250).

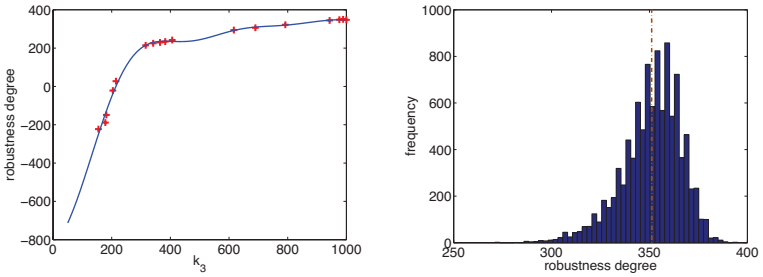
k_3 **optimisation**	Median	Range
Parameter	1000	[939.08, 1000]
Average robustness	350.6	[344.98, 353.74]
Probability satisfaction	1	1
n Rob. fcn evaluations	17	[16, 20]
n simulation runs	1700	[1600, 2000]
Rob. fcn eval. time (sec.)	22.565	[20.19,29.43]
Optim. time (sec.)	23.156	[20.58, 30.82]

Table 5.2: Statistics of the results of 10 experiments to optimise the parameter k_3 in the range $[50, 1000]$. We report the median and the range of the optimum parameter, the average robustness, the probability satisfaction, the number of robustness function evaluations (n Rob. fcn evaluations), the total number of simulation runs, and the time, in seconds, of the robustness function evaluations (Rob. fcn eval. time) and the optimisation (Optim. time). The number of runs for each evaluation, i.e., each SMC, is 100.

5.4.2 Type 1 Incoherent Feed Forward Loop

The second example we discuss is a small frequent motif in genetic regulatory networks (Alo07), known as the feed forward loop (FFL). FFL are composed by two genes, B and C, in which B regulates C and both are regulated by a third transcription factor, the product of gene A. The regulation is acyclic: there are no feedback loops. However, different roles played by A and B as regulators of the expression of B and C give rise to different behaviours, which are used within the cell to modulate the response to external (or internal) stimuli, changing the expression of A. In this study, in particular, we discuss the *incoherent type-1 FFL* (I1FFL), according to the nomenclature of (Alo07). I1FFL is characterised by a topology with two parallel but competitive paths: A activates the production of both B and C, while B is a repressor of C. We consider the case of an "AND" logic gate in C, corresponding to the situation in which we have production of C if A is above a certain value threshold and B is not above a certain value threshold (i.e., B repression is not active).

The dynamics of this network can be understood in terms of an input/output relationship, where C is the output signal. In the presence



(a) The estimated robustness function (b) Robustness distribution after the optimisation

Figure 5.4: The emulated robustness function in the optimisation of k_3 (left). The distribution of the robustness score for $k_3 = 1000$ and 10000 runs; the average robustness (red line) is 351.1 and the satisfaction probability is 1.

of an external input signal, which corresponds to an high value of A, the production of B and C is activated in parallel. B takes some time to accumulate and to cross the threshold to activate C repression. This effect translates in an initially high production of C followed by a decrease and a consequent stabilisation to a low steady state. This results in pulse-like response to the input signal, as can be seen in Figure 5.5(a), left.

The network is described by a Markov population process, in which activation and repression are modelled as Hill functions, while degradation is described in the standard mass action style (Gil77). The value of X_A is considered to be constant, as an input signal. More precisely, we have the following reactions:

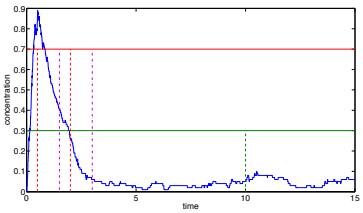
B **production:** $\varnothing \to B$, at rate $\beta_B \frac{X_A^n}{K_{AB}^n + X_A^n}$;

B **degradation:** $B \to \varnothing$, at rate $\alpha_B X_B$;

C **production:** $\varnothing \to C$, at rate $\beta_C \frac{X_A^n}{K_{AC}^n + X_A^n} \frac{1}{K_{BC}^n + X_B^n}$;

C **degradation:** $C \to \varnothing$, at rate $\alpha_C X_C$;

We normalise the system, dividing each value X_A, X_B, X_C, for the total population $X_A + X_B + X_C$. For simplicity, we will denote by X_A, X_B, X_C



(a) Stochastic simulation (b) Robustness distribution

Figure 5.5: Simulation of the I1FFL model. Trajectory of X_C, for parameters $K_{AB} = 1$, $K_{AC} = 1$, $K_{BC} = 0.17$, $\alpha_C = 1$, $\beta_C = 1$, $\alpha_B = 1$, $\beta_B = 1$ and n = 2, $X_A = 1$ and initial value $x_B(0) = 0$, $x_C(0) = 0$. The range of variables is expressed as a concentration (left). The distribution of the robustness degree for the formula 5.5 with $\theta_{high} = 0.7$, $T_r = 0.5$, $h = 1$, $T_s = 1.5$, $T_{off} = 10$, $T = 15$, and $\theta_{low} = 0.3$ for 10000 runs (right). The simulator for the model has been implemented in Java.

the normalised variables.
In order to capture the pulse-like behaviour, we use the STL formula

$$\phi_{pulse} = \mathcal{F}_{[T_r,T_r+h]}\mathcal{G}_{[0,T_s]}(X_C \geq \theta_{high}) \wedge \mathcal{G}_{[T_{off},T]}(X_C \leq \theta_{low}). \qquad (5.5)$$

The formula ϕ_{pulse} requires the output signal to be above an high threshold θ_{high} at a certain time $t \in [T_r, T_r+h]$ from the introduction of the input signal, and remains high for at least T_s time units. Furthermore, the formula imposes that the pulse has terminated after T_{off} units of time, so that the concentration of X_C stabilises to a low value (less than θ_{low}) for at least T time units.

In Figure 5.5(b), we show the distribution of the robustness degree of the formula (see caption for parameters) with 10000 simulations. The average robustness degree is -0.2198 and is almost equal to the negative average robustness, -0.2204. Indeed, the formula is "almost always" false, as confirmed by the satisfaction probability degree that is 0.0014. Figure 5.5(a) illustrates on the left that the negative robustness on a simulated trajectory is caused by the pulse peak, which does not last enough time. We show now how to tackle the system design problem of this stochastic model by using the robustness degree to guide the parameter

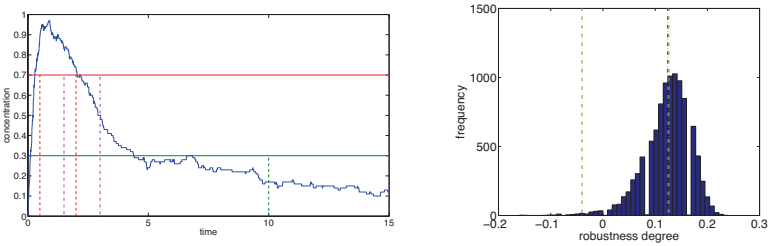
synthesis for which the pulse will become larger and longer.

System Design. In the I1FFL example, we try to optimise model parameters to enforce a specific shape to the pulse, namely a duration of 1.5 time units with an amplitude larger than 0.7. This is obtained in the STL framework by assigning the following parameters to the formula (5.5): θ_{high} = 0.7, T_r = 0.5, h = 1, T_s = 1.5,T_{off} = 10, T = 15 and θ_{low} = 0.3.

We considered two scenarios, assuming we can regulate the repression and degradation rates of the regulation of protein C. In the first case, we optimise only the degradation α_C, while in the second case, we optimise simultaneously both K_{BC} and α_C. For each scenario, we use the robustness score of the STL formula 5.5.

The setting of the algorithm are similar to the ones for the Schlögl model, except for the hyper-parameters of the kernel and the observations noise. In this case, the hyper-parameters of the kernel have been identified relying on a model selection criterion, i.e., optimising the robustness function as computed from an initial batch of observations, see (RW06) for more details. We improve also the treatment of the observation noise by using an heteroscedastic noise model. The noise of the robustness function for each explored point of the parameter space is estimated by bootstrapping. Furthermore, in this set of experiments, we use 10 initial random samples for each parameter.

We ran the optimisation algorithm 10 times. The results are reported in Figure 6.5.1 and Table 5.3. As we can see, the algorithm returns a precise value in one dimension, while it tends to be more erratic when searching the two dimensional space. This is typically a sign of uncertainty in the identification of parameters, meaning that there is some sort of dependency between the parameters we are exploring, resulting in a flat maximum or in a ridge of points more or less with the same robustness. This is confirmed in Figure 5.7, where we plot the emulated function at the end of one optimisation. We can see that in almost all the region identified by the parameter range of Table 5.3 for α_C and k_{BC}, the robustness degree is similar (the dark red region); furthermore, the variability is more evident for the α parameter than the threshold concentra-



(a) Stochastic simulation

(b) Robustness distribution

Figure 5.6: Simulation of the I1FFL model, for parameters $kBC = 0.1732$, $\alpha_C = 0.3555$, $K_{AB} = 1$, $K_{AC} = 1$, $\beta_C = 1$, $\alpha_B = 1$, $\beta_B = 1$ and n = 2, and initial value $x_A(0) = 1$, $x_B(0) = 0$, $x_C(0) = 0$. The range of variables is expressed as a concentration (left). The robustness distribution of the formula 5.5 with $\theta_{high} = 0.7$, $T_r = 0.5$, $h = 1$, $T_s = 1.5$, $T_{off} = 10$, $T = 15$, and $\theta_{low} = 0.3$ for 10000 simulations (right).

tion K_{BC}. Finally, we can note from Table 5.3 that also in this case most of the computational cost is spent in the evaluation of the robustness function, i.e., in simulation of the model and statistical model checking.

α_C, K_{BC} **optimisation**	α_C		α_C, K_{BC}	
	Median	Range	Median	Range
Parameter	0.3759	[0.3703, 0.4038]	0.3812, 0.1742	[0.3183, 0.4806], [0.1556, 0.1899]
Average robustness	0.1239	[0.1187, 0.1358]	0.1270	[0.1186, 0.1352]
Probability satisfaction	0.942	[0.899, 0.999]	0.973	[0.897, 0.989]
n rob. fcn evaluations	22	[17, 23]	58	[55, 62]
n simulation runs	2200	[1700, 2300]	580	[5500, 5800]
Rob. fcn eval. time (sec.)	28.38	[21.22, 30.83]	81.69	[77.59, 89.77]
Optim. time (sec.)	30.42	[22.17, 33.36]	108.98	[99.73, 129.91]

Table 5.3: Statistics of the results of 10 experiments to optimise the parameter α_C in the range $[0.035, 3.5]$ and simultaneously both α_C in the same range and the parameter K_{BC} in the range [0.017,1.7]. We report the median and the range of the optimal parameter, the average robustness, the probability satisfaction, the number of robustness function evaluations (n rob. fcn evaluations), the total number of simulation runs, and the time, in seconds, of the robustness function evaluations (Rob. fcn eval. time) and the optimisation (Optim. time). The number of runs for each function evaluation, i.e., the number of simulations for SMC, is 100.

5.4.3 Repressilator

The last case study is a genuine stochastic hybrid model of the Repressilator (EL00), a synthetic genetic clock composed of three genes expressing three transcription factors repressing each other in a cyclical fashion (see Figure 5.8). Genetic networks like the Repressilator can also be found in actual biological systems. Here we consider the putative regulatory network of the Circadian Clock in *Ostreococcus Tauri*, an unicellular alga that is widely studied as a model organism. The model and parameters are taken from (OMS13), where the authors start from experimental data and learn a stochastic hybrid model of the circadian clock network in *O. Tauri*, which is known to involve only two genes, expressing transcription factors TOC1 and CCA1. They conjecture the existence of a third regulatory protein X, similarly to the mechanism recently discovered for the circadian clock in *Arabidopsis Taliana*, forming a repression cycle as in Figure 5.8.

This system is modelled by a SHA with three continuous variables,

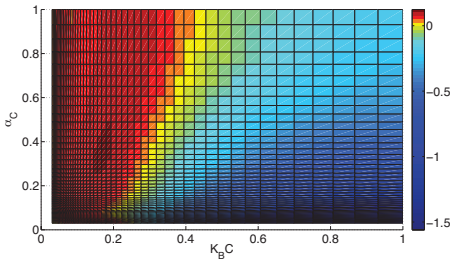
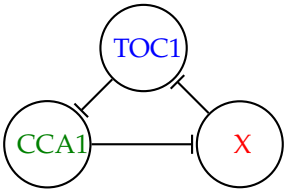
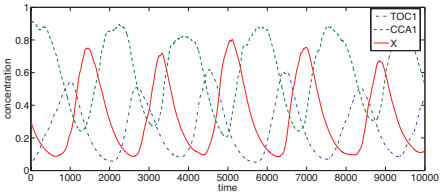


Figure 5.7: Part of the emulated robustness function in the optimisation of α_C and K_{BC} for the I1FFL example. The colour corresponds to the value of the average robustness in agreement with the legend on the right of the plot.



(a) Repressilator-like gene network of the O. Tauri circadian clock (OMS13)



(b) Hybrid stochastic simulation

Figure 5.8: The repressilator-like model of the O.Tauri circadian clock (left) is a cyclic negative-feedback loop composed of three repressor genes: TOC_1, CCA_1, and an unknown gene X. Oscillatory behaviour of the model (right), for model parameters taken from (OMS13).

X_{TOC1}, X_{CCA1}, and X_X, and with eight discrete modes, corresponding to all possible combinations of active and inactive states of each involved gene. The dynamics of gene repression is modelled as a telegraph process, i.e., as a two states Markov model, with a phenomenological rate of repression of gene i (binding of the protein to the gene) equal to

$$f_{bind,i}(\mathbf{X}) = k_{p_i} \exp(k_{e_i} X_j),$$

where X_j is the repressor of gene i. The unbinding rate, instead, is constant: $f_{unbind,i}(\mathbf{X}) = k_{m_i}$. The steady state distribution of the probability of gene i being active is a sigmoid (saturating) function of the repressor concentration. Production and degradation of protein i, instead, are modelled by stochastic differential equations of the form

$$dX_i = (A_i\mu_i + b_i - \lambda_i X_i)dt + \sigma dW,$$

where μ_i denotes the state of gene i (with $\mu_i = 1$ denoting the repressed state and $\mu_i = 0$ the active state), b_i is the basal production rate and $A_i < 0$ reduces it in case of repression, and λ_i is the degradation rate. The form of this model is particularly efficient to perform statistical inference of parameters in presence of observed data. Note also that the stochastic hybrid model we consider here is different from previous hybrid models of the repressilator (BP08; BP10), in that it assumes a different form for the rate of gene repression, and in that it models protein production and degradation by stochastic differential equations rather than ODEs. In Figure 5.8(b), we show a simulation of the model, for the parameter values taken from (OMS13), which exhibit sustained oscillations with a more or less stable period.

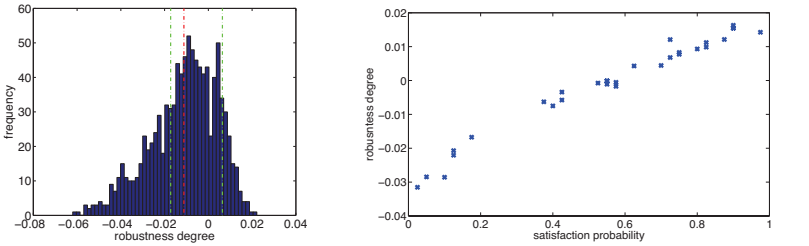
In order to specify the presence of oscillations, we use the STL formula

$$\begin{aligned}\psi = \mathcal{G}_{[0,T]}((X_i > k_{high}) \rightarrow \mathcal{F}_{[T_1,T_1+h]}(X_i < k_{low}))\wedge \\ \mathcal{G}_{[0,T]}((X_i < k_{low}) \rightarrow \mathcal{F}_{[T_2,T_2+h]}(X_i > k_{high})) \wedge \mathcal{F}_{[0,T]}(X_i > k_{high}),\end{aligned} \tag{5.6}$$

expressing the fact that high values of X_i alternate to low values, with a period between $T_1 + T_2$ and $T_1 + T_2 + 2h$, where i is TOC1, CCA1 or X. In particular, we require that a high value of X_i ($X_i > k_{high}$) is followed within time $[T_1, T_1 + h]$ by a low value of X_i, which is subsequently followed by a high value in a time between $[T_2, T_2 + h]$. The last part of the formula requires that the model indeed reaches a high value of X_i in order to trigger the chain of implications. The parameter h gives a value of the period stability, the higher the h, the more irregular is the period. As said before, X_i can be the concentration of one of the three proteins of the clock. In the next discussion, we focus on the unknown protein X.

Again, the robustness degree provides a measure of the satisfaction/violation of the formula. An example is shown in Figure 5.9(a) left, where

we can see that the formula, for the parameters in the caption, tends to be false (the average robustness is negative and the satisfaction probability is 0.256). This, of course, may depend on the choice of the parameters, which do not properly capture the amplitude, the period, and the shape of the oscillations. For instance, a negative robustness value of δ can be obtained if, from a point in which $X_i < k_{low}$, the system remains below $k_{high} - \delta$ for a whole (half) period of oscillation (which is constrained to be in $[T_2, T_2 + h]$). We will see now how these parameters can be chosen in a principled way, taking inspiration from requirement mining (JDDS13; GSC+09). In the same figure, we can also see the conditional average robustness, whose values are both close to the average robustness, suggesting that the low satisfaction probability is not very robust. In Figure 5.9(b), we plot the average robustness against the satisfaction probability, varying the property parameter T_1, showing once again the correlation between the two quantities. The dependency seems to be linear, with a Pearson correlation coefficient of 0.9841.



(a) Robustness distribution (b) Satisfaction probability vs robustness degree

Figure 5.9: Robustness distribution for Formula 5.6 parameters $k_{low} = 0.14$, $k_{high} = 0.5$, $T_1 = 800$, $T_2 = 700$, $h = 350$, $T = 7000$ and 1000 runs. The average robustness (red line) is -0.0151, conditional averages of robustness are -0,0183 and 0,0064 (green lines), and the estimated satisfaction probability is 0.256 (left). Satisfaction probability versus average robustness. T_1 was varied between 800 and 1100 in steps of 10 units (right).

Property Design. In this final scenario, we consider a different optimisation problem. We keep model parameters constant and we try to

optimise the parameters of the formula to make the robustness score as large as possible. This is a version of the requirement mining problem (JDDS13), which can be seen as a sort of dual problem to system design, in which the goal is to learn the emergent behaviour of the model in terms of the most robustly satisfied formula (of fixed structure). Furthermore, the parametrisation of a formula is usually an underestimated problem, as the satisfaction/robustness heavily depends on these parameters. This problem has been partially tackled, e.g., in (RBFS08; JDDS13) for deterministic models, but never for stochastic ones, to authors' knowledge.

In particular, we consider Formula (5.6), and optimise the temporal delays T_1 in the range $[100, 1700]$ and T_2 in the range $[100, 1000]$. This can be seen as an attempt to learn the best bounds on the oscillatory period, through the filter of the logical specification of oscillations of Formula (5.6). Inspecting the structure of the formula, we can observe that it is the conjunction of two temporal properties, one containing the parameter T_1 ($\phi_1 = \mathcal{G}_{[0,T]}((X_i > k_{high}) \rightarrow \mathcal{F}_{[T_1,T_1+h]}(X_i < k_{low}))$) and the other one containing the parameter T_2 ($\phi_2 = \mathcal{G}_{[0,T]}((X_i < k_{low}) \rightarrow \mathcal{F}_{[T_2,T_2+h]}(X_i > k_{high}))$). We can exploit this structure and perform the optimisation of ϕ_1 and ϕ_2 separately, as model parameters are fixed and the total robustness degree is simply the minimum of those of ϕ_1 and ϕ_2. In these formulae, we also keep T and h constant. T is related to the length of the signal we are observing, while h governs the length of the error we allow on the half period. In particular, observe that maximising h is meaningless: it is easy to show that the (average) robustness score will increase monotonically with h, so that the optimisation will always pick the upper bound. Hence, we fix $h = 350$. We decide to study the oscillation of the unknown gene X, i.e we set $i = X$.

The results of the optimisation of the two parameters are reported in Table 5.4, while in Figure 5.10(a) we show the emulated robustness function of T_1. The optimisation algorithm was set as for the I1FFL case, save for the number of initial observations, set to 12. The parameters of the model are fixed to those shown in the caption of Figure 5.8. As we can see from the robustness distribution in Figure 5.10 right, we find param-

T_1, T_2 **optimisation**	T_1		T_2	
	Median	Range	Median	Range
Parameter	903.65	[895.89, 947.56]	675.28.	[658.73, 688.11]
Average robustness	13.72	[11.45, 15.15]	0.0113	[0.0112, 0.0115]
Probability satisfaction	0.882	[0.874, 0.883]	0.689	[0.685, 0.695]
n rob. fcn evaluations	23	[22, 23]	20	[17, 22]
n simulation runs	2300	[2200, 2300]	2000	[1700, 22]
Rob. fcn eval. time (sec.)	2534	[2381, 2543]	2189	[1848, 2727]
Optim. time (sec.)	2537	[2385, 2546]	2189	[1849, 2729]

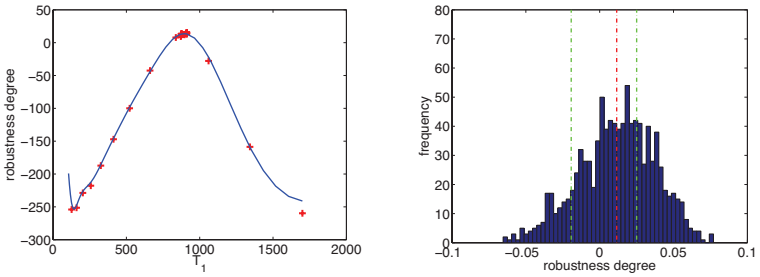
Table 5.4: Statistics of the results of 10 experiments to optimise the parameter T_1 in the range $[100, 1700]$ and of 10 experiments to optimise T_2 in the range $[100, 1000]$. We report the median and the range of the optimal parameter, the average robustness, the probability satisfaction, the number of robustness function evaluations (n rob. fcn evaluations), the total number of simulation runs, and the time, in seconds, of the robustness function evaluations (Rob. fcn eval. time) and the optimisation (Optim. time). The number of runs for each evaluation, i.e., each SMC, is 100.

eters increasing the average robustness score of the formula to a positive value. From the emulated robustness function of parameter T_1 (Figure 5.10 left), we can note that T_1 is identified precisely: the robustness function has a strict maximum. Finally, we observe that if we had taken the search domains of T_1 or T_2 to be much larger, we would have found additional maxima of similar height, corresponding to values of T_1 increased by one period, two periods, and so on.

5.5 Related Works

In the last years, there was a great effort to develop new and efficient techniques to guide the system design of models with uncertain parameters, using temporal logic.

For example, Batt et al. in (BYWB07) showed that the behaviour of a genetic regulatory network can be approximated with a piecewise multi-affine system. This class of models exhibits useful convexity properties which allows to compute a conservative finite-state automaton abstrac-



(a) The estimated robustness function

(b) Robustness distribution after the optimisation

Figure 5.10: The emulated robustness function in the optimisation of T_1 (left). The distribution of the robustness score for $T_1 = 903.65$ and $T_2 = 675.28$. The average robustness is 0.0113 and the estimated satisfaction probability is 0.689. (right)

tion where the states represent the reachable sets in the form of hyper-rectangles in which the original state-space is partitioned and the transitions among the states characterise an over-approximation of the flows among the reachable sets. In a model with uncertain parameters, several different finite-state automata can be derived starting from different parameter sets. The model checking of a temporal logic formula guides the selection of the parameters sets for which the conservative abstraction, and so the model, will violate the problem of interest. Recently, other authors have extended this approach (GBF+11; BLMP12), by introducing an optimal approximation algorithm, to biological models with generic nonlinear differential equations such as the cardiac cell excitability (GBF+11) and the bone remodelling (BLMP12) case studies. However, this approach cannot be applied to stochastic models and the use of an over-approximation abstracts away important timing relations, resulting in the selection of very coarse parameter sets.

An alternative approach is to use under-approximation techniques such as simulation or sampling. Following this direction, in the last years, there was a great scientific effort to enrich the classical qualita-

tive semantics of temporal logic or *satisfiability* (yes/no answer for the formula satisfaction of a trajectory) with more powerful and useful notions of quantitative semantics (FP07; FP09; DM10; RBFS08; ALFS11; AGBB14; HJK+15) (or *robustness degree*), providing a real value measuring the level of satisfaction or violation for a trajectory of the property of interest. Several tools, such as BIOCHAM (CFS06), S-TaLiRo (ALFS11) and Breach (Don10), are now available to perform robustness analysis on the time series collected in wet-lab experiments or produced by simulation-based techniques. The robustness degree have been successfully employed in the analysis of ODE-based biological models, to tune the parameters discriminating the behaviours observed experimentally. In (DFG+11), Donzé et al. proposed a multi-step analysis, where they adopt STL to express dynamical properties and they use robustness and sensitivity analysis to sample efficiently the parameter space, searching for feasible regions in which the model exhibit a particular behaviour. In (BBN13), we proposed a new approach, based on robustness degree, for the design of a synthetic biological circuit whose behaviour is specified in terms of signal temporal logic (STL) formulae. However, in all the aforementioned cases, stochasticity is not taken into account. It is worth mentioning that all these simulation based techniques are based on the (approximate) solution of reachability problems for non-linear ODE systems, which can be tackled with Bayesian optimisation techniques presented in this paper (BS14).

With regard to the stochastic models, the satisfiability analysis has been considered as a discriminating criterion to tune the parameters in the design process using both simulation-based statistical approximated methods (KNP04; BS15) and symbolic methods (LMST07; BGK+11). Lanotte et. al (LMST07) showed that for parametric probabilistic transition systems (or discrete time Markov chains) the problem of finding (symbolically) an instance of parameter values for a reachability property to be satisfied is equivalent to the problem of finding the roots of a general polynomial and so it is generally undecidable if proper restrictions are not considered. In (DG08b), the authors proposed a combined approach of a model checker together with a genetic algorithm to guide

the parameter-estimation process by reducing the distance between the desired behaviour and the actual behaviour. The work (HKM08) concerns with the parameter-synthesis problem, using symbolic methods, for parametric continuos time markov chains and time-bounded properties. Also in this case the problem results to be generally undecidable and the authors proposed (HKM08) an approximation method that provide a solution in most cases. Another related work in this sense is that of (BCDŠ13), where authors compute exactly upper and lower bounds on the satisfaction probability within a given region of the parameter space. All these approaches are designed to work with specific classes of stochastic models.

Chapter 6

Signal Spatio-Temporal Logic

In this chapter, we present the *Signal Spatio-Temporal Logic*, SSTL, (NB14; NBC+15), a spatial extension of *Signal Temporal Logic*, STL, described in Chapter 3. In particular, we extend STL with two spatial modalities: the *bounded somewhere* operator $\diamondsuit_{[d_1,d_2]}$ and the *bounded surround* operator $\mathcal{S}_{[d_1,d_2]}$. In the following, we first introduce the type of signals that the logic specifies, then we define the syntax and the semantics of SSTL; afterwards, we introduce the monitoring algorithms for this language. We defined also a stochastic semantics for SSTL, similarly to the one define for STL in the previous chapter. Finally, we present a number of case studies to illustrate the logic at work.

6.1 Spatio-Temporal Signals and Traces

SSTL is interpreted on spatio-temporal signals with continuous time and discrete space.

Formally, a spatio-temporal signal, is a function $\mathbf{s} : \mathbb{T} \times L \to \mathbb{E}$, where

- $\mathbb{T}$ is the time domain; it is a real-valued interval $[0, T] \subseteq \mathbb{R}_{\geq 0}$, for some $T > 0$,

- L is a discrete set of locations. In particular, the discrete space is described by a weighted graph $G = (L, E, w)$, where L is the set of locations (nodes), E is the set of edges, and $w : E \to \mathbb{R}$ is a function that associates a weight to each edge.

  The space is also equipped with a metric, i.e., a function $d : L \times L \to \mathbb{R}_{\geq 0}$ that returns a positive real value for each pair of elements in L; in particular, the metric is the *shortest weighted path distance*, i.e., the cost of the shortest path between two different locations. More details about the space representation can be found in Chapter 2.

- $\mathbb{E}$ is a subset of $\mathbb{R}^* = \mathbb{R} \bigcup \{+\infty, -\infty\}$. Signals with $\mathbb{E} = \mathbb{B} = \{0, 1\}$ are called *Boolean signals*, whereas those where $\mathbb{E} = \mathbb{R}^*$ are called real-valued or *quantitative signals*.

A *spatio-temporal trace* is a function $\mathbf{x} : \mathbb{T} \times L \to \mathbb{D}$, s.t. $\mathbf{x}(t, \ell) := (x_1(t, \ell), \cdots, x_n(t, \ell)) \in \mathbb{D} \subseteq \mathbb{R}^n$, where each $x_i : \mathbb{T} \times L \to D_1 \subseteq \mathbb{R}$, for $i = 1, ..., n$, is the projection on the i^{th} coordinate/variable. Note that these projections have the form of quantitative signals. They are called the *primary signals* of the trace. We can thus see the trace as a set of primary signals. This means that SSTL can specify property of spatio-temporal traces. Spatio-temporal traces can be obtained, e.g., by simulating a stochastic model or by computing the solution of a deterministic system. For example, the framework of patch-based population models, described in Chapter 2, are a natural setting from which both stochastic and deterministic spatio-temporal traces of the considered type can emerge. An alternative source of traces are measurements of real systems.

6.2 Syntax

The syntax of SSTL is given by

$$\varphi := \mu \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \, \mathcal{U}_{[t_1,t_2]} \, \varphi_2 \mid \; \diamondsuit_{[d_1,d_2]}\varphi \mid \varphi_1 \, \mathcal{S}_{[d_1,d_2]}\varphi_2.$$

where μ is an atomic predicate, negation and conjunction are the standard Boolean connectives, and $\mathcal{U}_{[t_1,t_2]}$ is the *until* operator, where $[t_1, t_2]$

is a real positive closed intervals with $t_1 < t_2$. The new spatial operators are the *somewhere* operator, $\diamondsuit_{[d_1,d_2]}$, and the *bounded surround* operator $\mathcal{S}_{[d_1,d_2]}$, where $[d_1, d_2]$ is a closed real interval with $d_1 < d_2$.

Similarly to STL, the atomic predicate μ_j is of the form $\mu_j(x_1, \ldots, x_n) \equiv (f_j(x_1, \ldots, x_n) \geq 0)$, for $f_j : \mathbb{R}^n \to \mathbb{R}$. Each atomic proposition gives rise to a spatio-temporal signal. In the Boolean case, given a trace $\mathbf{x}$, we can compute the Boolean signal $s_j(t, \ell) = \mu_j(\mathbf{x}(t, \ell))$ by point-wise lifting, where $s_j : \mathbb{T} \times L \to \mathbb{B}$; Similarly, a quantitative signal is obtained as the real-valued function $y_j : \mathbb{T} \times L \to \mathbb{R}$, with $y_j(t, \ell) = f_j(\mathbf{x}(t, \ell))$, this latter is also called the *secondary signal*.

The (bounded) until operator $\varphi_1 \, \mathcal{U}_{[t_1,t_2]} \, \varphi_2$ requires φ_1 to hold from now until, in a time between t_1 and t_2 time units in the future, φ_2 becomes true. The *eventually* operator $\mathcal{F}_{[a,b]}$ and the *always* operator $\mathcal{G}_{[a,b]}$ can be defined as usual: $\mathcal{F}_{[a,b]}\varphi := \top \mathcal{U}_{[a,b]}\varphi$, $\mathcal{G}_{[a,b]}\varphi := \neg F_{[a,b]} \neg\varphi$. In a similar way, we can derive the *everywhere* operator $\boxdot_{[d_1,d_2]}\varphi := \neg \diamondsuit_{[d_1,d_2]} \neg\varphi$.

We now describe in detail the new spatial operators. The somewhere and the everywhere operators were inspired from the modal operators of the *Multiprocess Network Logic* (RS85), described in Chapter 3. The idea came from the necessity to describe behaviours at a certain distance from a specific point, e.g., "from a bike sharing station, in a radius of 100 meters, there are more than 30 bikes" or "in all the positions around my location, at a distance less the 1 km, there are no infected individuals". Formally, the spatial somewhere operator $\diamondsuit_{[d_1,d_2]}\varphi$ requires φ to hold in a location reachable from the current one with a cost greater than or equal to d_1 and lesser than or equal to d_2. The cost is given by the shortest weighted distance between the locations, i.e., the sum of the weights of the edges of the shortest path (for a formal definition of the metrics see Chapter 2). We use the word "cost" to distinguish it from the classical spatial notion of distance. Indeed, we can have two streets with the same distance but different travel time due to traffic light or congestion. In Figure 6.1, we report some examples of spatial properties. In the graph of the figure, the orange point satisfies the property $\diamondsuit_{[3,5]}\, pink$. Indeed, there exists a point at a distance 3 from the orange point that satisfies the pink property. The *everywhere* operator $\boxdot_{[d_1,d_2]}\varphi := \neg\diamondsuit_{[d_1,d_2]}\neg\varphi$ requiring

φ to hold in all the locations reachable from the current one with a total cost between d_1 and d_2. In Figure 6.1, the orange point of the graph satisfies the property $\boxdot_{[2,3]}\ \mathit{yellow}$. Indeed, all the points at a distance between 2 and 3 from the orange point satisfy the yellow property.

The surround operator $\varphi_1\ \mathcal{S}_{[d_1,d_2]}\varphi_2$ was inspired by the spatial until modality of SLCS, *Spatial Logic for Closure Spaces*, (CLLM14), described in Chapter 3. It expresses the topological notion of being surrounded by a φ_2-region, while being in a φ_1-region, with additional metric constraints. The idea is that one cannot escape from a φ_1-region without passing from a node that satisfies φ_2 and, in any case, one has to reach a φ_2-node at a distance between d_1 and d_2. For example, this operator permits to describe properties as "there are no bikes in my station but all the bike stations directly connected with me have at least one bike and are at a distance less than 100 meters from me". Formally, the surround formula $\varphi_1\ \mathcal{S}_{[d_1,d_2]}\varphi_2$ is true in a location ℓ, when ℓ belongs to a set of locations A satisfying φ_1 and at a distance less than d_2 from ℓ, the external boundary $B^+(A)$ of A must contain only locations satisfying φ_2. Furthermore, locations in $B^+(A)$ must be reached from ℓ with a cost between d_1 and d_2. $B^+(A)$ is the set of all the locations that do not belong to A but that are directly connected with a location in A. In Figure 6.1, the green points satisfy $\mathit{green}\ \mathcal{S}_{[0,100]}\ \mathit{blue}$. Indeed, for each green point we can find a region that contains the point, such that all its points are green and all the points connected with an element that belongs to the region are blue and satisfy the metric constraint. Instead, the property $\mathit{green}\ \mathcal{S}_{[2,3]}\ \mathit{blue}$ is satisfied only by the dark green point. The reason is that such a dark green point is the only point for which there exists a region (the green region) such that all its elements are at a distance less than 3 from it and are green; and all the elements of the external boundary (the blue region) are at a distance between 2 and 3 from it and are green.

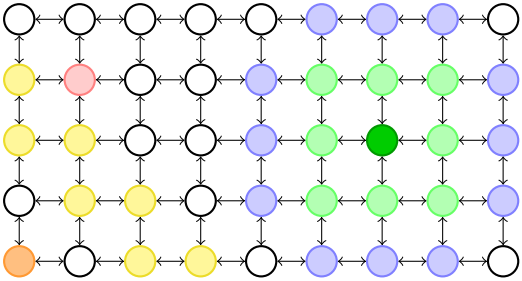


Figure 6.1: Example of spatial properties. The orange point satisfies $\diamondsuit_{[3,5]}\, pink$. The orange point satisfies $\boxdot_{[2,3]}\, yellow$. All green points satisfy $green\, \mathcal{S}_{[0,100]}\, blue$. The dark green point satisfies also $green\, \mathcal{S}_{[2,3]}\, blue$.

For more interesting examples of SSTL formulae we refer the reader to Section 6.5.

6.3 Semantics

We now define the Boolean and the quantitative semantics for SSTL. The Boolean semantics, as customary, returns true/false depending on whether the observed trace satisfies the SSTL specification.

6.3.1 Boolean Semantics

Definition 2 (SSTL Boolean semantics) *The Boolean satisfaction relation for an SSTL formula φ over a spatio-temporal trace $\mathbf{x}$ is given by:*

$$
\begin{aligned}
(\mathbf{x},t,\ell) &\vDash \mu && \Leftrightarrow && \mu(\mathbf{x}(t,\ell)) = 1\\
(\mathbf{x},t,\ell) &\vDash \neg\varphi && \Leftrightarrow && (\mathbf{x},t,\ell) \nvDash \varphi\\
(\mathbf{x},t,\ell) &\vDash \varphi_1 \wedge \varphi_2 && \Leftrightarrow && (\mathbf{x},t,\ell) \vDash \varphi_1 \wedge (\mathbf{x},t,\ell) \vDash \varphi_2\\
(\mathbf{x},t,\ell) &\vDash \varphi_1\, \mathcal{U}_{[t_1,t_2]} \varphi_2 && \Leftrightarrow && \exists t' \in t \oplus [t_1,t_2] : (\mathbf{x},t',\ell) \vDash\ \varphi_2 \wedge \forall t'' \in [t,t'],\\
& && && (\mathbf{x},t'',\ell) \vDash \varphi_1\\
(\mathbf{x},t,\ell) &\vDash \diamondsuit_{[d_1,d_2]} \varphi && \Leftrightarrow && \exists \ell' \in L : d_1 \leqslant d(\ell',\ell) \leqslant d_2 \wedge (\mathbf{x},t,\ell') \vDash \varphi\\
(\mathbf{x},t,\ell) &\vDash \varphi_1\, \mathcal{S}_{[d_1,d_2]} \varphi_2 && \Leftrightarrow && \exists A \subseteq L^{\ell}_{[0,d_2]} : \ell \in A \wedge \forall \ell' \in A, (\mathbf{x},t,\ell') \vDash \varphi_1 \wedge\\
& && && B^+(A) \subseteq L^{\ell}_{[d_1,d_2]} \wedge \forall \ell'' \in B^+(A), (x,t,\ell'') \vDash \varphi_2.
\end{aligned}
$$

A trace $\mathbf{x}$ satisfies φ in location ℓ, $(\mathbf{x},\ell) \vDash \varphi$, if and only if $(\mathbf{x},0,\ell) \vDash \varphi$.

In the semantics, $d : L \times L$ is the weighted distance function of the graph $G = (L, E, w)$; and $L^{\ell}_{[d_1,d_2]}$ is define as the set of locations ℓ' such that $d_1 \leq d(\ell, \ell') \leq d_2$, i.e., $L^{\ell}_{[d_1,d_2]} = \{\ell' \in \mathbb{SP} \mid d_1 \leq d(\ell, \ell') \leq d_2\}$. More detail about weighted distance and graph can be found in Chapter 2.

The verification of all the trace is done at time $t = 0$ because we are working with *future* temporal modalities that talk about true *now* as a function of some true in the future. Furthermore, the trace is verified in each point in space, as we assume no privilege direction or location.

6.3.2 Quantitative Semantics

The quantitative semantics returns a real value that can be interpreted as a measure of the strength with which the specification is satisfied or falsified by an observed trajectory. More specifically, the sign of such a satisfaction score is related to the truth of the formula (positive stands for true), while the absolute value of the score is a measure of the robustness of the satisfaction or dissatisfaction. This definition of quantitative measure is illustrated in Chapter 3, in the description of STL.

Definition 3 (SSTL Quantitative semantics) *The quantitative satisfaction function $\rho(\varphi, \mathbf{x}, t, \ell)$ for an SSTL formula φ over a spatio-temporal trace $\mathbf{x}$ is given by:*

$$
\begin{aligned}
\rho(\mu, \mathbf{x}, t, \ell) &= f(\mathbf{x}(t, \ell)) \quad \textit{where } \mu \equiv (f \geq 0) \\
\rho(\neg\varphi, \mathbf{x}, t, \ell) &= -\rho(\varphi, \mathbf{x}, t, \ell) \\
\rho(\varphi_1 \wedge \varphi_2, \mathbf{x}, t, \ell) &= \min(\rho(\varphi_1, \mathbf{x}, t, \ell), \rho(\varphi_2, \mathbf{x}, t, \ell)) \\
\rho(\varphi_1 \,\mathcal{U}_{[t_1,t_2]} \varphi_2, \mathbf{x}, t, \ell) &= \sup_{t' \in t + [t_1,t_2]} (\min\{\rho(\varphi_2, \mathbf{x}, t', \ell), \inf_{t'' \in [t,t']} (\rho(\varphi_1, \mathbf{x}, t'', \ell))\} \\
\rho(\diamondsuit_{[d_1,d_2]} \varphi, \mathbf{x}, t, \ell) &= \max\{\rho(\varphi, \mathbf{x}, t, \ell') \mid \ell' \in L, d_1 \leqslant d(\ell', \ell) \leqslant d_2\} \\
\rho(\varphi_1 \,\mathcal{S}_{[d_1,d_2]} \varphi_2, \mathbf{x}, t, \ell) &= \max_{A \subseteq L^{\ell}_{[0,d_2]}, \ell \in A, B^+(A) \subseteq L^{\ell}_{[d_1,d_2]}} (\min(\min_{\ell' \in A} \rho(\phi_1, \mathbf{x}, t, \ell'), \\
&\qquad \min_{\ell'' \in B^+(A)} \rho(\phi_2, \mathbf{x}, t, \ell'')))
\end{aligned}
$$

where ρ is the quantitative satisfaction function, returning a real number $\rho(\varphi, \mathbf{x}, t, \ell)$ quantifying the degree of satisfaction of the property φ by the trace $\mathbf{x}$ at time t in location ℓ. Moreover, $\rho(\varphi, \mathbf{x}, \ell) := \rho(\varphi, \mathbf{x}, 0, \ell)$.

The definition for the surround operator is essentially obtained from the Boolean semantics by replacing conjunctions and universal quantifications with the minimum and disjunctions and existential quantifications with the maximum, as done for STL.

The satisfaction score has some fundamental properties: if $\rho(\varphi, \mathbf{x}, t, \ell) > 0$, then $(\mathbf{x}, t, \ell) \vDash \varphi$, and similarly if $\rho(\varphi, \mathbf{x}, t, \ell) < 0$, then $(\mathbf{x}, t, \ell) \nvDash \varphi$. The absolute value $\mid \rho(\varphi, \mathbf{x}, t, \ell) \mid$, instead, gives a measure of the strength of the truth value.

6.4 Offline Monitoring Algorithms

In this section, we present the offline monitoring algorithms to check the validity of a formula φ on a trace $\mathbf{x}(t, \ell)$, in case of discrete spaces, described as graphs. The monitoring procedures extend the property monitors introduced in (MN04) for the Boolean semantics and in (DFM13) for the quantitative semantics, briefly described in Chapter 3.

As for STL formulae, our algorithms work with a bottom-up approach on the syntax tree of φ, iteratively computing the temporal signals of each subformula. Each node of the tree represents a subformula, the leaf are the atomic propositions and the root represents the whole formula. Given a spatio-temporal trace $\mathbf{x}(t, \ell)$, the algorithm starts computing the spatio-temporal Boolean/quantitative signals of all the atomic propositions, then it goes up on the tree computing the spatio-temporal Boolean/quantitative signals of a node using the signals of its child and a specific algorithm for each operator of the logic. Finally, the spatial Boolean/quantitative satisfaction function corresponds to the value of the signal at time zero $\rho_{\varphi}(0, \ell)$. An example of the the procedure can be seen in Figure 6.2.

In the case of the Boolean semantics, for each subformula ψ, it constructs a spatio-temporal signal s_{ψ} s.t. $s_{\psi}(\ell, t) = 1$ iff the subformula is true in position ℓ at time t. In the case of the quantitative semantics, for each subformula ψ, the signal s_{ψ} corresponds to the value of the quantitative satisfaction function ρ, for any time t and location ℓ. Here, we discuss in detail the algorithms to check the new spatial operators:

the somewhere and surround operators; the procedures for the others Boolean and temporal operators are similar to the STL and will be just quickly recalled. The processing of the somewhere operator is a simple extension of the disjunction operator. The treatment of the bounded surround modality $\psi = \varphi_1 \mathcal{S}_{[w_1,w_2]} \varphi_2$, instead, deviates substantially from all these procedures and will be discuss more in detail. In particular, in the following, we will present two recursive algorithms to compute the Boolean and the quantitative satisfaction, assuming the Boolean/quantitative signals of φ_1 and φ_2 being known.

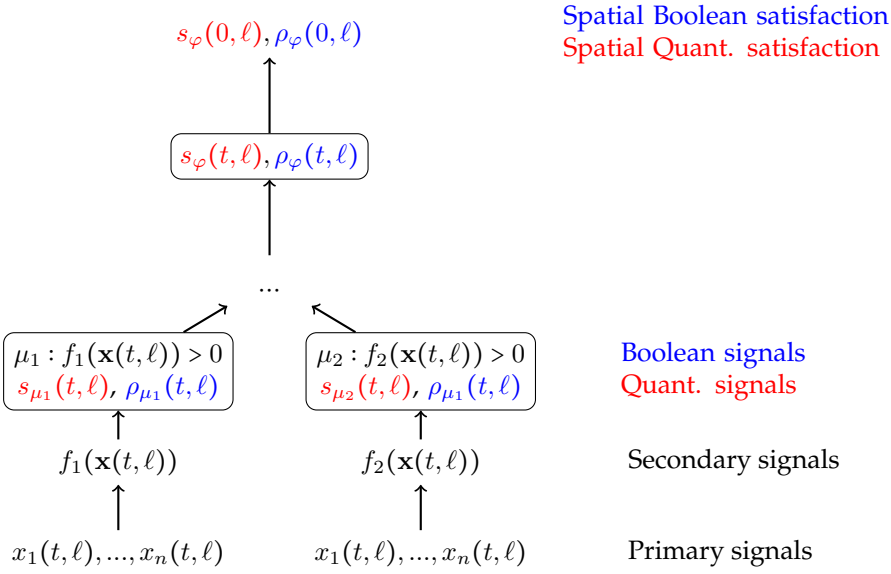


Figure 6.2: The monitoring procedure of an SSTL formula φ.

6.4.1 Boolean Semantics

The algorithm proceeds inductively bottom-up on the parse tree of the formula. Given a formula φ, to determine if $(\mathbf{x}, \ell) \vDash \varphi$, we construct, for every sub formula ψ, a Boolean signal $s_\psi : [0,T] \times L \to \mathbb{B}$ s.t. $s_\psi(t,\ell) =$

1 iff $(\mathbf{x}, t, \ell) \vDash \psi$ and 0 otherwise. At the termination of the algorithm, we have the signal $s_\phi(t, \ell)$ whose value at $t = 0$ determines if the trace $\mathbf{x}$ satisfies φ in location ℓ. The properties can be verified pointwise over each location and each time independently, due to Definition 3.2: $(\mathbf{x}, \ell, t) \vDash \varphi$ means "the trace $\mathbf{x}$ in location ℓ at time t satisfies the property ϕ ".

To optimise the monitoring procedure, we decide to split the time domain according to the *minimal interval covering* $\mathcal{I}_{s_1,\dots,s_n}$ *consistent with a set of temporal Boolean signals* $s_1, \dots s_n$, as in (MN04). We recall that a temporal Boolean signal is a function $s : [0, T] \to \mathbb{B}$. Note that, we can represent the signal $s_\psi : [0, T] \times L \to \mathbb{B}$ as a finite collection of temporal signals $\{s_{\psi,\ell}\}_{\ell \in L}$ where $s_{\psi,\ell}(t) \coloneqq s_\psi(\ell, t)$.

Definition 4 *Given an interval I, and a set of temporal signals $s_1, \dots s_n$ with $s_i : I \to \mathbb{B}$, the* **minimal interval covering** $\mathcal{I}_{s_1,\dots,s_n}$ **of** I **consistent with the set of signals** $s_1, \dots, s_n$ *is the shortest finite sequence of left-closed right-open intervals $I_1, ..., I_h$ such that $\bigcup_j I_j = I$, $I_i \bigcap I_j = \varnothing$, $\forall i \neq j$, and for $k \in \{1, \dots, n\}$, $s_k(t) = s_k(t')$ for all t, t' belonging to the same interval* [1]. *The* **positive minimal interval covering** *of s is $\mathcal{I}_s^+ = \{I \in \mathcal{I}_s \big| \forall t \in I : s(t) = 1\}$.*

The positive interval $\mathcal{I}^+_{s_{\psi,\ell}}$ corresponds to the *satisfaction set* of the formula over the signal $s_{\psi,\ell}$. Futhermore, any signal can be written as $s = s_1 \vee s_2 \vee \dots \vee s_k$ where each s_i is an *unitary signal*, meaning that it has a singleton positive interval, i.e., $\mathcal{I}^+_{s_i} = \{[t_1, t_2)\}$ for some $t_1 < t_2 \in \mathbb{R}_{\geqslant 0}$. Finally, $\mathcal{I}_{s_{\psi,\ell}} = \mathcal{I}^+_{s_{\psi,\ell}} \bigcup \mathcal{I}^-_{s_{\psi,\ell}}$ and $\mathcal{I}^+_{s_{\psi,\ell}} \bigcap \mathcal{I}^-_{s_{\psi,\ell}} = \varnothing$.

Using these definitions of signals, interval coverings, and satisfaction set, the procedure for the classic operators of STL is similar to the one described in paper (MN04). We briefly recall it in the following and then we describe the new spatial operators.

Atomic Predicates: $\psi = \mu$. The computation of the Boolean signal associated with an atomic predicate is a direct application of Definition 2: $s_{\mu,\ell}(t) = \mu(\mathbf{x}(t, \ell))$.

Negation: $\psi = \neg\phi$, then $\mathcal{I}^+_{s_{\neg\phi},\ell} = \mathcal{I}^-_{s_\phi,\ell}$.

Disjunction: $\psi = \phi_1 \vee \phi_2$, then, given $s_{\phi_1,\ell}$, $s_{\phi_2,\ell}$, let $\mathcal{I}$ be the minimal interval covering consistent with *both* signals. For each $I_i \in \mathcal{I}$, we construct

[1] The fact that we can always obtain a finite interval covering is a consequence of the restriction to closed intervals $[t_1, t_2]$, $t_1 < t_2$, in STL. For more detail about signals and interval covering see Chapter 3 or (MN04).

the signal $s_{\psi,\ell}(I_i) = s_{\phi_1,\ell}(I_i) \vee s_{\phi_2,\ell}(I_i)$ and we merge adjacent positive intervals to obtain $\mathcal{I}^+_{\psi,\ell}$.

Until: $\psi = \phi_1 \mathcal{U}_{[a,b]} \phi_2$. As we are working with future temporal modalities, we need to shift intervals *backwards*. This has to be done independently for each unitary signal, then taking the union of the so obtained satisfaction sets. Given two unitary signals p and q, the signal $\psi = p\mathcal{U}_{[a,b]}q$ is the unitary signal such that $\mathcal{I}^+_\psi = \{((I_p \cap I_q) \ominus [a,b]) \bigcap I_p\}$, where $[m,n) \ominus [a,b] = [m-b, n-a) \bigcap [0,T]$ is the Minkowski sum. In the general case, let $s_{\phi_1,\ell} = p_1 \vee \cdots \vee p_n$ and $s_{\phi_2,\ell} = q_1 \vee \cdots \vee q_m$ be signals written as union of unitary signals, then $\psi = s_{\phi_1,\ell} \mathcal{U}_{[a,b]} s_{\phi_2,\ell} = \bigvee_{i=1}^n \bigvee_{j=1}^m p_i \mathcal{U}_{[a,b]} q_j$. The proof of this result can be found in (MN04).

Somewhere: $\psi = \diamondsuit_{[d_1,d_2]}\varphi$. As remarked at the beginning of this section, and relying on the fact that we have a finite number of locations, we can process independently each location in the signal. Given the signal $s_{\psi,\ell}$, for a *fixed location* ℓ, we can rewrite the spatial operator as a disjunction between all signals in locations ℓ' s.t. $d_1 \leqslant d(\ell',\ell) \leqslant d_2$. This allows us to use the monitoring procedure for disjunction, constructing the minimal interval covering $\mathcal{I}$ consistent with all $s_{\phi,\ell'}$ signals s.t. $d_1 \leqslant d(\ell',\ell) \leqslant d_2$, and defining, for each $I_i \in \mathcal{I}$:

$$s_{\psi,\ell}(I_i) = \bigvee_{d_1 \leqslant d(\ell',\ell) \leqslant d_2} s_{\varphi,\ell'}(I_i).$$

The satisfaction set $\mathcal{I}^+_{s_{\psi,\ell}}$ is then the union of the positive I_i (i.e., I_i s.t. $s_{\psi,\ell}(I_i) = 1$), merging adjacent positive intervals.

We stress here that the introduction of the spatial somewhere operator is not merely syntactic sugar, for two reasons. First, its definition can be applied also to countable discrete spaces, and it can be easily generalised to continuous spaces. Secondly, even assuming a finite discrete space, expanding it as a disjunction would produce a blowup of the formula size *exponential* in the nesting level of spatial operators, and hence an exponential increase in the complexity of the monitoring procedure.

Surround: $\psi = \varphi_1 \mathcal{S}_{[d_1,d_2]} \varphi_2$. Algorithm 4 presents the procedure to monitor the Boolean semantics of ψ in a location ℓ, returning the temporal Boolean signal $s_{\psi,\ell}$ of ψ at location ℓ. The algorithm first computes the

set of locations $L^{\ell}_{[0,d_2]}$ that are at distance d_2 or less from ℓ, and then, recursively, the temporal Boolean signals $s_{\varphi_1,\ell'}$ and $s_{\varphi_2,\ell'}$, for $\ell' \in L^{\ell}_{[0,d_2]}$. These signals provide the satisfaction of the sub-formula φ_1 and φ_2 at each point in time, and for each location of interest. Then, a minimal interval covering consistent to all the signals $s_{\varphi_1,\ell'}$ and $s_{\varphi_2,\ell'}$ is computed, and to each such interval, a core procedure similar to that of (CLLM14) is applied. More specifically, we first compute the set of locations W in which both φ_1 and φ_2 are false, and that are in the external boundary of the locations that satisfy φ_1 (V) or φ_2 (Q). The locations in W are "bad" locations, that cannot be part of the external boundary of the set A of φ_1-locations which has to be surrounded only by φ_2-locations. Hence, the main loop of the algorithm removes iteratively from V all those locations that have a neighbour in W (set N, line 13), constructing a new set T containing only those locations in N that do not satisfy φ_2, until a fixed point is reached. As each location can be added to W and be processed only once, the complexity of the algorithm is linear in the number of locations and linear in the size of the interval covering. Correctness can be proven in a similar way as in (CLLM14) and is reported in Appendix A.1.

6.4.2 Quantitative Semantics

The quantitative semantics for STL is defined for arbitrary signals, but algorithms are provided only for piecewise linear continuous ones (DM10; DFM13), considered as the interpolation of continuous functions. Here, we deviate from this interpretation, and consider instead a simpler interpolation based on piecewise constant signals, as described below.

Piecewise constant approximation of quantitative signals. We discretise the time domain with step $h > 0$, so that our signals in each location ℓ, $s_\ell : [0,T] \to \mathbb{R}$, are represented by the finite set $\{s_\ell(0), s_\ell(h), \ldots, s_\ell(mh)\}$, where $mh = T$. Then, the piecewise constant approximation of $s_\ell(t)$ is the signal $\hat{s}_\ell(t) = s_\ell(kh)$ for $t \in [kh, (k+1)h)$. We further assume, without loss of generality[2], that all time bounds appearing in the

[2]Time bounds can be restricted to rational numbers, hence there always exists an $h > 0$

Algorithm 2 Boolean monitoring for the surround operator

input $\ell, \psi = \varphi_1 \mathcal{S}_{[d_1,d_2]} \varphi_2$
$\forall \ell' \in L^{\ell}_{[0,d_2]}$ compute $s_{\varphi_1,\ell'}, s_{\varphi_2,\ell'}$
compute $\mathcal{I}_{s_{\psi,\ell}}$ {the minimal interval covering consistent with $s_{\varphi_1,\ell'}, s_{\varphi_2,\ell'}$, $\ell' \in L^{\ell}_{[0,w_2]}$}
for all $I_i \in \mathcal{I}_{s_{\psi,\ell}}$ **do**
  $V = \{\ell' \in L^{\ell}_{[0,d_2]} | s_{\varphi_1,\ell'}(I_i) = 1\}$
  $Q = \{\ell' \in L^{\ell}_{[d_1,d_2]} | s_{\varphi_2,\ell'}(I_i) = 1\}$
  $T = B^+(Q \bigcup V)$
  while $W \neq \varnothing$ **do**
    $W' = \varnothing$
    for all $\ell \in W$ **do**
      $N = pre(\ell) \bigcap V = \{\ell' \in V | \ell E \ell'\}$
      $V = V \backslash N$
      $W' = W' \bigcup (N \backslash Q)$
    end for
    $W = W'$
  end while
  $s_{\psi,\ell}(I_i) = \begin{cases} 1 & \text{if } \ell \in V, \\ 0 & \text{otherwise.} \end{cases}$
end for
merge adjacent positive interval I_i, i.e., I_i s.t. $s_{\psi,\ell}(I_i) = 1$
return $s_{\psi,\ell}$

temporal operators of a SSTL formula are multiples of h.

Under the assumption that secondary signals are Lipschitz continuous[3], and letting K be the maximum of their individual Lipschitz constants, the following properties hold: (a) $s_\ell(kh) = \hat{s}_\ell(kh)$; and (b) $\|s_\ell(t) = \hat{s}_\ell(t)\| \leq Kh/2$, uniformly in t.

Monitoring the quantitative semantics.

We now turn to the monitoring algorithm for the quantitative semantics, assuming the input is a piecewise constant signal, where the time domain has been discretised with step h.

Monitoring Boolean operators is straightforward, we just need to apply the definition of the quantitative semantics pointwise in the discretisation. The time bounded until operator can also be easily computed by replacing the min and max over dense real intervals in its definition by the corresponding min and max over the corresponding finite grid of time points. In this case, however, we can introduce an error due to the discrete approximation of the Lipschitz continuous signal in intermediate points, yet this error accumulates at a rate proportional to Kh, where K is the previously defined Lipschitz constant.

Monitoring the somewhere operator $\diamondsuit_{[d_1,d_2]}\varphi$ is also immediate: once the location ℓ of interest is fixed, similarly to the Boolean semantics, we can just turn it into a disjunction of the signals $s_{\varphi,\ell'}$ for each location $\ell' \in L^{\ell}_{[d_1,d_2]}$.

The only non-trivial monitoring algorithm is the one for the spatial surround operator, which we discuss below. However, as the satisfaction score is computed at each time point of the discretisation and depends on the values of the signals at that time point only, this algorithm introduces no further error w.r.t. the time discretisation. Hence, we can globally bound the error introduced by the time discretisation:

Proposition 6.1 *Let the primary signal* $\mathbf{x}$ *be Lipschitz continuous, as the functions defining the atomic predicates. Let* K *be a Lipschitz constant for all sec-*

satisfying all assumptions.

[3]The assumption of Lipschitz continuity holds whenever the primary signal is the solution of an ODE with a locally Lipschitz vector field, as usually is the case.

ondary signals, and h be the discretisation step. Given a SSTL formula φ, let $u(\varphi)$ count the number of temporal until operators in φ, and denote by $\rho(\varphi, \mathbf{x})$ its satisfaction score over the trace $\mathbf{x}$ and by $\rho(\varphi, \hat{\mathbf{x}})$ the satisfaction score over the discretised version $\hat{\mathbf{x}}$ of $\mathbf{x}$ with time step h. Then $\|\rho(\varphi, \mathbf{x}) - \rho(\varphi, \hat{\mathbf{x}})\| \leq u(\varphi)Kh$.

Proof: We first observe that the monitoring algorithm for Boolean and spatial operators preserve the error of the input quantitative signals. This means that if $\|s_{\varphi_j,\ell} - \hat{s}_{\varphi_j,\ell}\| \leq \varepsilon$, then $\|s_{\psi,\ell} - \hat{s}_{\psi,\ell}\| \leq \varepsilon$, for ψ one of $\neg\varphi_1$, $\varphi_1 \wedge \varphi_2$, $\varphi_1 \mathcal{S}_{[d_1,d_2]} \varphi_2$, $\diamondsuit_{[d_1,d_2]}\varphi_1$. Hence, temporal discretisation introduces errors only for temporal operators.

Now, let $I = [t_1, t_2]$ be such that $t_j = k_j h$, and denote the Minkowski sum by $\oplus$, so that $t \oplus I = [t + t_1, t + t_2]$. Denote by $\hat{I}$ the discretised version of I, with step h, $\hat{I} = \{k_1 h, (k_1 + 1)h, \ldots, k_2 h\}$. We observe two facts for the maximum, with identical statements holding for the minimum.

- Let $f(t)$ be Lipschitz with constant K. Let $g(t) = \max_{t' \in t \oplus I} f(t)$ and $\hat{g}(t) = \max_{t' \in t \oplus \hat{I}} f(t)$. Then $\|g(t) - \hat{g}(t)\| \leq Kh/2$. This holds by applying the Lipschitz property between a generic point in $t \oplus I$ and the closest point in $t \oplus \hat{I}$, and noting that the maximum distance between such points is $h/2$.
- If $\tilde{f}$ is such that $\|\tilde{f}(t) - f(t)\| \leq \varepsilon$ uniformly in t, and we let g, $\hat{g}$ as above, and $\tilde{g}(t) = \max_{t' \in t \oplus \hat{I}} \tilde{f}(t)$, then

$$\|g(t) - \tilde{g}(t)\| \leq \|g(t) - \hat{g}(t)\| + \|\hat{g}(t) - \tilde{g}(t)\| \leq Kh/2 + \varepsilon.$$

Hence, the second property implies that the additional error we introduce by evaluating a time bounded until is an additive term no larger than Kh, as in the definition of the quantitative semantics of the until, there are a nested minimum and a maximum over dense time intervals. Hence the total error will be bounded by Kh times the number of temporal operators. ∎

Monitoring Algorithm for the Surround. The quantitative monitoring procedure for the bounded surround operator is shown in Algorithm 3. Similarly to the Boolean case, the algorithm for the surround formula $\psi =$

$\varphi_1 \mathcal{S}_{[d_1,d_2]} \varphi_2$ takes as input a location ℓ and returns the quantitative signal $s_{\psi,\ell}$, or better its piecewise constant approximation with time-step h (an additional input, together with the signal duration T). As a first step, it computes recursively the quantitative satisfaction signals of subformula φ_1, for all locations $\ell' \in L^{\ell}_{[0,d_2]}$, and of subformula φ_2, for all locations $\ell' \in L^{\ell}_{[d_1,d_2]}$. Furthermore, it sets all the quantitative signals for φ_1 and φ_2 for the other locations to the constant signal equal to minus infinity. The algorithm runs a fixpoint computation for each time instant in the discrete time set $\{0, h, 2h, \ldots, mh\}$. The procedure is based on computing a function $\mathcal{X}$, with values in the extended reals $\mathbb{R}^*$, which is executed on the whole set of locations L, but for the modified signals equal to $-\infty$ for locations not satisfying the metric bounds for ℓ. The function $\mathcal{X}$ is defined below.

Definition 5 *Given a finite set of locations L and two functions $s_1 : L \to \mathbb{R}^*, s_2 : L \to \mathbb{R}^*$. The function $\mathcal{X} : \mathbb{N} \times L \to \mathbb{R}$ is inductively defined as:*

1. $\mathcal{X}(0, \ell) = s_1(\ell)$
2. $\mathcal{X}(i+1, \ell) = \min(\mathcal{X}(i, \ell), \min_{\ell' | \ell E \ell'}(\max(\mathcal{X}(i, \ell'), s_2(\ell'))))$

The algorithm then computes the function $\mathcal{X}$ iteratively, until a fixed-point is reached.

Theorem 6.1 *. Let s_1 and s_2 be as in Definition 5, and*

$$s(\ell) = \max_{A \subseteq L, \ell \in A} (\min(\min_{\ell' \in A} s_1(\ell'), \min_{\ell' \in B^+(A)} s_2(\ell'))),$$

then

$$\lim_{i \to \infty} \mathcal{X}(i, \ell) = s(\ell), \qquad \forall \ell \in L.$$

Moreover, $\exists K > 0$ s. t. $\mathcal{X}(j, \ell) = s(\ell), \forall j \geq K$.

The following corollary provides the correctness of the method. It shows that, when $\mathcal{X}$ is computed for the modified signals constructed by the algorithm, it returns effectively the quantitative satisfaction score of the spatial surround.

Algorithm 3 Quantitative monitoring for the surround operator

inputs: $\ell, \psi = \varphi_1 \mathcal{S}_{[d_1,d_2]} \varphi_2$, h, T
for all $\ell' \in L$ **do**
  if $0 \leq d(\ell, \ell') \leq w_2$ **then**
    compute $s_{\varphi_1,\ell'}$
    if $d(\ell, \ell') \geq d_1$ **then**
      compute $s_{\varphi_2,\ell'}$
    else $s_{\varphi_2,\ell'} = -\infty$
  else $s_{\varphi_1,\ell'} = -\infty, s_{\varphi_2,\ell'} = -\infty$
end for
for all $t \in \{0, h, 2h, \ldots, T\}$ **do**
  for all $\ell' \in L$ **do**
    $\mathcal{X}_{prec}(\ell') = +\infty$
    $\mathcal{X}(\ell') = s_{\varphi_1,\ell}(t)$
  end for
  while $\exists \ell' \in L,$ s.t. $\mathcal{X}_{prec}(\ell') \neq \mathcal{X}(\ell')$ **do**
    $\mathcal{X}_{prec} = \mathcal{X}$
    for all $\ell' \in L$ **do**
      $\mathcal{X}(\ell') = \min(\mathcal{X}_{prec}(\ell'), \min_{\ell'' | \ell' E \ell''}(\max(s_{\varphi_2,\ell''}(t), \mathcal{X}_{prec}(\ell''))))$
    end for
  end while
  $s_{\psi,\ell}(t) = \mathcal{X}(\ell)$
end for
return $s_{\psi,\ell}$

Corollary 6.1 *Given an $\hat{\ell} \in L$, let $\psi = \varphi_1 \mathcal{S}_{[d_1,d_2]} \varphi_2$ and*

$$s_1(\ell) = \begin{cases} \rho(\phi_1, \mathbf{x}, t, \ell) & \text{if } 0 \leq d(\hat{\ell}, \ell) \leq d_2 \\ -\infty & \text{otherwise.} \end{cases}$$

$$s_2(\ell) = \begin{cases} \rho(\phi_2, \mathbf{x}, t, \ell) & \text{if } d_1 \leq d(\hat{\ell}, \ell) \leq d_2 \\ -\infty & \text{otherwise.} \end{cases}$$

Then $\rho(\psi, \mathbf{x}, t, \hat{\ell}) = s(\hat{\ell}) = \max_{A \subseteq L, \hat{\ell} \in A} (\min(\min_{\ell \in A} s_1(\ell), \min_{\ell \in B^+(A)} s_2(\ell)))$.

In order to discuss the complexity of the monitoring procedure, we need an upper bound on the number of iterations of the algorithm. This is given by the following.

Proposition 6.2 *Let d_G be the diameter of the graph G and $\mathcal{X}(\ell)$ the fixed point of $\mathcal{X}(i, \ell)$, then $\mathcal{X}(\ell) = \mathcal{X}(d_G + 1, \ell)$ for all $\ell \in L$.*

It follows that the computational cost for each location is $O(d_G|L|m)$, where m is the number of sampled time-points. The cost for all locations is therefore $O(d_G|L|^2 m)$.

The proofs of Theorem 6.1, Corollary 6.1 and Proposition 6.2 are reported in Appendix A.2.

6.4.3 Stochastic Semantics

The extension of the definition of the SSTL semantics to stochastic systems follows directly from the definition of the stochastic semantics of STL, defined in the previous chapter. Remember that the verification of a spatio-temporal property is done over each specific location. Given a spatio-temporal trace $\mathbf{x}(t, \ell)$ and a property ϕ when we check its satisfaction or robustness score what we obtain are a satisfaction and a robustness function, i.e., we have a value for each location. Let us denote by $\mathcal{D}$ the space of all possible trajectories of the stochastic spatio-temporal process. It can be seen as the space of the cadlag functions $\mathcal{D}([0, \infty], \mathbb{D})$, where $\mathbb{D} = \mathbb{D}_1 \bigoplus \cdots \bigoplus \mathbb{D}_m$, with $m = |L|$ the number of locations and $\mathbb{D}_i$ the state space of the temporal trajectories in location ℓ_i. We denote by $\mathcal{D}_{\ell_i}$ the projection on the $\mathbb{D}_i$ state space of the location ℓ_i.

$\mathbf{X}$ induces a probability measures on $\mathcal{D}$. We can interpret the Boolean and the quantitative satisfaction function of ϕ as functionals on the space $\mathcal{D}$, assigning to each trajectory $\mathbf{x} \in \mathcal{D}$ its corresponding set of truth values according to the Boolean semantics or its set of satisfaction degrees according to the quantitative semantics, one for each location. In the Boolean case, the functional is $I_\phi : \mathcal{D} \to \{0,1\}^m$, such that $I_\phi(\mathbf{x})[\ell] = 1$ if and only if $(\mathbf{x}, \ell) \vDash \phi$, where $\ell \in L$. Hence, it identifies, in each location, the subset of temporal trajectories that satisfy the formula ϕ. It follows that

$$P(\phi, \ell) = \mathbb{P}\{(\mathbf{x}, \ell) \in \mathcal{D}_\ell \mid I_\phi(\mathbf{x})[\ell] = 1\} = \mathbb{P}\{(\mathbf{x}, \ell) \in \mathcal{D}_\ell \mid (\mathbf{x}, \ell) \vDash \phi\}.$$

In the quantitative case, the robustness (satisfaction degree) function $\rho(\phi, \mathbf{x}, \ell)$ can be seen as a functional $\mathtt{R}_\phi$ from the trajectories in $\mathcal{D}$ to $\mathtt{R}^{\mathtt{m}}$, defined as $R_\phi(\mathbf{x})[\ell] = \rho(\phi, \mathbf{x}, \ell) = \rho(\phi, \mathbf{x}, 0, \ell)$. If $\mathtt{R}_\phi$ is measurable, we can define, then, a set of real-valued marginal random variables $R_\phi[\ell] = \mathtt{R}_\phi(\mathbf{X})[\ell], \ell \in \mathtt{L}$, with probability distribution:

$$\mathbb{P}\left(\mathtt{R}_\phi(\mathbf{X})[\ell] \in [\mathtt{a}, \mathtt{b}]\right) = \mathbb{P}\left(\mathbf{X}_\ell \in \{(\mathbf{x}, \ell) \in \mathcal{D}_\ell \mid \rho(\phi, \mathbf{x}, \ell) \in [a, b]\}\right).$$

Applying this definition to a stochastic spatio-temporal model, we obtain a distribution of robustness degrees. From these distributions we can compute the average robustness in each location $\mathbb{E}(R_\phi[\ell])$ that gives a measure of how strongly a formula is satisfied.

After that, we can use a *Bayesian statistical model checking* approach to estimate the satisfaction probability, described in Chapter 4 and classic statistical tools for the average robustness degree, as in Chapter 5.

To prove that $\mathtt{R}_\phi$ is measurable, we can first note that the new spatial operators do not increase the time horizone $T_\phi = \max_{\ell \in L}\{T_\phi[\ell]\}$ of the formula and that they work in a similar way as the conjunction and disjunction operators. Indeed, $T_{\diamondsuit_{[d_1,d_2]}\phi}[\ell] = \max_{\ell' | d(\ell,\ell') \in [d_1,d_2]} T_\phi[\ell']$, and a similar formula can be derived for the surround operator.

Then, we have to extend Lemma 5.3 to SSTL formulae. Let ϕ be an SSTL formula. Let $\hat{\mathtt{R}}_\phi$ be the functional associated with it, $\hat{\mathtt{R}}_\phi : \mathcal{D}([0,T], \mathbb{D}) \to \mathcal{D}([0, T - T_\phi], \mathbb{D})$, s.t. $\hat{\mathtt{R}}_\phi(\mathbf{x})(t)[\ell] = \rho(\phi, \mathbf{x}, t, \ell)$. To prove that $\hat{\mathtt{R}}_\phi$ is

measurable, we have to extend the measurability on the spatial formulas. This means we have to prove that $\hat{\mathtt{R}}_{\diamondsuit_{[d_1,d_2]}\phi}[\ell]$ and $\hat{\mathtt{R}}_{\phi_1\mathcal{S}_{[d_1,d_2]}\phi_2}[\ell]$ are measurable for an arbitrary location ℓ, supposing $\hat{\mathtt{R}}_{\phi}$, $\hat{\mathtt{R}}_{\phi_1}$, and $\hat{\mathtt{R}}_{\phi_2}$ measurable (for the structural induction).

Somewhere $\diamondsuit_{[d_1,d_2]}\phi$**.** $\hat{\mathtt{R}}_{\phi_1\wedge\phi_2} = \max_{\ell' | d(\ell,\ell')\in[d_1,d_2]} \hat{\mathtt{R}}_{\phi}[\ell']$, which is measurable in virtue of Lemma 5.2 b), of the fact that measurability composing measurable functions and of the fact that the set L is finite.

Surround $\phi_1\mathcal{S}_{[d_1,d_2]}\phi_2$**.** The procedure is similar. Indeed, $\hat{\mathtt{R}}_{\phi_1\mathcal{S}_{[d_1,d_2]}\phi_2} = \max_A \min\{\min_{\ell'\in A}\{\hat{\mathtt{R}}_{\phi_1}[\ell']\} \min_{\ell'\in B^+(A)}\{\hat{\mathtt{R}}_{\phi_2}[\ell'']\}\}$, where A is a finite set of locations of L. This is again measurable by the same arguments above, and the fact that there is a finite number of sets A.

The measurability of $\mathtt{R}_{\phi} : \mathcal{D}([0,\mathtt{T}],\mathbb{D}) \to \mathbb{R}^{\mathtt{m}}$ then follows from the fact that $\mathtt{R}_{\phi} = \pi_0 \circ \hat{\mathtt{R}}_{\phi}$, i.e., $\mathtt{R}_{\phi}(\mathbf{x})[\ell] = \rho(\phi,\mathbf{x},\ell) = \rho(\phi,\mathbf{x},0,\ell) = \hat{\mathtt{R}}_{\phi}(\mathbf{x})(0)[\ell]$.

6.5 Case Studies

6.5.1 Spread of a Cholera Infection

As first case study, we consider a model of a cholera outbreak. Cholera is an infection of the intestine caused by the bacterium Vibrio cholerae and a prominent example of a waterborne disease (BAM+08; MBR+12). Typically, the infection is transmitted by contaminated food or water. An explicit modelling of the hydrological space where the infection spreads is therefore a crucial aspect to understand and analyse this kind of disease. For this reason this example is very suitable to show the potentiality of this logic, here we will see how our logic can easily describe elaborated spatio-temporal behaviours by means of the spatial operators. In particular, with this example, we will explore the potentiality of the somewhere operator.

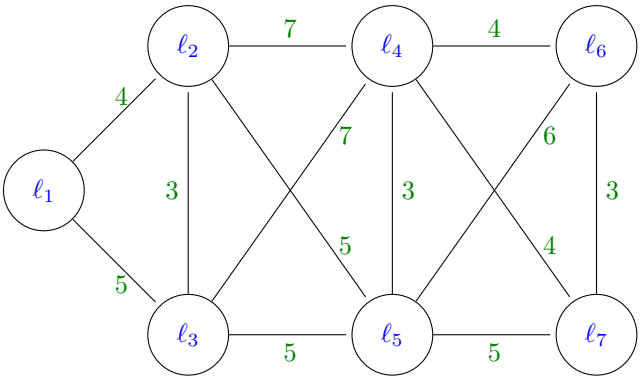


Figure 6.3: The graph of the population spatial distribution /locations distribution. Note that this is not the graph of the patch-PCTMC model but only the graph of the locations. The nodes ℓ_1, ..., ℓ_7 represent the different communities; the edges represent the connection between the communities through the water basin. The green numbers correspond to the values w of the distance between locations.

Model. We represent space as a weighted graph, shown in Figure 6.3. The nodes represent the human communities and the edges describe the links between water basins of different areas. The idea is to analyse the diffusion of an epidemic along the communities that live close to a river. There are two agent classes: the bacteria and the individuals. The bacteria have only one state (B) but they can be transported to different nodes via the river. An individual, instead, can be in three different states: susceptible (S) infected (I) and recovered (R), but cannot change location. Ignoring human mobility is a simplification justified by the fact that we are considering communities that live in specific places. Furthermore, here our focus is to illustrate the logic at work, rather than presenting a fully realistic model. Extension to more complex scenarios, however, are relatively straightforward (MBR+12), and can be easily described within our formal framework, so that analogous or more complex spatio-temporal logical properties can still be checked.

The model of this system has state variables X_S, X_I, X_R, X_B, counting the number of susceptible, infected, and recovered individuals, and

the concentration of the bacteria in each location. Discrete space is described by the graph in Figure 6.3, which is equipped with a weight function $w : E \to \mathbb{R}$ that returns the cost of each edge. In this specific case, we interpret the cost $w(\ell_i, \ell_j)$ of an edge between nodes ℓ_i and ℓ_j as the distance between them. We compute, then, the distance function $d : L \times L \to \mathbb{R}$ that evaluates the shortest distance for each pair of locations in L. So, for example, in Figure 6.3, $d(\ell_1, \ell_5) = w(\ell_1, \ell_2) + w(\ell_2, \ell_5) = 9$. The movement of bacteria from a location ℓ_i to a directly connected location ℓ_j is specified by the inter-patch transitions $\nu_{mov} = (mov, B, g_{mov})$. Here $g_{mov}(\mathbf{X}(t, \ell), \ell_i, \ell_j) = l p_{ij} X_B(t, \ell_i)$ is the rate function, where l is the total water flow (assumed equal for all nodes) and p_{ij} is the fraction of water flowing out of node ℓ_i which reaches node ℓ_j. Such a rate function describes the bacteria mobility as a passive mobility due to the water flow. The probability $p_{i,j}$ to go from ℓ_i to ℓ_j is

$$(p_{i,j})_{\ell_i, \ell_j \in L} = \begin{pmatrix} 0 & 2/5 & 3/5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2/10 & 3/10 & 1/2 & 0 & 0 \\ 0 & 2/10 & 0 & 3/10 & 1/2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3/10 & 2/5 & 3/10 \\ 0 & 0 & 0 & 3/10 & 0 & 3/10 & 2/5 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1/2 \\ 0 & 0 & 0 & 0 & 0 & 1/2 & 0 \end{pmatrix}$$

The sum of the values p_{ij} in each row is equal to one, i.e $\sum_k p_{ik} = 1$, except for the locations ℓ_6 and ℓ_7 where is less than one to capture the continuation of the river. The other transitions describe the change of the individual state, according to a model similar to the SIR epidemics model introduced in the Example 2.1, with the notable difference of the infection rate, which now depends on the concentration of bacteria in water and not directly on the number of infected individuals. We also consider transitions describing decrease or growth of the bacterium population.

The patch population model is thus $((\mathbb{S}, \mathbf{X}, \mathcal{T}), G, \mathcal{V})$, where $\mathbb{S} = \{S, I, R, B\}$ is the set of states, $\mathbf{X} = (X_S, X_I, X_R, X_B)$ is the state vector, $G = (L, E, w)$, with $L = \{\ell_1, ..., \ell_n\}$, is the graph as in Figure 6.3, $\mathcal{V} = \{(mob, B, g_{mob})\}$ is the set of inter-patch transition, as described above, and $\mathcal{T}$ is

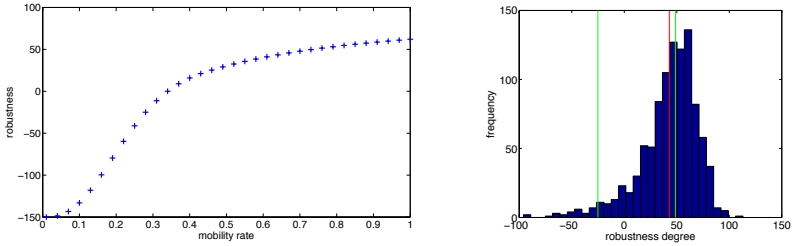
the set containg the following intra-patch transitions. More specifically:
⋆ τ_{inf} is the *infection* transition: $S + B \to I + B$ with rate $f_{inf}(\mathbf{X}) = \beta(\ell)\frac{X_B}{K+X_B}X_S$, where K is the half saturation constant, $\beta(\ell)$ represents the site-dependent rate of exposure to contaminated water,
⋆ τ_{Snat} is the *natality* transition: $\varnothing \to S$, with rate $f_{Snat}(\mathbf{X}) = \mu H(\ell)$. $H(\ell)$ is the size of the community in location ℓ, it is assumed to be at a demographic equilibrium with μ being the human mortality rate,
⋆ τ_{Smort} is the *mortality* of a susceptible individual: $S \to \varnothing$, with rate μ,
⋆ τ_{Imort} is the *mortality* of a infected individual: $I \to \varnothing$, with rate $\mu + \alpha$, where α is the mortality rate due to cholera,
⋆ τ_{rec} is the *recovery* transition: $I \to R$, with rate γ,
⋆ τ_{Bdeg} is the *degradation* transition: $B \to \varnothing$, with rate μ_B,
⋆ $\tau_{Bgrowth}$ is the *bacterial growth* transition: $\varnothing \to B$, with rate function $f_{Snat}(\mathbf{X}) = \frac{p}{W}X_I$, where p is the rate at which bacteria are produced by one infected person and W is the volume of the contaminated water.

Analysis. The first spatio-temporal behaviour that we consider is how the epidemic propagates along the river. The idea is to consider a model that starts with the infection in only one location, ℓ_1, and then to check whether the infection has propagated at a certain distance from ℓ_1 after a certain time. This behaviour can be captured by the SSTL formula:

$$\phi_1 = \mathcal{F}_{[0,T_{inf}]} \diamondsuit_{[d_1,d_2]} (X_I > c_{inf}), \tag{6.1}$$

verifying it in location ℓ_1. The exact meaning of the formula is: eventually, in less than T_{inf} unit time, the number of infected individual becomes more than c_{inf} in a location ℓ with $d(\ell_1, \ell) \in [d_1, d_2]$, i.e., at a distance from location ℓ_1 equal or greater than d_1 and equal or less than d_2.

We analyse a system with 7 locations, all with the same size, with distance between locations $d(\ell_i, \ell_j)$, in agreement with the green integer values labelling the edges in Figure 6.3. We choose the parameter values following (BAM+08). As initial variables we set, for all the locations apart ℓ_1, the same number of susceptible individuals, $X_S = 500$, and we start with zero infected individuals, $X_I = 0$; whereas, for ℓ_1 we set 100



(a) Mobility rate vs Robustness

(b) Robustness distribution

Figure 6.4: (a) Dependency of the robustness degree on the mobility rate (for the ODE interpretation of the model). (b) Empirical robustness distribution of the formula 6.1 with $d_1 = 12$ and $d_2 = 15$ and $T_{inf} = 5$, obtained from 1000 simulation runs.

infected individuals and 400 susceptible. There are no recovered individuals at the beginning. We set also the bacterium concentration equal to zero in all the locations except ℓ_1. From (BAM+08), the parameters of the model have been set as $\mu = 0.0005, H(\ell_i) = 500, \beta = 1, K = 5, \gamma = 0.2, \alpha = 0.0004, \mu_B = 0.228, p = 0.2, W = 50, l = 0.5$ while the matrix $(p_{ij})_{i,j\leq 7}$ has as non-zero entries those specified in Figure 6.3. For the formula parameters, we set $d_1 = 12$ and $d_2 = 15$, $c_{inf} = 150$ and $T_{inf} = 5$ unit times.

In Figure 6.4, we show some results of the monitoring of this formula. In the top panel, we consider the ODE interpretation and show how the robustness score increases as a function of the mobility rate. In the bottom panel, we show the empirical distribution from 1000 runs of the robustness degree from the stochastic model, with vertical lines denoting the average (red lines) and conditional averages on the formula being false/ true (green lines). Changing the mobility rate in the stochastic model from 0.3 to 0.6, the satisfaction probability varies from 0.397 to 0.975, while the average robustness score varies (monotonically) from -16.93 to 52.72. We stress here how the robustness score can be used for system design purposes, trying to robustly match spatio-temporal SSTL specifications, following, e.g., the procedure described in the previous

chapter.

In order to illustrate in more detail the expressive power of this logic, we discuss now two additional properties, using the following building blocks:

$$\psi_1 = \mathcal{F}_{[0,T_{start}]}(\diamondsuit_{[0,d_{near}]}(X_I > c_{inf})))$$
$$\psi_2 = (\mathcal{F}_{[T_{start},T_{start}+DT]} \diamondsuit_{[d_{far},d_{max}]} (X_I > c_{inf}))$$

The first one is $\boxdot_{[0,d_{max}]}(\psi_1 \longrightarrow \psi_2)$, stating that a large infection (with at least c_{inf} individuals) happening at some time $t \in [0, T_{start}]$ and localised within distance d_{near} from a given reference point, will spread further away, at a location at distance between d_{far} and d_{max}, at some time $t' \in [T_{start}, T_{start} + DT]$. Furthermore, this is true for every reference point (say at distance at most d_{max} from ℓ_1). Checking this formula on 1000 runs of the stochastic model, with parameters $d_{near} = 3$, $d_{far} = 13$, $d_{max} = 15$, $T_{start} = 1$ and $DT = 4$, we obtain a satisfaction probability of 0.9220 ± 0.0085 (all results reported at 95% confidence), and an average robustness degree of 42 ± 0.8771. The robustness score for the ODE interpretation, though, is 30.26.

The other formula we consider is $\psi_2 \longrightarrow \psi_1$, stating that a high infection level at a far-away location at a late time must have been high at a nearer location some time before, i.e., that the current reference point is close to the epicentre of the epidemic. In this case, with the same parameters of the previous formula, verifying it in location ℓ_2, we obtain a satisfaction probability of 0.997 ± 0.0017, an average robustness score of 55.48 ± 0.5677 and a robustness score of the ODE model of 58.08.

In general, we observe that the ODE interpretation generates robustness scores that are in agreement with the ones of the stochastic model, at a much cheaper computational price. This may be the effect of some convergence theorem at work, and may be exploited for system design purposes, as numerically solving ODEs is in general computationally cheaper than analysing or simulating stochastic models.

We implemented this model in `Matlab`, both as a set of differential equations and as a stochastic process. In order to compute the Boolean semantics of SSTL formulae, we exploited a dedicated Java implementation; for the quantitative semantics instead, we exploited the routines

provided by the `Breach` toolbox (Don10), adapting them to check SSTL formula. ODEs describing the dynamics of the systems, obtained from the prescriptions of Section 2.1, have been numerically integrated using a Matlab built-in solver. The stochastic interpretation of the population model, instead, has been analysed by a dedicated Java implementation, combining standard Monte Carlo simulation (by the Gillespie algorithm (Gil77)) and Bayesian statistical model checking (JCL$^+$09b). Recently, the whole monitoring procedure has been implemented in a Java toolbox, jSSTL, described in Chapter 8. All the experiments were run on a Intel Core i5 2.6 GHz CPU, with 8GB 1600 MHz RAM.

6.5.2 Pattern Formation in a Reaction-Diffusion System

With this case study, we show how SSTL can be used to identify the formation of *patterns* in a reaction-diffusion system. Patterning is a ubiquitous feature of biological organisms, and the presence of regular geometric motifs on many organisms has long fascinated scientists. Pattern formation is also the subject of one of the earliest, and most influential, computational systems biology works, Alan Turing's pioneering work on morphogenesis (Tur52a). Turing's insight was that biological patterns can be viewed as emergent behaviour (in modern terminology) arising from local interactions of microscopic agents. More precisely, Turing considered spatially distributed systems whose local concentration vector $\mathbf{u}$ obeys a *reaction-diffusion* partial differential equation (PDE)

$$\frac{\partial \mathbf{u}}{\partial t} = D\nabla^2\mathbf{u} + f(\mathbf{u}). \tag{6.2}$$

Equation (6.2) defines the time evolution of the local concentration $\mathbf{u}$ as the sum of two terms: a dispersal or diffusion term $D\nabla^2\mathbf{u}$, which globally drives the system towards a uniform equilibrium, and a reaction term $f(\mathbf{u})$, which accounts for local interactions of the chemicals. Turing then proved that, under certain conditions on the reaction/ diffusion parameters, these two counteracting processes could give rise to regular patterns of concentration, providing a plausible mechanistic model of biological pattern formation.

Turing's ideas have been empirically demonstrated in many areas of biochemistry (see (MWB+12) for a recent review), and are still influential in particular in the field of developmental biology (see, e.g., (FI14) for a recent paper building on these ideas). The crucial idea in the application of reaction-diffusion systems to development is that these mechanisms would underpin the local concentration patterns of regulatory proteins, which would instruct different genetic programs to be executed at different spatial locations. These special regulatory proteins are called *morphogens* in developmental biology, as they are believed to be responsible for the establishment of the shape of an organism in higher organisms.

The natural analogue, systems of agents moving in continuous space, is however prohibitively expensive computationally; an approach that is more amenable to analysis is to discretise space into a number of cells (voxels) which are assumed to be spatially homogenous, and to replace spatial diffusion with transitions between different cells.

From the point of view of formal verification, the formation of patterns is an inherently spatio-temporal phenomenon, in that the relevant aspect is how the spatial organisation of the system changes over time.

Model. Our model, similar to (GBB14; HJK+15), describes the production of skin pigments that generate spots in animal furs. The reaction-diffusion system is discretised, according to a Finite Difference scheme (Olv14), as a system of ODEs whose variables are organised in a $K \times K$ rectangular grid. More precisely, we treat the grid as a weighted undirected graph, where each cell $(i, j) \in L = \{1, \ldots, K\} \times \{1, \ldots, K\}$ is a location (node), edges connect each pairs of neighbouring nodes along four directions (so that each node as at most 4 adjacent nodes), and the weight of each edge is always equal to the spatial length-scale δ of the system[4]. We consider

[4]For simplicity, here we fix $\delta = 1$. Note that using a non-uniform mesh the weights of the edges of the resulting graph will not be uniform.

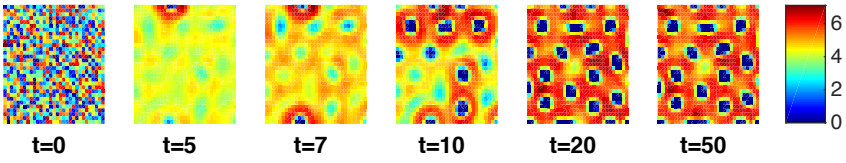


Figure 6.5: Value of x^A for the system (6.3) for $t = 0, 5, 7, 12, 20, 50$ time units with parameters $K = 32, R_1 = 1, R_2 = -12, R_3 = -1, R_4 = 16, D_1 = 5.6$ and $D_2 = 25.5$. The initial condition has been set randomly. The colour map for the concentration is specified in the legend on the right.

two species A and B in a $K \times K$ grid, obtaining the system:

$$\begin{cases} \frac{dx^A_{i,j}}{dt} = R_1 x^A_{i,j} x^B_{i,j} - x^A_{i,j} + R_2 + D_1(\mu^A_{i,j} - x^A_{i,j}) & i = 1.., K,\ j = 1, .., K, \\ \frac{dx^B_{i,j}}{dt} = R_3 x^A_{i,j} x^B_{i,j} + R_4 + D_2(\mu^B_{i,j} - x^B_{i,j}) & i = 1.., K,\ j = 1, .., K, \end{cases} \tag{6.3}$$

where: $x^A_{i,j}$ and $x^B_{i,j}$ are the concentrations of the two species in the cell (i, j); R_i, $i = 1, ..., 4$ are the parameters that define the reaction between the two species; D_1 and D_2 are the diffusion constants; $\mu^A_{i,j}$ and $\mu^B_{i,j}$ are the inputs for the (i, j) cell, that is

$$\mu^n_{i,j} = \frac{1}{|\nu_{i,j}|} \sum_{\nu \in \nu_{i,j}} x^n_\nu \qquad n \in \{A, B\}, \tag{6.4}$$

where $\nu_{i,j}$ is the set of indices of cells adjacent to (i, j). The spatio-temporal trace of the system is the function $\mathbf{x} = (x^A, x^B) : [0, T] \times L \to \mathbb{R}^{K \times K} \times \mathbb{R}^{K \times K}$ where each x^A and x^B are the projection on the first and second variable, respectively. In Fig. 6.5, we report the concentration of A for a number of time points, generated by the numerical integration of System 6.3; at time $t = 20$ and $t = 50$, the shape of the pattern is apparent and remains stable. We can see that some regions (in blue) have a low concentration of A surrounded by regions with a high concentration of A. We consider as spots of our pattern the regions with low concentration of A. The opposite happens for the value of B (high density regions surrounded by low density regions, not shown).

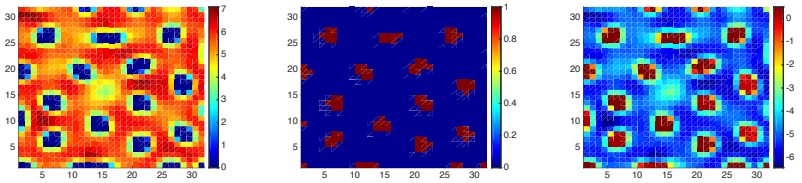
Analysis. The following shows how we can use the surround operator to characterise the behaviour of this system. In order to classify spots, one should identify the sub-regions of the grid that present a high (or low) concentration of a certain species, surrounded by a low (high, respectively) concentration of the same species. Formally, one can, e.g., capture the spots of the A species using the spatial formula

$$\phi_{\text{spot}} := (x^A \leq h)\mathcal{S}_{[d_1,d_2]}(x^A > h). \tag{6.5}$$

A trace $\mathbf{x}$ satisfies ϕ_{spot} at time t, in the location (i,j), $(\mathbf{x}, t, (i,j)) \vDash \phi_{\text{spot}}$, if and only if there is a subset $L' \subset L$, that contains (i,j), such that all elements have a distance less than d_2 from (i,j), and x^A, at time t, is less or equal to h. Furthermore, each element in the boundary of L' has a concentration of A, at time t, greater than h, and its distance from (i,j) is in the interval $[d_1, d_2]$. Note that the use of distance bounds in the surround operator allows one to constrain the size/ diameter of the spot to $[d_1, d_2]$. Recall that we are considering a spatio-temporal system, so this spatial property alone is not enough to describe the formation of a pattern over time; to identify the insurgence time of the pattern and whether it remains stable over time we have to combine the spatial property with temporal operators in this way:

$$\phi_{\text{pattern}} := \mathcal{F}_{[T_{\text{pattern}}, T_{\text{pattern}}+\delta]}\mathcal{G}_{[0,T_{\text{end}}]}(\phi_{\text{spot}}); \tag{6.6}$$

ϕ_{pattern} states that eventually at a time between T_{pattern} and $T_{\text{pattern}}+\delta$ the property surround becomes true and remains true for at least T_{end} time units. In Fig. 6.6 we show the validity of the property ϕ_{pattern} in each cell $(i,j) \in L$, for both the Boolean and the quantitative semantics. We recall that $(\mathbf{x}, \ell) \vDash \varphi$, if and only if $(\mathbf{x}, 0, \ell) \vDash \varphi$; for this reason the plots show the satisfaction at time $t = 0$. It is evident how well the procedure is able to identify which locations belong to the spots or not. If we make the distance constraint stricter, by reducing the width of the interval $[d_1, d_2]$, we are able to identify only the "centre" of the spot, as shown in Fig. 6.7. However, in this case we may fail to identify spots that have an irregular shape (i.e., that deviate too much from a circular shape).



(a) Snapshot (b) Boolean Semantics (c) Quantitative semantics

Figure 6.6: Validity of formula (6.6) with parameters $h = 0.5, T_{\text{pattern}} = 19, \delta = 1, T_{\text{end}} = 30, d_1 = 1$, and $d_2 = 6$. (a) Concentration of A at time t = 50; (b) Boolean semantics of property (6.6); the cells (locations) that satisfy the formula are in red, the others are in blue; (c) Quantitative semantics of the property (6.6); the value of the robustness is given by a colour map as specified in the legend on the right of the figure.

Formula ϕ_{pattern} describes the persistence of a spot in a specific location. To describe the global spatial pattern, that every location is part of a spot or has a nearby spot, the following SSTL formula can be used:

$$\phi_{\text{ST-pattern}} := \boxdot_{[0,d_{max}]} \diamondsuit_{[0,d_{spot}]} \phi_{\text{pattern}}, \tag{6.7}$$

where $\diamondsuit$ and $\boxdot$ are the everywhere and somewhere operators, d_{max} is chosen to cover all space, and d_{spot} measures the maximal distance between spots. Checking this formula in a random location of our space is enough to verify the presence of the pattern; this is enough because the first part of the formula, $\boxdot_{[0,d_{max}]}$, permits us to reach all the locations of the grid. This is an example of how we can describe global property also with a semantics that verifies properties in single locations. We verify the property (6.7) with $d_{max} = 45$ and $d_{spot} = 15$ (the other parameters as in Fig. 6.6), for a solution of the system (6.3) obtaining true for the Boolean semantics and 0.3 for the quantitative one. The low value of the quantitative semantics is due to the choice of the threshold h.

Changing the diffusion constants D_1 and D_2 affects the shape and size of the spots or disrupts them, as we can see in Figure 6.8. We evaluate the pattern formula (6.7) with parameters as in Fig. 6.6, for the patterns in Figure 6.8, where $D = [1.5, 23.6]$ and $D = [8.5, 40.7]$ and the other

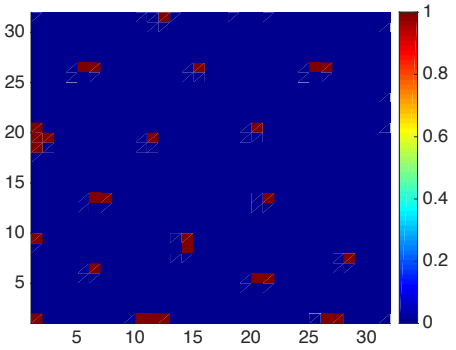


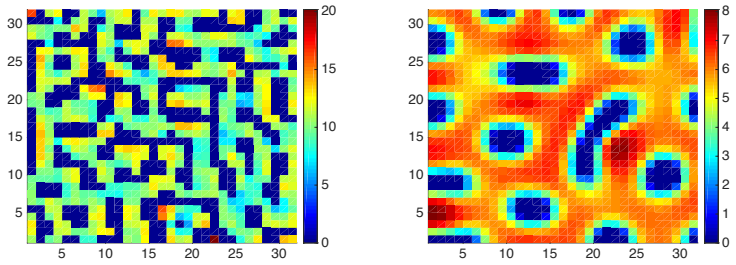
Figure 6.7: Boolean semantics of formula (6.6) with parameters $h = 0.5, T_{\text{pattern}} = 19, \delta = 1, T_{\text{end}} = 30, d_1 = 1$, and $d_2 = 4$; the cells (locations) that satisfy the formula are in red, the others are in blue.

parameters equal to the previous model, and it results false with a quantitative value equal to -0.05 for both. Formula (6.6), though, is still true in some locations. This is due to the irregularity of the spots (where, as in Fig. 6.8(a), some spots can have a shape similar to the model in Fig. 6.6(a)), or due to particular boundary effects on the border of the grid (where fractions of spots still remain, as in Fig. 6.8(a)).

A strength of spatio-temporal logics is the possibility to nest the temporal and spatial operators. We illustrate this in the following scenario. We set as initial conditions for the system (6.3) its stable state, i.e., the concentrations of A and B at time 50 (see Fig. 6.6(a)). We introduce a small perturbation, by changing a single value in a specific location in the centre of a spot. The idea is to study the effect of this perturbation, i.e., checking if it will disrupt the system or not. Specifically, we perturb the cell $(6, 6)$, by setting $x^A_{6,6}(0) = 10$. Dynamically, the perturbation is quickly absorbed and the system returns to the previous steady state. Formally, we consider the following property:

$$\phi_{\text{pert}} := (x^A \geq h_{\text{pert}}) \wedge (\phi_1 \mathcal{S}_{[d_m, d_M]} \phi_2); \tag{6.8}$$

$(\mathbf{x}, (i, j)) \vDash \phi_{\text{pert}}$, i.e., a trace $\mathbf{x}$ satisfies ϕ_{pert} in the location (i, j), if



(a) (b)

Figure 6.8: Snapshots at time $t = 50$ of x^A for the model (6.3) with parameters with parameters $K = 32, R_1 = 1, R_2 = -12, R_3 = -1, R_4 = 16$, and $D = [1.5, 23.6]$ in (a) and $D = [8.5, 40.7]$ in (b).

and only if $x^A_{i,j}(0) > h_{\text{pert}}$ (the location is perturbed) and if there is a subset $L' \subseteq L$ that contains (i, j) such that all its elements have a distance less than d_M from (i, j) and satisfy $\phi_1 = \mathcal{F}_{[0,T_p]}\mathcal{G}_{[0,T_d]}(x^A < h')$; ϕ_1 states that the perturbation of x^A is absorbed within T_p units of time, stabilising back to a value $x^A < h'$ for additional T_d time units. Furthermore, within distance $[d_m, d_M]$ from the original perturbation, where d_M is chosen such that we are within the spot of the non-perturbed system, $\phi_2 := \mathcal{G}_{[0,T]}(x^A < h')$ is satisfied; i.e., no relevant effect is observed, the value of x^A stably remains below h'. The meaning of ϕ_{pert} is that the induced perturbation remains confined inside the original spot. In Fig. 6.9, we report the evaluation of the quantitative semantics for ϕ_{pert}, zooming in on the 15×15 lower left corner of the original grid. As shown in the figure, the perturbed location $(6, 6)$ satisfies the property.

Model (6.3) has been coded in Matlab/Octave, and the monitoring has been performed by our Java implementation, jSSTL, described in Chapter 8. Regarding time performance, the verification of property ϕ_{pattern} took $1.04s$ (Boolean) and $69.39s$ (quantitative) for all locations and 100 time points, while property $\phi_{\text{ST-pattern}}$ took $1.81s$ and $70.06s$, and property ϕ_{pert} took $28, 19s$ and $55, 31s$, respectively. The compu-

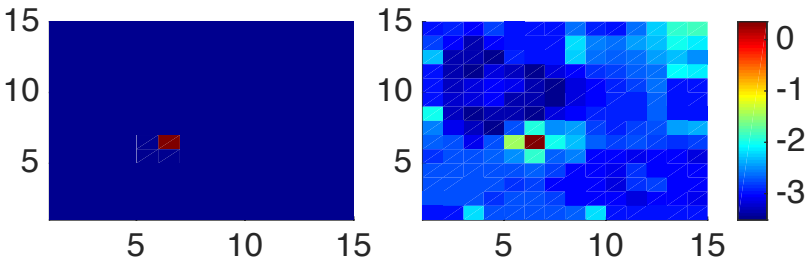


Figure 6.9: Boolean and quantitative semantics for formula 6.8 with parameters $h_{\text{pert}} = 10$, $w_m = 1$, $w_M = 2$, $T_p = 1$, $T_d = 10$, $h' = 3$, and $T = 20$.

tation of the distance matrix can be done just once because it remains always the same for a given system, this takes about $23s$. All the experiments were run on a Intel Core i5 2.6 GHz CPU, with 8GB 1600 MHz RAM.

Chapter 7

Statistical Analysis of Stochastic Spatio-Temporal Systems

In this chapter, we extend the methodology presented in Chapter 5 to SSTL, the spatio-temporal logic defined in the previous chapter. Furthermore, we extend, in a similar way, the *smoothed model checking* technique developed in (BMS16), and described in Chapter 4. The entire framework can then be used to analyse and design systems with stochastic spatio-temporal dynamics. In particular, we apply it to study a french-flag model of the Drosophila's Bicoid morphogen (work published in (BBM+15)).

7.1 Methodology

The main objective of this chapter, and of the entire thesis, is to be able to analyse and design complex systems with spatio-temporal dynamics. Complex systems are often very large-scale systems. As we explained in Chapter 5, an exhaustive parameter exploration is particularly expensive for these kind of models, one reason being the high cost of stochastic simulations. When the spatial dimension is added to the system and to

its dynamics, the analysis becomes much more complicated and computationally costly.

In this section, we report the methodologies that we use to perform system design/parameter synthesis and model checking in presence of parametric uncertainty of stochastic spatio-temporal systems.

We extend the approach described in Chapter 5 and the *smoothed model checking* (BMS16), described in Chapter 4, where only temporal formulae were considered, to work with SSTL formulae and to be able to consider models with spatio-temporal dynamics. The extension basically consists in substituting the STL and MITL logics with the SSTL logic and its monitoring procedures, showing that this is a consistent operation. The approach works then with the robustness value $\rho(\varphi, \mathbf{x}, t, \ell)$ and the satisfaction of a spatio-temporal formula ϕ, $p(\phi = \mathbf{true}|\mathcal{M}_\theta, \ell)$.

From the implementation point of view, we integrated jSSTL, as a Java library, in the U-Check tool, both described in Chapter 8.

Despite the fact that the extension does not pose many challenges from the theoretical point of view, it has strong implications from the perspective of applications. It permits us not only to analyse behaviours that we could not describe by a temporal logic, but also to avoid a blowup of the formula size (which would be the case in any attempt to syntactically turn a spatio-temporal property into a purely temporal one), exponentially in the nesting level of spatial operators, and hence an exponential increase in the complexity of the monitoring procedure.

Below, we report briefly the two techniques: robust parameter synthesis and smoothed model checking.

7.1.1 Robust Parameter Synthesis

Given a population model, M, depending on a set of parameters $\boldsymbol{\theta} \in \boldsymbol{K} \subseteq \mathbb{R}^d$, and a specification ϕ, the problem of robust parameter synthesis (or system design) is that of identifying the parameter combination $\boldsymbol{\theta}^*$ such that the system satisfies ϕ as *robustly* as possible. For a stochastic temporal model, the problem was translated in finding the model parameters that maximise the average robustness degree $E[R_\phi]$.

The problem for a spatio-temporal formula is to determine the parameter combination $\boldsymbol{\theta}^*$ such that the system satisfies a spatio-temporal formula ϕ as *robustly* as possible in a given location ℓ.We stress the fact that by verifying properties in a specific location, we can still specify global behaviours, by properly use the everywhere and the somewhere operators. The location where the property is verified can be considered as a sort of "initial position", the equivalent of $t_0 = 0$ for the time.

According to the quantitative semantics of SSTL, the robustness value $\rho(\varphi, \mathbf{x}, \ell)$ expresses the level of satisfaction of ϕ by a trajectory $\mathbf{x}$ in location ℓ. We recall that in this case, instead of having a quantitative satisfaction degree, we have a spatial quantitative satisfaction function that gives a value for each locations. As described in the previous chapter, we can interpret then the robustness of ϕ for stochastic spatio-temporal models as the functional $R_\phi : \mathcal{D} \to \mathbb{R}^m$, where $\mathcal{D}([0, \infty], \mathbb{D})$ is the trajectory space of our system with $\mathbb{D} = \mathbb{D}_1 \bigoplus \cdots \bigoplus \mathbb{D}_m$, $m = |L|$ the number of locations and $\mathbb{D}_i$ the state space of the temporal trajectories in location ℓ_i. Hence, the problem is rephrased as the identification of the model parameters that maximise the average robustness in location ℓ, $E(R_\phi[\ell])$, of a spatio-temporal formula ϕ.

We are therefore interested in maximising the expected quantitative score:

$$E(R_\phi[\ell]) = \int \rho(\phi, \mathbf{x}, \ell) p(\mathbf{x}, \ell) d\mathbf{x} \tag{7.1}$$

where $p(\mathbf{x}, \ell)$ is the probability density of trajectory $\mathbf{x}_\ell(t) := \mathbf{x}(t, \ell)$. For a specified location ℓ, the expectation $E(R_\phi[\ell])$ constitutes an objective function, of which we can obtain noisy estimates by generating samples from the trajectory space via stochastic simulation. So, the system design corresponds to finding the parameter configuration $\boldsymbol{\theta}^*$ that maximises the function $f : \boldsymbol{K} \to \mathbb{R}$ s.t. $f(\boldsymbol{\theta}) := E(R_\phi[\ell])[\boldsymbol{\theta}]$ for a fixed ℓ.

We employ then the Gaussian Process regression and the Gaussian Process Optimisation algorithms described in Chapter 4 to find the parameter configuration $\boldsymbol{\theta}^*$ that maximises $f(\boldsymbol{\theta})$. We briefly recall the technique. The algorithm is initialised with a random grid of points, for each of which $f(\boldsymbol{\theta})$ is approximated via statistical means. Hence, using these points as a training set, the objective function is approximated by a *Gaus-*

sian Process (GP) regression, i.e., the GP is used to make predictions regarding the value $f(\boldsymbol{\theta})$ at different parts of the search space. We calculate the GP posterior for a set of test points, that involves calculating an estimate of the expected robustness and its associated variance. The GP optimisation algorithm dictates then that the point that maximises the upper quantile of the GP posterior is added to the training set, after being evaluated for its associated robustness via simulations. This process is repeated for a number of iterations, and the training set is progressively updated with new potential maxima. For a smooth objective function, the algorithm is proved to converge to the global optimum in (SKKS12).

7.1.2 Smoothed Model Checking

Smoothed Model Checking, described in Chapter 4, is a technique to characterise the satisfaction probability of a time-bounded linear time property ϕ as a function of model parameters. Here, we extend this procedure to characterise the satisfaction probability of a SSTL formula.

Let $\mathcal{M}_{\boldsymbol{\theta}}$ be an uncertain CTMC model, i.e., a model whose transition rates depend smoothly on a set of parameters $\boldsymbol{\theta} \in \boldsymbol{K} \subseteq \mathbb{R}^d$. We define then the satisfaction probability function of a SSTL formula ϕ as the function $f : \boldsymbol{K} \times L \to [0, 1]$, where L is the set of locations, s.t.

$$f(\boldsymbol{\theta}, \ell) \equiv p(\phi = \mathbf{true}, \ell | \mathcal{M}_{\boldsymbol{\theta}}).$$

It corresponds to the probability that ϕ is true in location ℓ, given a model with parameter $\boldsymbol{\theta}$. We recall that we defined $P(\phi, \ell)$ in Chapter 6 as $P(\phi, \ell) = \mathbb{P}\{(\mathbf{x}, \ell) \in \mathcal{D}_\ell \mid (\mathbf{x}, \ell) \vDash \phi\}$, i.e., the probability that a trajectory in location ℓ satisfies ϕ.

In (BMS16) the authors prove that the satisfaction function of a MITL formula is a smooth function of the parameters if the transition rates of $\mathcal{M}_{\boldsymbol{\theta}}$ depends smoothly on the parameters $\boldsymbol{\theta}$ and polynomially on the state of the system. Note that $f(\boldsymbol{\theta}, \ell)$ can be seen as a product of smooth functions $f_\ell := \pi_\ell \circ f$, which shows that it is smooth.

The method then is similar to the one described in Chapter 4. Computing analytically this function is almost impossible. The technique exploits then the smoothness of the satisfaction function to emulate it using

the Gaussian Progress Regression, described in Chapter 4. Hence, fixed a certain location ℓ, $\forall \boldsymbol{\theta}^* \in \boldsymbol{K}$, it computes an estimation of $f(\boldsymbol{\theta}, \ell)$ and a confidence interval for such a prediction. As stated before, the strength of the method is that fewer samples are needed to obtain a good level of accuracy, compared to standard parameter synthesis techniques that use only SMC.

7.2 Case Study: The French Flag Model

In this section, we apply the described techniques to a reaction-diffusion model of segmentation in *Drosophila melanogaster* and the spatio-temporal pattern characterising it, known as the French Flag model. In particular, we want to study the effects of the morphogen Bicoid parameters on the satisfaction of the French Flag property.

We presented an overview about pattern formation and reaction-diffusion systems in Chapter 6, Section 6.5.2. One of the most widely studied models of morphogenesis is the establishment of spatial patterning (stripes) along the body of the fruit fly *Drosophila melanogaster*. Several morphogens are known in *Drosophila*; mostly, these are maternal proteins that are produced in a localised area of the embryo (in correspondence to a maternal deposit of messenger RNA), and then establish a concentration gradient during development, effectively providing cells within an embryo with a spatial reference. An important morphogen is the protein *Bicoid*.

Before describing the Bicoid model, it is worth remarking on a fundamental shift of perspective that has happened since Turing's pioneering work, the realisation of the importance of stochasticity in biology. Numerous lines of evidence indicate that biology at the single cell level is intrinsically stochastic. Stochasticity cannot be ignored when modelling early embryogenesis, when only a handful of cells are present. Morphogenetic reaction-diffusion models can therefore be modified to account for the intrinsic discreteness of biology at the microscopic level. Morphogenetic systems, and in particular the *Bicoid* system, have already been analysed from a simulation perspective in (WMR07) and from a statistical perspective in (DOS10). Here, we present a first analysis of

this system from the point of view of spatio-temporal logic, to analyse directly the system's behaviour at the level of the emergent properties of the trajectories.

7.2.1 The Bicoid Gradient Model

The *Bicoid* (Bcd) molecule was the first protein to be identified among the morphogens. In the *Drosophila* embryos, the Bcd protein is distributed along the Anterior-Posterior axis (A-P axis). The Bcd mRNA is translated at the anterior pole of the embryo, and the synthesised protein spreads through the A-P axis by *diffusion* accompanied by *decay*.

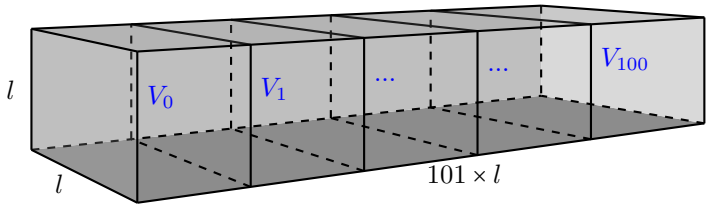


Figure 7.1: A schematisation of the *Drosophila* embryo volume. The volume is divided in 101 cubic subvolumes, $V_0, ..., V_{100}$, with side $l = 5\mu m$.

We will describe the dynamics of the Bcd protein by a stochastic *reaction-diffusion* system, as reported in (WMR07). Given a certain volume where the Bcd protein is distributed, we can divide it into a series of subvolumes or voxels that are small enough to be regarded as well mixed. Then, we can consider the *decay* reaction as a transition that happens inside the subvolumes and the *diffusion* as exchange of molecules between neighbouring voxels. In particular, we consider 101 homogeneous cubic subvolumes with side $l = 5\mu m$ that comprise the entire volume as in Fig. 7.1. The length of the side l and the number of subvolumes were chosen in light of those of actual Drosophila embryos, which are $500\mu m$ long. The first subvolume ($j = 0$), corresponds to the anterior pole of the embryo and it is the only subvolume where the Bcd protein is synthesised.

We can describe the set $\mathcal{R}$ of reactions governing the stochastic dy-

namics of Bcd as:

$$\begin{aligned}
\nu_p &: \quad \varnothing \to B_0 && \text{at rate } J, && && \text{(production)}\\
\nu_{deg_j} &: \quad B_j \to \varnothing && \text{at rate } w, && \text{for } j = 0,...100, && \text{(degradation)}\\
\nu_{dif_j^+} &: \quad B_j \to B_{j+1} && \text{at rate } \frac{D}{l^2}, && \text{for } j = 0,...99, && \text{(diffusion to the right).}\\
\nu_{dif_j^-} &: \quad B_j \to B_{j-1} && \text{at rate } \frac{D}{l^2}, && \text{for } j = 1,...100, && \text{(diffusion to the left).}
\end{aligned}$$

where B_j is a Bcd protein in the jth subvolume.

The state vector of the system is then $\mathbf{x}_B = (x_{B_0}, ..., x_{B_{100}})$ where x_{B_j} is the number of Bcd molecules in the jth subvolume. From the set $\mathcal{R}$ we can derive the infinitesimal generator matrix of the CTMC that formally represents the dynamics of the system. The CTMC can then be simulated with a standard algorithm, such as SSA or τ-leaping.

Note that, from the set of reactions $\mathcal{R}$, we can easily revert the discretisation process and obtain a semantics in terms of Reaction-Diffusion Rate Equation (RDRE). This is obtained by converting variables into concentrations, taking the length of voxels to zero, and interpreting each rate as a flow, both in the degradation and in the diffusion reactions. In this way, we can define the system

$$\frac{\partial u}{\partial t} = D\frac{\partial^2 u}{\partial y} - wu, \tag{7.2}$$

where $u(y,t)$ is the concentration of Bcd at time t in position y, measured in μm, $y \in [0, 500]$, giving the boundary conditions $\frac{\partial u}{\partial y}\big|_{y=0} = -\frac{J}{\Delta}$ and $\frac{\partial u}{\partial y}\big|_{y=500} = 0$, where $\Delta = l^3$.

7.2.2 Segmentation and the French Flag property

The spatial distribution of the Bicoid protein has a crucial role in the formation of the horizontal segmentation in the development of the Drosophila's embryo. One of the most important interpretations of this distribution is given by the *French Flag model* (Wol68), and more generally by the theory of *gap genes* (Jae10; WTA15). The body of the fruit fly *Drosophila melanogaster*, as in most arthropods, exhibits a particular type of spatial patterning called *segmentation*, whereby the main body is composed of

several segments. Gap genes were discovered and named following mutagenetic experiments, whereby biologists observed that deletion of certain genes resulted in the omission of a segment in the fly's body, as if the mutant organism had a gap. This observation implies that gap genes must be expressed in a precisely spatially co-ordinated manner, i.e., the biochemistry of the fruit fly must possess a way of measuring distances.

The French Flag model is a simplified model of gap gene regulation in early embryogenesis involving only four genes, the Bicoid morphogen protein and three target genes. The underlying assumption is that the spatial distribution of Bicoid protein, which as we have seen tends to decrease along the A-P axis (see Fig. 7.2), provides the ruler with which the *Drosophila* embryo measures distances. Gap genes are activated in a concentration dependent manner by Bicoid, so that a set of genes are activated at the high concentrations near the anterior part of the embryo (the blue in the French Flag), a different set of genes is activated in the central part (the white) and a third set is activated a low concentrations near the posterior end (red). This model has survived with some modifications (JMA09) until this day, its beauty providing a paradigm for pattern development in many areas of biology. From our point of view, this model is particularly interesting because it refocuses attention from local intensive quantities (local concentrations) towards the importance of a global emergent property of the system (the establishment of a gradient), which is ideally suited for reasoning upon in terms of spatio-temporal logics. We will see now how to describe the French Flag pattern using a spatio-temporal logic.

To describe the French Flag pattern we have first to define the trajectories that we want to characterise and its related graph. Let consider a trace (a simulation) $(\mathbf{x}_B(t))_{t\in[0,T]} = (x_{B_0}(t), ..., x_{B_{100}}(t))_{t\in[0,T]}$ of the Bicoid model described in the previous section, where $[0, T]$ is the time domain, with $T > 0$. We can transform the temporal trace in a spatio-temporal trajectory defining $x_B : L \times [0, T] \to \mathbb{R}$ s.t. $x_B(V_i, t) := x_{B_i}(t)$, where $L = \{V_0, ..., V_{100}\}$ is the set of locations. The graph $G = (L, E, w)$ of the system is a one-dimensional graph where each V_i is connected only to V_{i-1} and V_{i+1}, with $w(V_i, V_{i+1}) = 1$, i.e., all the edges have weight equal

to 1. The weight between two arbitrary locations is given by the weight of the shortest path connecting them.

We can now use the logic to specify the French Flag model. As we described in Section 7.2, this pattern is used to represent the effect of a morphogen in the expression of different genes, i.e., to represent the correlation between the concentration of the morphogen and the activation or repression of other genes. In particular, the spatial distribution of the morphogen, at the steady state, is divided in three regions: a blue, a white and a red region, as shown in Fig.7.2 (left), that activate different target genes.

We can describe this behaviour with the property

$$\psi_{flag} := \phi_{blue} \wedge \phi_{white} \wedge \phi_{red} \tag{7.3}$$

$$\begin{array}{lcl} \phi_{blue} & := & \boxdot_{[0,w_{blue}]}(x_B > K_{blue} - h_{bw}) \\ \phi_{white} & := & \boxdot_{[w_{blue},w_{white}]}((x_B < K_{blue} + h_{bw}) \wedge (x_B > K_{white} - h_{wr})) \\ \phi_{red} & := & \boxdot_{[w_{white},w_{max}}(x_B < K_{white} + h_{wr}) \end{array} \tag{7.4}$$

The verification of the formula is done in the location V_0. $(x, V_0) \vDash \psi_{flag}$ iff it satisfies each subformulae $\phi_{blue}, \phi_{white}, \phi_{red}$; $(x, V_0) \vDash \phi_{blue}$ iff, in all the locations V_i s.t. $w(V_0, V_i) \leq w_{blue}$, the number of Bicoid molecules is higher than $K_{blue} - h_{bw}$, i.e $x_B > K_{blue} - h_{bw}$. In a similar way we can describe ϕ_{white} and ϕ_{red}. The meaning of the property is that the spatial distribution of the Bicoid protein is divided in three regions, the blue, where the $x_B > K_{blue} - h_{bw}$, the white, where $K_{blue} + h_{bw} > x_B > K_{white} - h_{wr}$, and the red, where $x_B < K_{white} + h_{wr}$. Note that h_{bw} and h_{wr} parameters have the role to relax the thresholds that define different regions, to properly deal with noise in Bcd expression, we will discuss this point more in detail in the Section 7.2.3.

At steady state, the concentration of the Bicoid protein is exponentially distributed along the anterior-posterior (A-P) axis, with higher concentrations towards the anterior. We can identify the insurgence time of this pattern, and if it remains stable, combining the spatial property with temporal operators as follows:

$$\psi_{stableflag} \quad := \quad \mathcal{F}_{[T_{flag}, T_{flag}+\delta]}(\mathcal{G}_{[0,T_{end}]}\psi_{flag}) \tag{7.5}$$

$\psi_{stableflag}$ means that eventually, in a time between T_{flag} and $T_{flag}+\delta$, the property ψ_{flag} remains true for at least T_{end} time units.

7.2.3 Results

In this subsection, we perform a series of experiments to explore the sensitivity and robustness of the French Flag property w.r.t. changes in the rates of production J and degradation w, and the diffusion rate parameter D. The size of the cubic subvolumes is known, that is $l = 5\mu m$, as it is one of the main modelling assumptions.

Experimental Data Following (PKT09; WMR07), we chose as parameters of the $\psi_{stableflag}$ property (7.5), specified in Section 7.2.2, T_{flag} = 3950, δ = 10, T_{end} = 1000, w_{blue} = 35.5, w_{while} = 67.5 and w_{max} = 101. The w_{blue} and w_{while} parameters mean that the blue area involves the subvolumes between V_0 and V_{35}, the white area extends from volume V_{36} to V_{67}, and finally the red one from V_{68} to V_{100}; the time is in terms of seconds.

In order to fix the thresholds parameters K_{blue}, K_{white} and h_{bw},h_{wr} we use the Bicoid fluorescence concentration at cycle 13 (where the gradient is considered to be in the steady state) downloaded from the FlyEx database (fly). The choice of the data follows the analysis done in (WMR07). To the best of our knowledge, all the quantifications of the Bicoid protein in the Drosophila embryo refers to the measurements of fluorescence concentrations, rather than direct observations of the Bicoid molecular population. From (WMR07), we define the fluorescence concentration $I = m \times x_B$, where m is a scaling factor that denotes the fluorescence-to-molecule ratio. Our approach is to rescale the thresholds reported in terms of fluorescence concentrations with the m factor.

The data has been given originally in the form of two-dimensional coordinates paired, the A-P and D-V coordinate, from the central 10% strip. As in (WMR07), we choose the embryos where the variation inside each spatial subregions is low, in particular in these embryos the inverse of

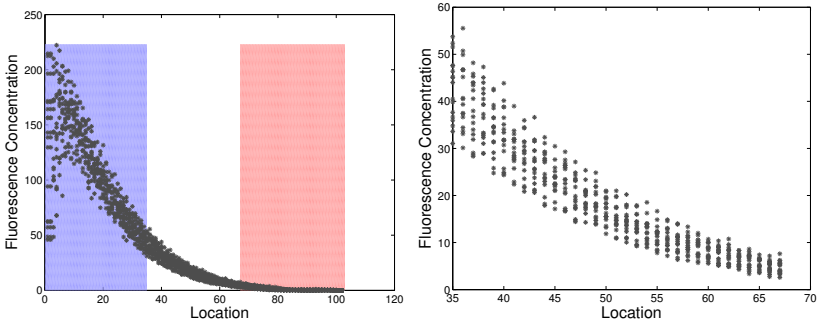


Figure 7.2: Left: Fluorescence concentrations of the Bicoid protein for 17 embryos during the cycle 13. Right: The same concentrations in the area between locations 35 and 67, which define the white area in the French flag property.

the spatial exponential coefficient varied by less that 1%. We have transformed the data to obtain a single concentration value for each of the 101 discretised locations. Fig. 7.2 depicts the result. On the left-side figure, we see how the different locations lie within the areas prescribed by the French Flag property. Although the shape of the data is apparently negative exponential, there is a considerable amount of noise, which has to be taken into consideration in terms of the French Flag property. We therefore define the thresholds in the form regions, rather than strict values. On the right-side of Fig.7.2, we see a magnified version of the figure, where only the white area is depicted. The majority of the concentrations recorded for volumes from V_{36} to V_{67} are between 60 and 2. In the same way, we can empirically derive zones of desired concentration levels for the blue and read areas. Therefore we have $K_{blue} = 45/m$, $h_{bw} = 15/m$, $K_{white} = 6/m$, and $h_{wr} = 4/m$.

Optimisation of Expected Robustness We now explore how the model parameters (including the scaling factor m) can be tuned to increase the robustness of the French Flag pattern.

We applying the GP optimisation algorithm discussed in Chapter 5,

for a four-dimensional space that involves the parameters: $w \in [0.001, 0.01]$, $J \in [10, 400]$, $D \in [1, 40]$, and $m \in [0.01, 1]$. The parameter ranges have been selected so that the resulting space is a superset of the explored space in (WMR07). Regarding the fluorescence-to-molecule ratio in particular, we note that the extremes considered in (WMR07) were0.07 and 0.7.

For each evaluation of the expected robustness, the system has been simulated up to time $t = 4000$ sec, which is when the steady-state is approached according to (WMR07). The robustness expectation has been approximated statistically using 12 simulation runs for each parameter set. The algorithm has been initialised by 80 evaluations of the objective function at random points; a number of 282 evaluations were performed at points selected by the optimisation process, until convergence was detected. Convergence has been determined when no significant improvement of the expected robustness has been observed for 200 iterations. An improvement is considered significant, if it is more than 1% increase over the previously recorded maximum robustness.

At the end, a total of 362 function evaluations have been performed, which is arguably a small number of samples to explore a 4-dimensional space. The execution times have been 85 minutes for the initial 80 evaluations, and 263 minutes for the actual optimisation process. Stochastic simulations have been performed in parallel using 12 threads. The experiments have been performed on an Intel® Xeon® CPU E5-2680 v3 2.50GHz. The majority of the computational effort was spent in simulation, despite the fact that only 12 trajectories have been generated for each parameter set considered. Therefore the idea of reducing the number of samples by exploiting the smoothness of the objective function has been a sensible practice.

The values returned by the optimisation process have been: $w^* = 0.0038$, $J^* = 390$, $D^* = 32.5$, and $m^* = 0.048$. The robustness of the optimum returned has been 2.99, implying that the property is robustly satisfied for the given solution. In Fig. 7.3, we present a sample trajectory for the given parameter configuration, and the average of 40 random trajectories, along with the associated 99.8% confidence bounds. The sample

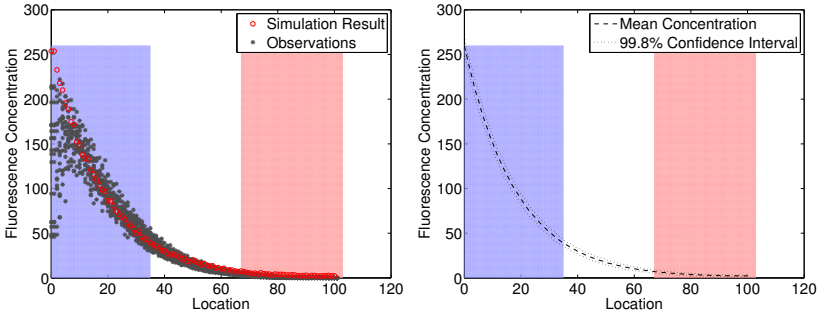


Figure 7.3: Left: Sample trajectory for the parameter configuration that maximises the robustness of the French Flag property. Right: Average of 40 random trajectories; the dotted lines indicate the 99.8% confidence interval.

trajectory is plotted against the experimental data that were used to adjust the threshold parameters of the French Flag property. We see that the optimised model has a behaviour very similar to the one observed in real-world experiments. However, it appears that the simulation results are much less noisy, when compared to the actual observations. This finding is in agreement with the result of (WMR07), where it was argued that the intrinsic noise as modelled by the stochastic dynamics of the master equation is not sufficient to explain the variability in the data, i.e., the noise in the fluorescence measurement as a crucial role that has to be taken into account.

Parameter Exploration with Smoothed Model Checking We perform, then, a more thorough exploration of the parameter space. Our objective is to discover dependencies among the parameters, considering the satisfaction probability of the French Flag property. On that respect, the fluorescence-to-molecule ratio m is not significant, as this will have an obvious effect on the thresholds for the property. We fix the fluorescence-to-molecule ratio m to 0.048, which is the optimal value reported by the optimisation algorithm in the previous section. The rest of the model parameters, $w \in [0.001, 0.01]$, $J \in [10, 400]$, and $D \in [1, 40]$, are explored via

the smoothed model checking approach.

During the initialisation step of the algorithm, we have performed 216 evaluations of the satisfaction function of (7.3), for a regularly distributed set of values. As in the previous section, the satisfaction probability is approximated by statistical model checking using 12 simulation runs for each parameter configuration, where the system is simulated up to time $t = 4000$ sec.

The duration of this initial statistical model checking process has been nearly 170 minutes, on an Intel® Xeon® CPU E5-2680 v3 2.50GHz, using 12 threads in parallel. The hyperparameter optimisation that is required to tune the GP probit regression model subsequently required only 20 seconds, which is a trivial price to pay compared to the massive simulation cost. The final GP probit regression for a grid of 4096 points required only 1.2 seconds. Most importantly, it is only this last cost that we are required to pay to produce any further estimations of the satisfaction function.

Fig.7.4 depicts the satisfaction function for the French Flag property for parameters $\theta = \{w, J, D\}$, as this has been approximated by smoothed model checking. Each of the depicted subfigures shows the satisfaction probability as function of the production rate J and the diffusion parameter D, for a different value of the degradation rate w. Regarding the confidence of the estimated probabilities, we report that the 73.6% of the values are associated with 95% confidence intervals of width less than 0.2.

As a general remark, it appears that the manifestation of the gradient pattern, as this is captured by the French Flag property, is associated with a fine balance among the model parameters. There is a small area in the parameter space for which the property is satisfied with high probability. As we increase the decay parameter w however, we observe two behaviours regarding this area: its size is being increased, and its location is being shifted to the right. This implies that w is positively correlated with the production rate J. In other words, a particular ratio between protein production and decay is required for the formation of the particular pattern. At the same time, increasing the decay rate means that the

formula may be satisfied for a wider range of the diffusion parameter.

It also appears that there is a negative correlation between the production rate J and the diffusion parameter D. This behaviour is present for the entire range of w examined, but it tends to become more obvious as w is increased. It is reasonable to conclude that a simultaneous increase of J and D would destroy the exponential shape of the Bicoid distribution across space. This observation suggests that the property is more likely to be satisfied if all the rate constants in the system have high values. This reflects in a less erratic stochastic behaviour.

We remark that we chose a relative simple model and property, putting more effort in the accuracy of the specification and in the methodology. This has been particularly effective from a computational point of view, since stochastic simulation of the spatio-temporal model in question is very expensive; hence, standard parameter exploration techniques are not feasible, as a large number of samples is needed to obtain valid results.

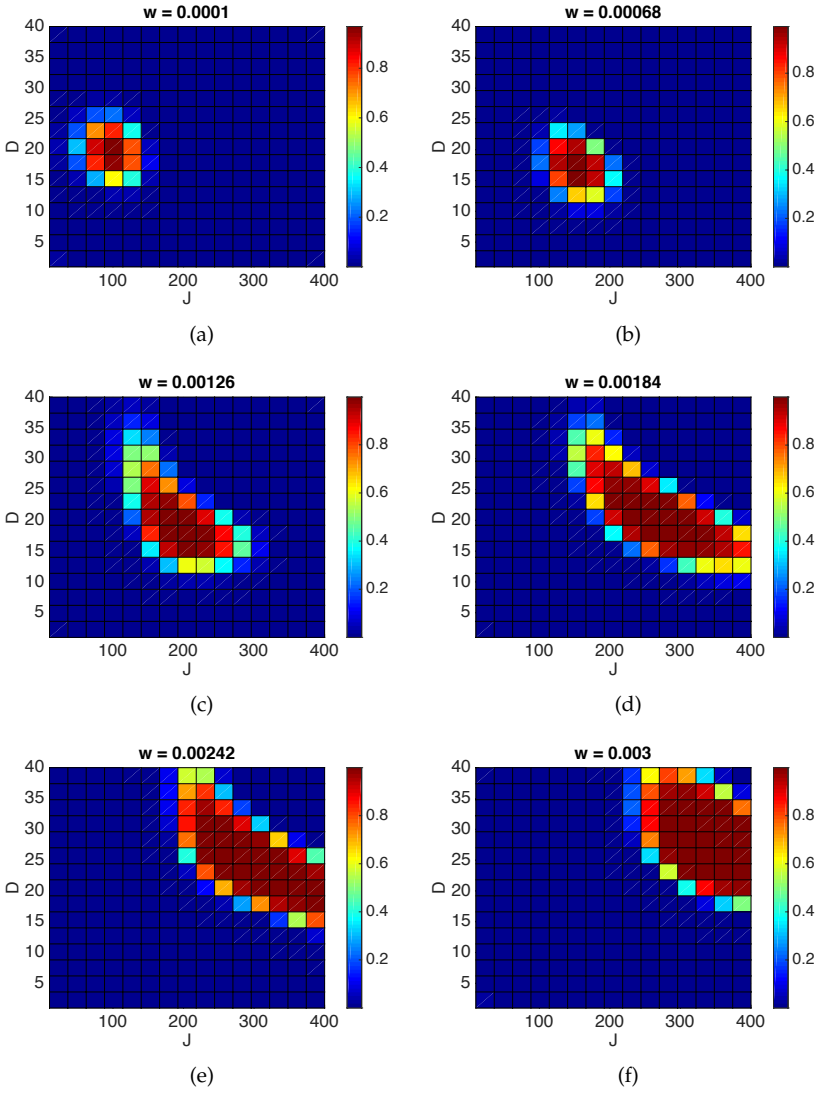


Figure 7.4: Emulated satisfaction probability of the French Flag property as function of $\theta = \{w, J, D\}$. Each subfigure has the w parameter fixed.

Chapter 8

Tool Support

In the first section of this chapter, we present jSSTL, a Java tool for the specification and verification of SSTL properties. In particular, we present a Java library and an ECLIPSE plugin of jSSTL. The tool has been implemented in collaboration with Michele Loreti and Luca Bortolussi and is still in development. In the second section, we briefly introduce U-check (BMS15), a Java tool for analysing stochastic models with uncertainty in the parameter space.

8.1 jSSTL

To support qualitative and quantitative monitoring of SSTL properties, a prototype tool has been developed. This tool, developed in Java, consists of a Java library (jSSTL API) and a front-end, integrated in ECLIPSE. Both the library and the ECLIPSE plugin can be downloaded from `http://quanticol.sourceforge.net/`. The source code is available at `https://bitbucket.org/LauraNenzi/jsstl`. The library can be used to integrate jSSTL within other applications and tools, whereas the ECLIPSE plugin provides a user friendly interface to the tool. Furthermore, the modular approach of the implementation allows to develop different front-end for jSSTL. We report a scheme of the implementation in Table 8.1.

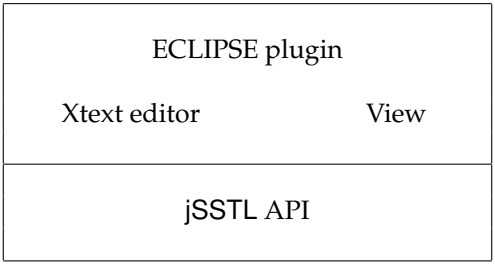


Table 8.1: The diagram of the implementation

We describe now in detail the library and the plugin.

8.1.1 jSSTL Java library

Our library has been designed in a modular way, which allows us to change/improve specific classes in the library without changing the overall behaviour. It consists of three main packages: util, core, and io.

Package util handles the temporal signals of the logic. Its BooleanSignal class defines the temporal Boolean signals of the logic. They are represented as a sequence of time intervals with value true or false. The class BooleanSignalTransducer implements a method to compute the interval covering of a set of Boolean signals, a method to convert piecewise constant signals into Boolean signals and the methods to perform operations between temporal signals: and, or, not, until, eventually, always. The class QuantitativeSignal defines the temporal quantitative signals of the logic, represented as piecewise constant signals. The class QuantitativeSignalTransducer implements the same methods as the Boolean case, for the operations between quantitative signals.

Package core is divided in three subpackages: space, monitor and formula. In the space subpackage, the class SpaceModel is an interface that represents a generic space model. It is parametrised with respect to the type of data that represents the nodes and the edges. It provides

three methods to obtain informations about nodes, edges and the external border of the model. The GraphModel class implements the interface as a space model based on a graph, and contains the methods to design, work, and have informations about the graph. It exploits the JGraphT[1] package to compute the matrix distance of the graph.

The monitor package handles the spatial Boolean and quantitative signals of the logic. They are treated as an HashMap that associates a temporal (Boolean or quantitative) signal to each location of the graph. The classes SpatialBooleanSignalTransducer and SpatialQuantitativeSignalTransducer contains all the methods to make operations between spatio-temporal signals: surround, somewhere, everywhere, and, or, not , until, always, eventually. The method surround corresponds to the implementations of the monitoring algorithms presented in Chapter 6.

The third subpackage, formula, provides the classes used to represent SSTL formulae. These classes mimic the *abstract syntax tree* of formulas. The base interface is the class formula, it contains the methods booleanCheck and quantitativeCheck that compute the spatial Boolean and quantitative signals, taking as input the GraphModel and the Signal. The Signal class contains the method to translate the data, i.e., the primary signals, in a spatial Boolean or quantitative signal. Each operator has a specific class formula parametrised over one or two sub-formulae (depending on whether the operator is unary or binary) and an interval for some operators. The methods in each subformula refer to the methods in the SignalTransducer class of the specific operator. For example, the SurroundFormula class has the method booleanCheck that returns the monitoring evaluation SpatialBooleanSignalTransducer.surround of the signals of the two subformulae for the given (spatial) interval. Hence, the Monitoring algorithm is implemented following the *visitor pattern*. It is performed via a visit of a formula that implements a bottom-up evaluation. It is important to remark that the use of this pattern simplifies the integration of possible alternative monitoring algorithms.

Package io provides a set of classes that can be used to read graph mod-

[1] `http://jgrapht.org`

els and input signals from an input stream and to write monitoring results to an output stream. Currently, CSV and tabular based ascii files are supported for both input and output of signals. Specific interfaces are also provided to simplify the integration of new specific input/output data formats.

8.1.2 ECLIPSE Plugin

The ECLIPSE plugin uses the Java library, described in the previous section and provides an editor based on Xtext[2] for supporting the definition of SSTL *scripts* and a *view* to visualise the results of the analyses.

In the **jSSTL editor** it is possible to define a script that contains the list of properties that we want to analyse using jSSTL. The syntax of the script of our language is reported in Figure 8.1. Besides the list of formulae, each script contains the list of the variables considered in the model, a set of constants, and a list of parameters that may occur in the formula. The parameters can be declared as belonging to an interval. When the monitoring procedure is performed, the user can select a specific value for each parameter in the corresponding interval. Standard expression can be used to define both constant and parameter intervals. In the script, each formula is associated with a name, that is used to select the specific property during the monitoring procedure.

The **jSSTL view** provides three different panels to visualise the spatial models, the relevant data declared in a script and to plot the system's trajectories and the Boolean and quantitative satisfaction signals. In Appendix B, we describe in detail the jSSTL view with a running example.

8.2 U-check

U-check is a Java toolbox that implements a number of methodologies combining formal analysis techniques of stochastic systems with machine learning methods for optimisation and regression problems. It can be downloaded at `http://homepages.inf.ed.ac.uk/dmilios/ucheck`.

[2] `https://eclipse.org/Xtext/`

script	::=	element*
element	::=	variableDec | constDec | parameterDec | formulaDec
variableDec	::=	**variable** *name*
constDec	::=	**const** *name* = expr
parameterDec	::=	**parameter** *name* **in** interval
interval	::=	[expr, expr]
expr	::=	baseExpr | expr + expr | expr × expr | ⋯
baseExpr	::=	int | float | literalExpr
formulaDec	::=	**formula** *name* = formula
formula	::=	formula**&**formula | formula|formula
	|	formula**U**interval formula | **G**interval formula
	|	**F**interval formula | formula**S**interval formula
	|	<<>> interval formula | [[]]interval formula
	|	**!**formula | relExpr
relExpr	::=	expr<expr | expr≤expr | expr>expr | expr≥expr
	|	expr==expr | ! =expr

Figure 8.1: jSSTL formula syntax.

We briefly describe the main characteristics of the tool, for more details we refer to the tool paper (BMS15).

U-check allows to perform a number of formal analysis of stochastic models. It is possible to load models written for PRISM (KNP11) and Bio-PEPA (CH09) for stochastic system, and SimHya for hybrid systems (PBXB08). As logic specification, it uses the MITL (AFH96) logic and the STL logic.

In particular, it permits three type of analysis:

- **Parameter estimation from qualitative observations**, a methodology to evaluate the parameter values that better explain the behaviour of a set of observations. Observations are assumed to be truth value of MITL formulae (AFH96) over few independent experiments. Parameters are then estimated by maximising the likelihood of the parameter configuration that better explains the dataset, using Bayesan MC to evaluate it. More details about this procedure can be found in (BS13; BS15).

- **System Design via Robustness maximisation**, using the methodology presented in Chapter 5.

- **Smoothed Model Checking**, a model checking for systems with uncertainty in the parameter space, briefly described in Chapter 4.

All these techniques have in common the fact that they exploit the GP regression and the GP-UCB algorithms, described in Chapter 4, to emulate and/or optimise an unknown function with possibly noisy observations.

Chapter 9

Conclusions

9.1 Concluding Remarks

In this thesis, we presented a framework for the formal analysis of stochastic and deterministic complex systems with a spatio-temporal dynamics.

We defined Signal Spatio-Temporal Logic, a spatio-temporal extension of STL (DM10), in which space is a finite metric structure represented by an undirected weighted graph. SSTL is interpreted over spatio-temporal signals with continuous time and discrete space. It permits the specification and verification of spatio-temporal property of systems embedded into a discrete space. It has the same operators as STL, plus two spatial operators: the bounded somewhere operator and the bounded surround operator. In SSTL, spatial and temporal operators can be arbitrarily nested. We provided the logic with a Boolean and a quantitative semantics in the style of STL (DM10), and defined novel monitoring algorithms to evaluate such semantics on spatio-temporal trajectories. The monitoring of the surround operator requires a different algorithm from those developed for timed modalities, as space is bi-directional, thus it makes sense to observe both *reaching* and *being reached*; classical path-based model checking does not coincide with spatial model checking also because loops in space are not relevant in the definition of *surrounded* operators. We remark that the logic is not tied to a specific class of pro-

cesses. We can, in principle, handle any system for which we can generate sample execution trajectories. In fact, we could even use traces generated from the execution of an actual system rather than through simulation of a model.

We extended the robustness degree of STL formulae in a probabilistic setting. We formally defined a robustness distribution of a formula. The average robustness degree of this distribution, $\mathbb{E}(R_\phi)$, gives an important measure of how strongly a formula is satisfied. This means that higher is the value, the more robust is the satisfaction of the formula. In particular, we showed that it provides valuable information that is not captured by the satisfaction probability alone. We equipped then also SSTL with this stochastic semantics.

We showed the logic at work on different case studies as a Turing reaction-diffusion system and an epidemic scenario of the spreading of cholera. We chose to study biological scenarios because they are very paradigmatic in the area of complex systems. Our methods, however, can be directly applied to socio-technological and Cyber-Phisical scenarios. The Turing system models a process of morphogenesis (Tur52b) in which spots are formed over time. We showed how to use the logic to characterise the formation of spots. Then, we demonstrated that the logic can be used also to specify global behaviours as the entire pattern of the system. Furthermore, we displayed how we can nest spatial and temporal operators to describe the absorption of a perturbation. In the epidemic scenario of the spreading of cholera among neighbouring communities, we showed how our logic is suitable to describe the epidemic propagation along a river, considering both the deterministic and the stochastic dynamics. In this specific example, we worked only with few locations/nodes, but the same type of analysis can easily be applied to more complex spatial structures, even with thousands of nodes. The problem in the analysis of models with more complex spatial structures is not in the verification techniques but in the simulation of the spatio-temporal model, specially for stochastic systems. Furthermore, the same logics can be applied to more general notions of weighted graphs, not necessarily representing a physical space. For instance, the graph may

represent sensors connected in a network, with weights representing the energy cost of transmission.

The logic has been implemented in a Java toolbox and is available at `http://quanticol.sourceforge.net/` as a Java library or as an ECLIPSE plugin.

We exploited then the logic for system design or robust parameter synthesis and formal analysis under parametric uncertainty of spatio-temporal properties. The framework combines statistical machine learning techniques based on Gaussian processes with the algorithm for monitoring SSTL properties and its average robustness. The optimisation is carried out using state-of-the-art optimisation algorithms coming from active learning, which emulate the true function from just few samples, and perform very well in a simulation based scenario. We also considered the problem of learning the most effective parameters of a given formula maximising the robustness score.

As case study, we analysed the occurrence of the French Flag pattern in the Bicoid gradient, during the development of Drosophila embryo. Studying how this property depends on the parameters of the model is challenging due to the very high computational cost of simulating a spatio-temporal model, and has only been possible by adopting recent efficient verification techniques that employ machine learning methodologies (BMS15). The combination of these new techniques with SSTL permits exploring behaviours that are extremely difficult to express (and monitor) with standard temporal logics, where each individual location would need to be accounted for.

From the implementation point of view, the experiments for the temporal systems have been performed in `MATLAB` using a Java library for the modelling and simulation part, instead, the experiments for the spatio-temporal systems have been realised in Java integrating jSSTL whitin the U-check toolbox.

This is a step towards the ambitious goal of finding suitable procedures to specify, analyse and design emergent behaviours (described as temporal logic formulae) from models and from experimental data. Many problems need still to be faced to achieve this goal, such as how

to choose or learn the structure of the formula, how to avoid overfitting in the design procedure, how to deal with the curse of dimensionality afflicting GP-UCB and other optimisation algorithms.

9.2 Future Works

The work presented in this thesis can be extended in several directions.

We plan to perform a more thorough investigation of the expressivity of SSTL, applying it on further case studies and compare its expressiveness with that of other spatio-temporal logics such as SpaTeL (GBB14; HJK+15).

We would like to consider also other representations of space like more general quasi-discrete metric spatial structures and continuous metric spaces. In the first case, we can exploit the topological notion of closure spaces (CLLM14) and extend it to the metric case. Note that the current monitoring algorithms work already for more general spatial structures, like finite directed weighted graphs with a quasi-metric, but we plan to provide a more precise characterisation of the class of discrete spatial structures to which they can be applied. The case of continuous space covers traces of model like spatio-temporal point processes or PDE systems. The syntax and the semantics are relatively easily extended to continuous spaces using topological paths, the main challenge being the design of efficient monitoring algorithms. One direction is to consider the discretisation of continuous spaces with *Finite Difference Methods* (FDM) (Olv14), i.e., regular grids, or *Finite Element Methods* (FEM) (Olv14), i.e., "triangular" meshes, analysing the error produced by the discretisation. Another interesting spatial extension is to consider directionality, i.e., to add a set of possible direction to the spatial operators. This will permit to specify properties as "I will find a bike at a distance no more the 100 meters in the north-east direction". A possibility in this direction, is to label edges of the space graph and add a regular expression on these label to the spatial operators. An example can be $\diamondsuit^{C}_{[0,d]}\,\phi$, where the regular expression is $C = [north, east]^{+}$, i.e., an infinite combination of north and east. A location satisfies this property

if it satisfies ϕ, the distance constraints and it can be reached from the current location moving only in the north or in the east direction.

We are also analysing different spatial properties to see if we need to design other spatial operators that can catch new spatial behaviours. We are considering a new more global definition of the $\phi_1 \mathcal{S}_{[d_1,d_2]} \phi_2$ surround operator where the distance does not depend on the particular location where the property is verified but is related to the diameter of the region that satisfies ϕ_1 and is surrounded by a ϕ_2-region. We are considering also to use some clustering algorithms, e.g., spectral clustering, to search "ϕ clusters" in the graph.

Moreover, we want to give a definition of the semantics for the spatial operators via paths, so that it can be used to design distributed algorithms instead of the current sequential one. This will greatly improve the performance of the monitoring procedure in practical scenarios. To deal with the possible uncertainty in the spatio-temporal trajectories or in the formula parameters we plan to consider also imprecision in SSTL. Hence, this line of future work can be a starting point in the design of online monitoring algorithms of SSTL where we have just partial information about the signals.

The present work uses advanced machine learning concepts to address the system design problem in formal modelling; this is a relatively new line of work (BS15; BS14; BMS16; KJMA+14; BBS14a; LS14; BBS+14b; HJK+15) which opens significant new avenues for further research. From the practical point of view, more extensive testing and an efficient and robust implementation (exploiting some of the possible parallelisms, e.g., in SMC) will be important for the tool to be adopted. From the theoretical perspective, we plan to use multi-objective optimisation to find good parametrisation for conflicting objectives. Another interesting direction is to combine the design problem with the inference problem, which has recently been addressed for a number of continuous time stochastic systems (OMS13); this would open the possibility of addressing the control problem for such systems, simultaneously inferring the state of the system and designing the optimal input to lead it to a desired state.

Appendix A

Correctness of The Monitoring Algorithms for the Surround Operator

In this appendix, we prove the correctness of the Boolean and the quantitative monitoring algorithms for the surround operator.

A.1 Correctness of the Boolean Monitoring Algorithm for the Surround Operator

In this section we prove the correctness of Algorithm 2, Chapter 6, that, for simplicity, we report again below. Let's call the algorithm BoolSurround.

Theorem A.1 *Given a graph $G = (L, w, E)$, two properties ϕ_1 and ϕ_2, a trace $\mathbf{x}$ and a location ℓ, let $s_{\psi,\ell}$=BoolSurround$(G, \mathbf{x}, \phi_1, \phi_2, \ell)$ and $\mathcal{I}_{s_{\psi,\ell}}$ be the minimal interval covering consistent with $\{s_{\phi_1,\ell'}, s_{\phi_2,\ell'}\}_{\ell' \in L^{\ell}_{[0,w_2]}}$, then, for all $I_i \in \mathcal{I}_{s_{\psi,\ell}}$*

$$s_{\psi,\ell}(I_i) = 1 \iff (\mathbf{x}, t, \ell) \vDash \phi_1 \mathcal{S}_{[d_1,d_2]} \phi_2 \quad \forall t \in I_i$$

Proof: First we note that $s_{\psi,\ell}(I_i) = 1 \iff \ell \in V_{I_i}$, where V_{I_i} is the set V at the end of the iteration of the I_i interval. Then, it is enough to prove

Algorithm 4 Boolean monitoring for the surround operator

input $\ell, \psi = \varphi_1 \mathcal{S}_{[d_1,d_2]} \varphi_2$
$\forall \ell' \in L^{\ell}_{[0,d_2]}$ compute $s_{\varphi_1,\ell'}, s_{\varphi_2,\ell'}$
compute $\mathcal{I}_{s_{\psi,\ell}}$ {the minimal interval covering consistent with $s_{\varphi_1,\ell'}, s_{\varphi_2,\ell'}$, $\ell' \in L^{\ell}_{[0,w_2]}$}
for all $I_i \in \mathcal{I}_{s_{\psi,\ell}}$ **do**
  $V = \{\ell' \in L^{\ell}_{[0,d_2]} | s_{\varphi_1,\ell'}(I_i) = 1\}$
  $Q = \{\ell' \in L^{\ell}_{[d_1,d_2]} | s_{\varphi_2,\ell'}(I_i) = 1\}$
  $T = B^+(Q \bigcup V)$
  while $W \neq \varnothing$ **do**
    $W' = \varnothing$
    for all $\ell \in W$ **do**
      $N = pre(\ell) \bigcap V = \{\ell' \in V | \ell E \ell'\}$
      $V = V \backslash N$
      $W' = W' \bigcup (N \backslash Q)$
    end for
    $W = W'$
  end while
  $s_{\psi,\ell}(I_i) = \begin{cases} 1 & \text{if } \ell \in V, \\ 0 & \text{otherwise.} \end{cases}$
end for
merge adjacent positive interval I_i, i.e., I_i s.t. $s_{\psi,\ell}(I_i) = 1$
return $s_{\psi,\ell}$

that, for all $I_i \in \mathcal{I}_{s_{\psi,\ell}}$

$$\ell \in V_{I_i} \iff (\mathbf{x}, t, \ell) \vDash \phi_1 \mathcal{S}_{[d_1,d_2]} \phi_2 \quad \forall t \in I_i.$$

Furthermore, for the definition of the minimal interval covering (Definition 4, Chapter 6), $s_{\varphi_1,\ell'}, s_{\varphi_2,\ell'}, \ell' \in L^{\ell}_{[0,w_2]}$ have the same value in each $I_i \in \mathcal{I}_{s_{\psi,\ell}}$. This implies, for the Boolean semantics of the surround operator, that, $(\mathbf{x}, \hat{t}, \ell) \vDash \phi_1 \mathcal{S}_{[d_1,d_2]} \phi_2$ for a specific $\hat{t} \in I_i$ if and only if it satisfies the property for all $t \in I_i$.

Let's consider now the distance constraints of the formula. We redefine the property ϕ_1 and ϕ_2 in this way:

$$(\mathbf{x}, t, \ell) \vDash \hat{\phi_1} \iff (\mathbf{x}, t, \ell') \vDash \phi_1 \wedge d(\ell, \ell') \leq d_2,$$

and

$$(\mathbf{x}, t, \ell) \vDash \hat{\phi_2} \iff (\mathbf{x}, t, \ell') \vDash \phi_1 \wedge d(\ell, \ell') \in [d_1, d_2].$$

Hence, we have that $V = \{\ell' | s_{\hat{\phi_1},\ell'}(I_i) = 1\}$ and $Q = \{\ell' | s_{\hat{\phi_2},\ell'}(I_i) = 1\}$.

Furthermore, a location ℓ is a bad location if it can reach a point satisfying $\neg\hat{\phi_1}$ passing for a node $\neg\hat{\phi_2}$. Let's consider the set

$$\mathcal{C}_\ell = \{i \in \mathbb{N} | \exists p : \ell \rightsquigarrow \infty . G, (\mathbf{x}, t, p(i)) \vDash \neg\hat{\phi_1}, \text{ and } \forall j \in \{1, \cdots, i\} (\mathbf{x}, t, p(j)) \vDash \neg\hat{\phi_2}\},$$

where $p : \ell \rightsquigarrow \infty . G$ is a path of the graph G, starting from ℓ, then

$$(\mathbf{x}, t, \ell) \vDash \phi_1 \mathcal{S}_{[d_1,d_2]} \phi_2 \iff (\mathbf{x}, t, \ell) \vDash \hat{\phi_1} \wedge \mathcal{C}_\ell = \varnothing.$$

Hence, what we have to prove at the end is that

$$\ell \in V_{I_i} \iff (\mathbf{x}, t, \ell) \vDash \hat{\phi_1} \wedge \mathcal{C}_\ell = \varnothing, \text{ for a } t \in I_i.$$

We will prove it by induction. From this point on, we fix the trace $\mathbf{x}$ and the time t and we will write $\ell \vDash \phi$ to indicate $(\mathbf{x}, t, \ell) \vDash \phi$ and V for V_{I_i}.

(⇒) We have to prove that if $(\mathbf{x}, t, \ell) \vDash \hat{\phi_1} \wedge \min \mathcal{C}_\ell = k$ then ℓ is removed at iteration k from V.

(basis step) $\ell \vDash \hat{\phi_1} \wedge \min \mathcal{C}_\ell = 1$ then $p(1) \vDash \neg\hat{\phi_1} \wedge p(1) \vDash \neg\hat{\phi_2}$. This implies that $\exists \ell' \in T = B^+(Q \bigcup V)$, and $(\ell, \ell') \in E$, then ℓ is removed from V at the first iteration.

(inductive step) Let's suppose that if $\ell \vDash \hat{\phi}_1 \wedge \min \mathcal{C}_\ell = k$ then ℓ is removed at iteration k from V. We have to prove that this is true also for $k+1$. Let's suppose that $\ell \vDash \hat{\phi}_1 \wedge \min \mathcal{C}_\ell = k+1$. This implies that $p(k+1) \vDash \neg\hat{\phi}_1$ and $\forall j \in \{1, \cdots, k\}$, $p(j) \vDash \neg\hat{\phi}_2$. But if $k+1 = \min \mathcal{C}_\ell$ then $\ell' = p(1) \vDash \hat{\phi}_1$ and $\min \mathcal{C}_{\ell'} = k$, i.e., ℓ' is removed at iteration k from V_{I_i}, then ℓ is removed at iteration $k+1$ because $(\ell, \ell') \in E$.

($\Leftarrow$) We have to prove that if ℓ is removed at iteration k from V_{I_i} then $\ell \vDash \hat{\phi}_1 \wedge \min \mathcal{C}_\ell = k$.

(basis step) If ℓ is removed from V at the first iteration then $\exists \ell' \in T$ s.t. $(\ell, \ell') \in E$ and $(\mathbf{x}, t, \ell') \vDash \neg\hat{\phi}_1 \wedge \neg\hat{\phi}_2$, this implies $\min \mathcal{C}_\ell = 1$.

(inductive step) Let's suppose that if ℓ is removed at iteration k from V then $\ell \vDash \hat{\phi}_1 \wedge \min \mathcal{C}_\ell = k$. We have to prove that this is true also for $k+1$. Let's suppose that ℓ is removed at iteration $k+1$ from V. This implies that $\exists \ell' \in L$ s.t. $(\ell, \ell') \in E$ and $\ell' \in T$ but this means that ℓ' was removed from V at the previous iteration k and from the inductive step we have $\min \mathcal{C}_{\ell'} = k$. If $\min \mathcal{C}_{\ell'} = k$ then $\exists p : \ell' \rightsquigarrow \infty.G$ s.t. $p(k) \vDash \neg\hat{\phi}_1$ and, $\forall i \in \{1, \cdots, k\}, p(i) \vDash \neg\hat{\phi}_2$. But $(\ell, \ell') \in E$ and $(\mathbf{x}, t, \ell') \vDash \neg\hat{\phi}_2$ (because $\ell' \in T$) then $\exists p' : \ell \rightsquigarrow \infty.G$ s.t. $p(k+1) \vDash \neg\hat{\phi}_1$ and, $\forall i \in \{1, \cdots, k+1\}, p(i) \vDash \neg\hat{\phi}_2$. This implies that $\min \mathcal{C}_\ell \leq k+1$, but it can be less than $k+1$ because in that case it has to be removed before. Hence, we can conclude that $\min \mathcal{C}_\ell = k+1$. ∎

A.2 Correctness of the Quantitative Monitoring Algorithm for the Surround Operator

In this section, we present the proofs of Theorem 6.1, Corollary 6.1 and Proposition 6.2. For simplicity, we report again the statements.

Theorem A.2 . *Let s_1 and s_2 be as in Definition 5, and*

$$s(\ell) = \max_{A \subseteq L, \ell \in A} \left(\min\left(\min_{\ell' \in A} s_1(\ell'), \min_{\ell' \in B^+(A)} s_2(\ell')\right)\right),$$

then

$$\lim_{i\to\infty} \mathcal{X}(i,\ell) = s(\ell), \qquad \forall \ell \in L.$$

Moreover, $\exists K > 0$ *s. t.* $\mathcal{X}(j,\ell) = s(\ell), \forall j \geq K$.

Note that s is equivalent to the quantitative semantics of the surround operator $\varphi_1 \mathcal{S} \varphi_2$, with s_i denoting the robustness of φ_i, without the distance constraints. We first present two lemmas, followed by the proof of Theorem A.2.

Lemma A.1 *If* $\mathcal{X}(k+1,\ell) = \mathcal{X}(k,\ell)$ *for all* $\ell \in L$ *then,* $\forall i > k$, $\mathcal{X}(i,\ell) = \mathcal{X}(k,\ell)$.
Proof: *By induction.*

- *(basis step) i=k +1 is true by hypothesis,*
- *(inductive step) suppose the assert holds for* $i > k$*, i.e.,* $\mathcal{X}(i,\ell) = \mathcal{X}(k,\ell)$ *(I.H.), then we have to prove that it holds for* $i+1$.

$$\begin{aligned}
\mathcal{X}(i+1,\ell) &= \min(\mathcal{X}(i,\ell), \min_{\ell' | \ell E \ell'} (\max(\mathcal{X}(i,\ell'), s_2(\ell')))) && \{\textit{by Def. of } \mathcal{X}\} \\
&= \min(\mathcal{X}(k,\ell), \min_{\ell' | \ell E \ell'} (\max(\mathcal{X}(k,\ell'), s_2(\ell')))) && \{\textit{by I.H.}\} \\
&= \mathcal{X}(k+1,\ell) = \mathcal{X}(k,\ell). && \{\textit{by Def. of } \mathcal{X}\}
\end{aligned}$$

∎

Lemma A.2 *Let* A_ℓ *be the subregion that maximizes* $s(\ell)$*, then,* $\forall \ell' \in A_\ell$, $s(\ell') \geq s(\ell)$.
Proof: *If* A_ℓ *is the subregion that maximizes* $s(\ell)$ *then*

$$s(\ell) = \min(\min_{\ell' \in A_\ell} s_1(\ell'), \min_{\ell' \in B^+(A_\ell)} s_2(\ell')))$$

Suppose by contradiction that $\exists \hat{\ell} \in A_\ell$ *s.t.* $s(\hat{\ell}) < s(\ell)$*. Let* $Q = \{A \subseteq L, \hat{\ell} \in A\}$. *Then*

$$s(\hat{\ell}) = \max_{A \in Q}(\min(\min_{\ell' \in A} s_1(\ell'), \min_{\ell' \in B^+(A)} s_2(\ell')))$$

and $s(\hat{\ell}) < s(\ell)$ *implies*

$$\max_{A \in Q}(\min(\min_{\ell' \in A} s_1(\ell'), \min_{\ell' \in B^+(A)} s_2(\ell'))) < \min(\min_{\ell' \in A_\ell} s_1(\ell'), \min_{\ell' \in B^+(A_\ell)} s_2(\ell')))$$

But A_ℓ *is a subset of L and* $\hat{\ell} \in A_\ell$ *therefore* $A_\ell \in Q$*, thus the inequality can not hold.* ∎

Proof: [of Theorem 6.1] We have to prove that (1) $\mathcal{X}(i,\ell)$ converges in a finite number of steps, in each location ℓ, to $\mathcal{X}(\ell) \in \mathbb{R}^*$ and that (2) $\forall \ell \in L$, $\mathcal{X}(\ell) = s(\ell)$.

1. Convergence of $\mathcal{X}$.
   First note that $\mathcal{X}(i,\ell) \geq \min(\mathcal{X}(i,\ell), \min_{\ell' | \ell E \ell'}(\max(\mathcal{X}(i,\ell'), s_2(\ell')))) = \mathcal{X}(i+1,\ell)$, thus $\mathcal{X}_{|\ell}$ is a monotonic decreasing function. Second, note that $\mathcal{X}(i,\ell) \in \{s_j(\ell) \big| j \in \{1,2\}, \ell \in L\}$ is a finite set of sortable values. So, in every step, $\mathcal{X}$ takes a value of a sortable finite set. Finally, if it happens that for a step, for all $\ell \in L$, $\mathcal{X}(i,\ell)$ does not change then, from Lemma A.1, it will remain the same for all the next steps. The convergence of $\mathcal{X}$ to the maximum fixed point follows then from Tarsky's theorem.

2. We have to prove that $\forall \ell$, $\mathcal{X}(\ell) = s(\ell)$.

   Let A_ℓ be the subregion that maximizes $s(\ell)$ then

   $$s(\ell) = \min(\min_{\ell' \in A_\ell} s_1(\ell'), \min_{\ell' \in B^+(A_\ell)} s_2(\ell'))).$$

   First we prove that (2a) $\forall \ell$, $\mathcal{X}(\ell) \geq s(\ell)$ and then that (2b) they are equal.

   2a) To prove that $\mathcal{X}(\ell) \geq s(\ell)$ it suffices to prove that, for a generic ℓ, $\forall i \in \mathbb{N}$, $\mathcal{X}(i,\ell) \geq s(\ell)$, and for the convergence of $\mathcal{X}$ that $\exists j \in \mathbb{N}$ s.t. $\mathcal{X}(\ell) = \mathcal{X}(j,\ell), \forall \ell, \forall j \geq i$. The proof is by induction.

   - (basis step)

     $$\begin{aligned}
     \mathcal{X}(0,\ell) &= s_1(\ell) && \{\text{by Def. of } \mathcal{X}\} \\
     &\geq \min_{\ell' \in A_\ell} s_1(\ell') && \{\text{Because } \ell \in A_\ell\} \\
     &\geq \min(\min_{\ell' \in A_\ell} s_1(\ell'), \min_{\ell' \in B^+(A_\ell)} s_2(\ell'))) && \{\min \text{ property}\} \\
     &= s(\ell) && \{\text{by Def. of } s(\ell)\}
     \end{aligned}$$

   - (inductive step) Assume $\mathcal{X}(i,\ell) \geq s(\ell)$, to prove that $\mathcal{X}(i+1,\ell) \geq s(\ell)$;

$$\mathcal{X}(i+1,\ell) = \min(\mathcal{X}(i,\ell), \min_{\ell'|\ell E \ell'}(\max(\mathcal{X}(i,\ell'), s_2(\ell')))) \quad \{\text{by Def. of } \mathcal{X}\}$$

We know by I.H. that $\mathcal{X}(i,\ell) \geq s(\ell)$, so it remains to be shown that also:

$$\min_{\ell'|\ell E \ell'}(\max(\mathcal{X}(i,\ell'), s_2(\ell'))) \geq s(\ell) \tag{A.1}$$

Note that it is assumed that $\ell \in A_\ell$ and that ℓ' are direct neighbours of ℓ. Therefore we can distinguish the following two cases:

- Suppose $\ell' \in A_\ell$. By I.H. we know that $\mathcal{X}(i,\ell') \geq s(\ell')$ and by Lemma A.2 we also know that $s(\ell') \geq s(\ell)$. For what concerns $s_2(\ell')$, if $s_2(\ell') \leq \mathcal{X}(i,\ell')$ then the max leads to $\mathcal{X}(i,\ell') \geq s(\ell)$. If instead $s_2(\ell') \geq \mathcal{X}(i,\ell') \geq s(\ell)$, then obviously also $s_2(\ell') \geq s(\ell)$. So inequation (A.1) holds in this case.
- Suppose $\ell' \in B^+(A_\ell)$. Then, by definition of $s(\ell)$ we know that $s_2(\ell') \geq s(\ell)$. So, if $s_2(\ell') \geq \mathcal{X}(i,\ell')$ then the inequation holds. If $\mathcal{X}(i,\ell') \geq s_2(\ell')$ then since $s_2(\ell') \geq s(\ell)$, inequation (A.1) also holds.

2b) Suppose by contradiction that $\exists \hat{\ell} \in L$ s.t. $\mathcal{X}(\hat{\ell}) > s(\hat{\ell})$. At the fixed point we have that

$$\mathcal{X}(\hat{\ell}) = \min(\mathcal{X}(\hat{\ell}), \min_{\ell|\hat{\ell} E \ell}(\max(\mathcal{X}(\ell), s_2(\ell))))$$

This means that the inequality

$$\min_{\ell|\hat{\ell} E \ell}(\max(\mathcal{X}(\ell), s_2(\ell))) > s(\hat{\ell}) \tag{A.2}$$

has to be true.

Let $A \subseteq L$, we define:

- $C(A) := \{\ell \in L \,|\, \exists \ell' \in A \text{ s.t. } \ell' E \ell \wedge \mathcal{X}(\ell) \geq s_2(\ell)\}$
- $C^i(A) = C(C^{i-1}(A))$

We can then compute the closure of C, as $C^*(A) = A \bigcup_{i=0}^{\infty} C^i(A)$. Because of the definition of C and the inequality (A.2), we have that $s_1(\ell) \geq \mathcal{X}(\ell) > s(\hat{\ell})$, $\forall \ell \in C^*(\{\hat{\ell}\})$, and that $s_2(\ell) > s(\hat{\ell})$, $\forall \ell \in B^+(C^*(\{\hat{\ell}\}))$; hence

$$\min(\min_{\ell \in C^*(\{\hat{\ell}\})} s_1(\ell), \min_{\ell \in B^+(C^*(\{\hat{\ell}\}))} s_2(\ell))) > s(\hat{\ell})$$

i.e,.

$$\min(\min_{\ell \in C^*(\{\hat{\ell}\})} s_1(\ell), \min_{\ell \in B^+(C^*(\{\hat{\ell}\}))} s_2(\ell))) > \min(\min_{\ell \in A_{\hat{\ell}}} s_1(\ell), \min_{\ell' \in B^+(A_{\hat{\ell}})} s_2(\ell')))$$

but this contradicts the assumption of maximality of $A_{\hat{\ell}}$. ∎

Corollary A.1 *Given an $\hat{\ell} \in L$, let $\psi = \varphi_1 \mathcal{S}_{[d_1,d_2]} \varphi_2$ and*

$$s_1(\ell) = \begin{cases} \rho(\phi_1, \mathbf{x}, t, \ell) & \text{if } 0 \leq d(\hat{\ell}, \ell) \leq d_2 \\ -\infty & \text{otherwise.} \end{cases}$$

$$s_2(\ell) = \begin{cases} \rho(\phi_2, \mathbf{x}, t, \ell) & \text{if } d_1 \leq d(\hat{\ell}, \ell) \leq d_2 \\ -\infty & \text{otherwise.} \end{cases}$$

Then $\rho(\psi, \mathbf{x}, t, \hat{\ell}) = s(\hat{\ell}) = \max_{A \subseteq L, \hat{\ell} \in A} (\min(\min_{\ell \in A} s_1(\ell), \min_{\ell \in B^+(A)} s_2(\ell)))$.

Proof: We recall that

$$\rho(\psi, \mathbf{x}, t, \hat{\ell}) = \max_{A \subseteq L^{\hat{\ell}}_{[0,d_2]}, \ell \in A, B^+(A) \subseteq L^{\hat{\ell}}_{[d_1,d_2]}} (\min(\min_{\ell \in A} \rho(\phi_1, \mathbf{x}, t, \ell), \min_{\ell \in B^+(A)} \rho(\phi_2, \mathbf{x}, t, \ell))),$$

where $L^{\hat{\ell}}_{[d_1,d_2]} := \{\ell \in A | d_1 \leqslant d(\ell, \hat{\ell}) \leq d_2\}$. This means that $\ell \in A$ iff $d(\ell, \hat{\ell}) \leq d_2$ and, for all $\ell' E \ell$, $d_1 \leqslant d(\ell', \hat{\ell}) \leq d_2$.

So, we consider a restricted number of subsets of L for ρ and all the possible subsets of L for s. Furthermore, the value of the locations considered by both are always the same, i.e., the value of s_1 and s_2 differ only in the locations considered by s and not by ρ. For this reason $s(\ell) \geq \rho(\ell)$.

Let A_ρ be the subset that maximizes ρ of $\hat{\ell}$ and A_s the subset that maximizes s of $\hat{\ell}$. And suppose by contradiction that

$$\min(\min_{\ell \in A_s} s_1(\ell), \min_{\ell' \in B^+(A_s)} s_2(\ell))) > \min(\min_{\ell \in A_\rho} \rho(\phi_1, \mathbf{x}, t, \ell), \min_{\ell \in B^+(A_\rho)} \rho(\phi_2, \mathbf{x}, t, \ell))),$$

but the values considered by s and not by ρ are all equal to $-\infty$ (see line 8 of Alg. 3), so if A_s has a location that cannot be considered by ρ it means that

$$\min(\min_{\ell \in A_s} s_1(\ell), \min_{\ell' \in B^+(A_s)} s_2(\ell))) = -\infty$$

but minus infinity cannot be bigger than any number. ∎

Proposition A.1 *Let d_G be the diameter of the graph G and $\mathcal{X}(\ell)$ the fixed point of $\mathcal{X}(i, \ell)$, then $\mathcal{X}(\ell) = \mathcal{X}(d_G + 1, \ell)$ for all $\ell \in L$.*

Proof: The graph diameter of G is equal to $d_g = \max_{\ell, \ell' \in L} d(\ell, \ell')$. Recall that $\mathcal{X}(d_g, \ell) \in \{s_j(\ell) \,\big|\, j \in \{1, 2\}, \ell \in L\}$ is a finite set of sortable values. At step zero the value of $\mathcal{X}$ is equal to s_1 in all the locations. At each next step, the value of $\mathcal{X}(i, \ell)$ depends only on the value of $\mathcal{X}$ in the same location at the previous step and the value of s_2 and $\mathcal{X}$ in the previous step in the direct neighbours of ℓ, $\ell' E \ell$. This means that, after a number of steps equal to the diameter of the graph, i.e., the longest shortest path of the network, $\mathcal{X}$, for all nodes ℓ, has taken into account the values s_1 and s_2 of all the nodes. ∎

Appendix B

The jSSTL View of the ECLIPSE Plugin

In this appendix, we describe the jSSTL view with a running example.

Let us consider the Cholera case study presented in Chapter 6, Section 6.5. The model describes the spread of a Cholera infection along the communities that live close to a river. In Figure B.1, we can see the ECLIPSE plugin. On the left, there is the editor containing the script with the SSTL properties that we want to analyse in our scenario. On the right side, instead, there is the jSSTL view (where the panel with the signal is selected). As we can see, the jSSTL view provides three different panels: to visualise the spatial models (*Model* panel), the relevant data declared in a script (*Script* panel), and to plot the system's trajectories and the Boolean and quantitative satisfaction signals (*Signal* panel).

The *Model* panel can be used to see a graph-based representation of a spatial model, Figure B.2. The spatial model can be can be imported as a `.tra` file.

The *Script* panel, Figure B.3, represents the main information of our scenario: the list of variables, parameters and formulae names. Currently, CSV and tabular based ascii files are supported for both input and output of signals. The traces have to be imported with a single file for each variable and location, e.g., for the infected individual I we will

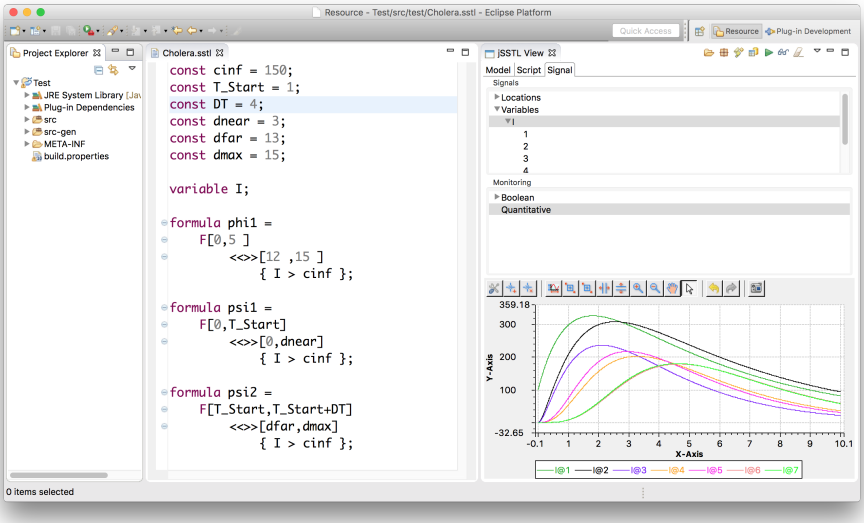


Figure B.1: The jSSTL ECLIPSE plugin

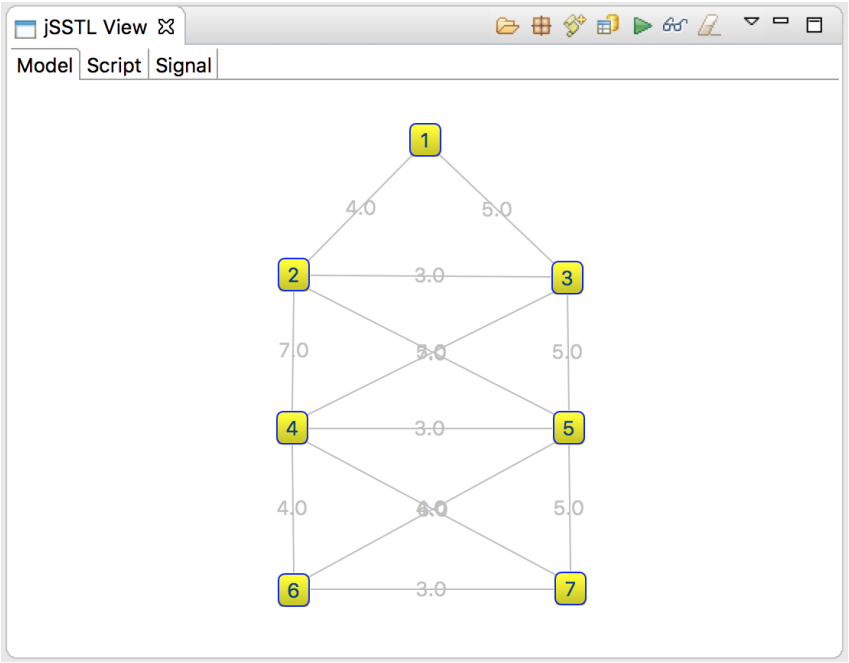


Figure B.2: Visualisation of the space in the jSSTL ECLIPSE plugin.

have the files values_i_I.dat, with $i \in \{1, .., 7\}$.

In the *Signal* panel, the trajectories and the Boolean and quantitative signals can be visualised. In Figure B.4, we can see the trace x_I, the variation in time of the number of infected individuals in each location.

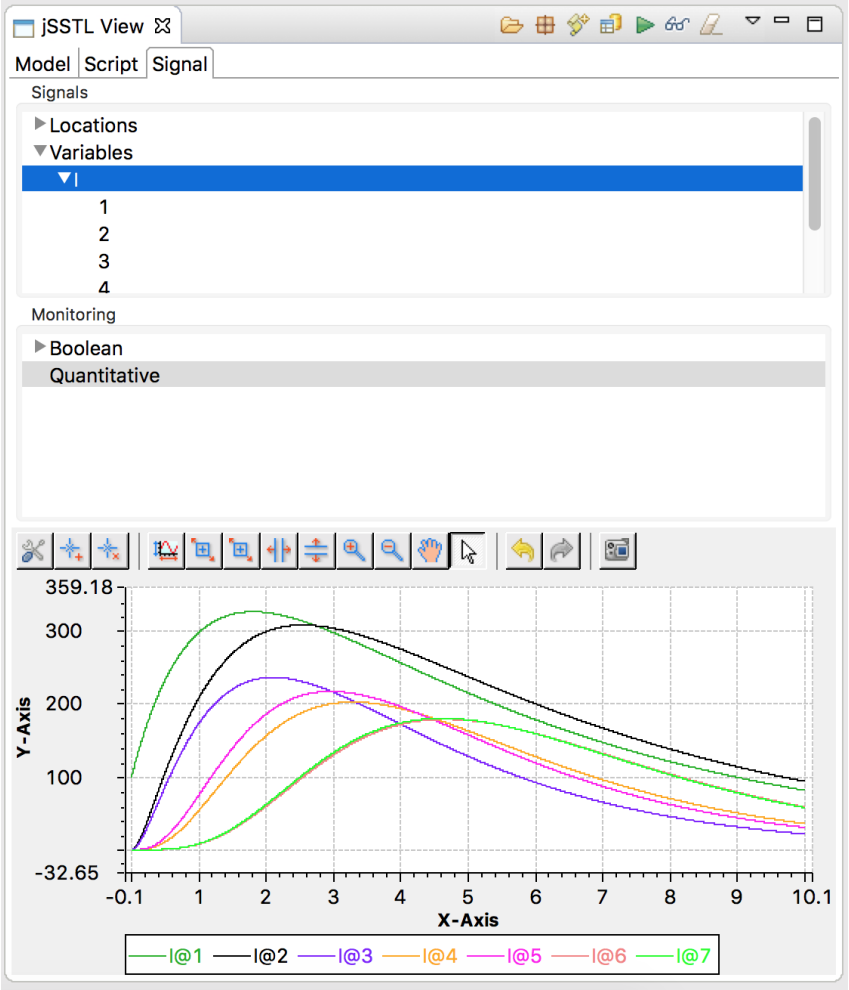


Figure B.4: Plot of x_I, number of infected individual in each location.

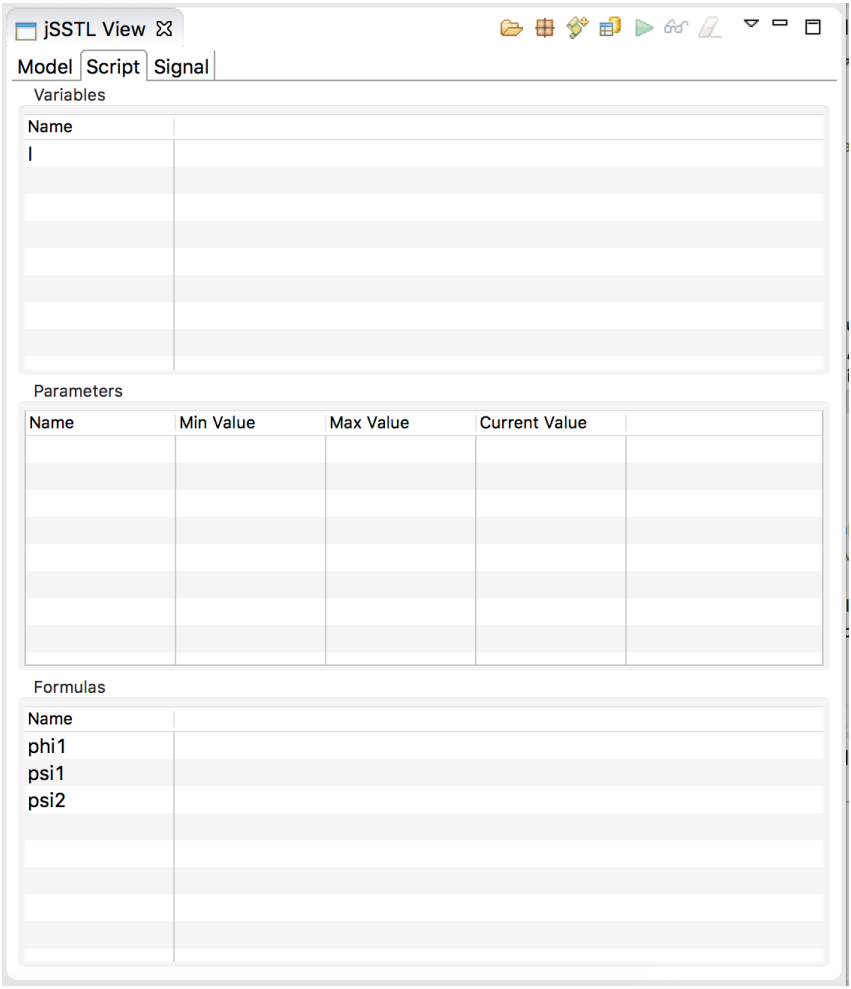


Figure B.3: List of variables, formula parameters and formula names in the jSSTL view.

Figure B.5 shows the spatio-temporal Boolean satisfaction of the property psi1 (in each location), and Figure B.6 the quantitative one. We recall that the satisfaction of a formula corresponds to the value at time zero. We can choose which variables and which locations to plot.

The plugin is still in development and new features will be integrated. For instance, we are working on an online monitoring procedure where trajectories are collected from external data sources and on new kinds of format for input and output of data.

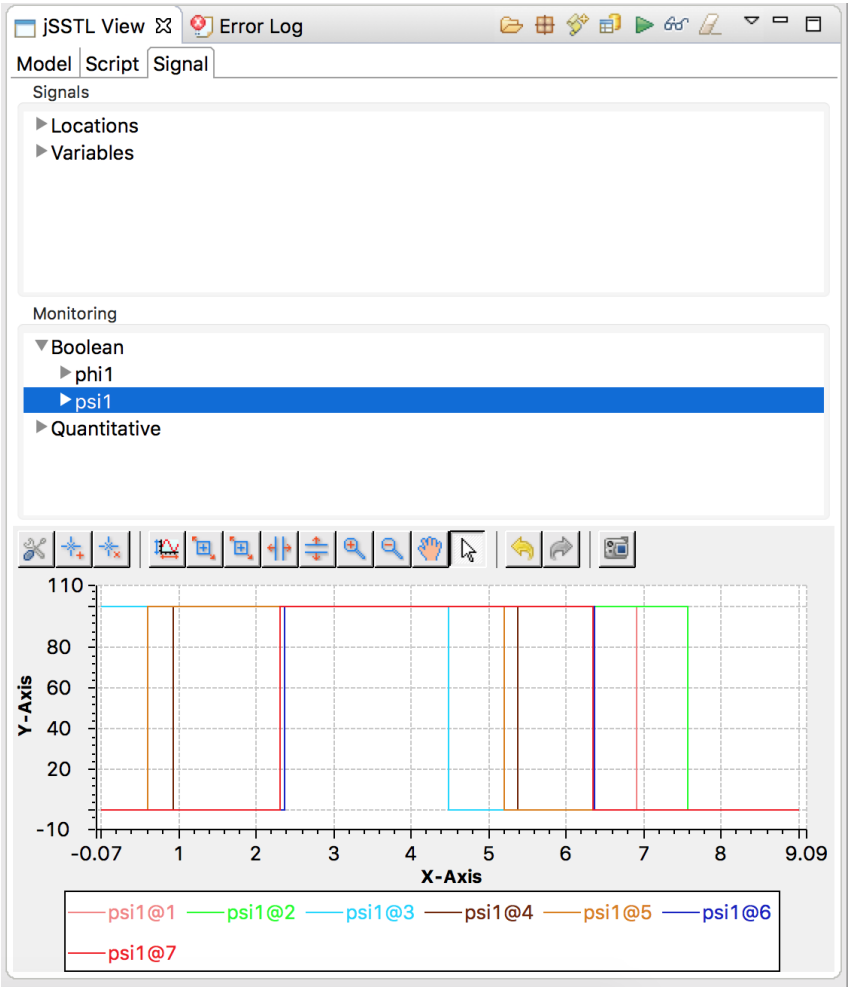


Figure B.5: Plot of the Boolean semantics of the properties phi1.

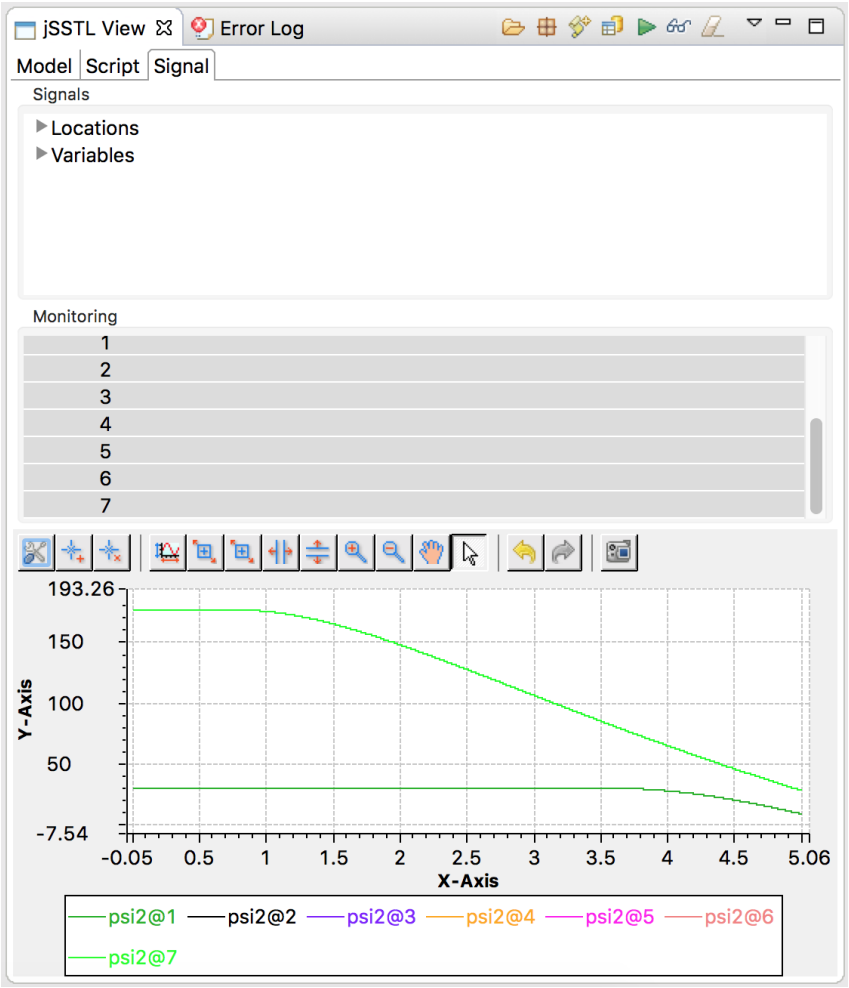


Figure B.6: Plot of the quantitative semantics of the properties phi1.

Contributions to co-authored papers

As described in the acknowledgements, most part of this thesis has been published. In the publication of the computer science community, the authors are usually in alphabetical order. To clarify my contribution, I added here the details of what I have done in each paper.

In (BBNS13; BBNS15), my main contributions are in the design of all the case studies, their implementation, simulation, analysis and their system design experiments.

In (NB14), I worked, supervised by Luca Bortolussi, on the whole paper: syntax, semantics, algorithms and case study.

In (NBC+15), I defined the syntax and the semantics together with the other authors, the algorithm for quantitative surround was designed principally in collaboration with Michele Loreti and the proof of the correctness with the help of Luca Bortolussi. I did the first version of the Java implementation of SSTL, supervised by Michele Loreti and Luca Bortolussi. I implemented and analysed the case study.

In (BBM+15), I designed and implemented the case study, I described SSTL and specified the french flag property, I elaborated the real data, and worked on the experiments in collaboration with Dimitrios Milios.

References

[AFH96] R. Alur, T. Feder, and T.A. Henzinger. The benefits of relaxing punctuality. *J. ACM*, 1996. 5, 30, 142

[AGBB14] E. Aydin Gol, E. Bartocci, and C. Belta. A formal methods approach to pattern synthesis in reaction diffusion systems. In *Proc. of CDC 2014: the 53rd IEEE Conference on Decision and Control*, page to appear. IEEE, 2014. 41, 87

[ALFS11] Y. Annapureddy, C. Liu, G. Fainekos, and S. Sankaranarayanan. S-TaLiRo: A Tool for Temporal Logic Falsification for Hybrid Systems. In *Proc. of TACAS 2011: the 17th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, volume 6605 of *Lecture Notes in Computer Science*, pages 254–257. Springer Berlin / Heidelberg, 2011. 87

[Alo07] U. Alon. *An introduction to systems biology: design principles of biological circuits*. Chapman & Hall/CRC, 2007. 7, 70, 75

[APHvB07] M. Aiello, I. Pratt-Hartmann, and J. van Benthem, editors. *Handbook of Spatial Logics*. Springer, 2007. 6, 38

[ASSB96] A. Aziz, K. Sanwal, V. Singhal, and R. Brayton. Verifying continuous time markov chains. pages 269–276. Springer, 1996. 43

[BAM+08] E. Bertuzzo, S. Azaele, A. Maritan, M. Gatto, I. Rodriguez-Iturbe, and A. Rinaldo. On the space-time evolution of a cholera epidemic. *Water Resources Research*, 44(1):W01424, 2008. 9, 107, 110, 111

[BBM+15] E. Bartocci, L. Bortolussi, D. Milios, L. Nenzi, and G. Sanguinetti. Studying Emergent Behaviours in Morphogenesis Using Signal Spatio-Temporal Logic. In Alessandro Abate and David afrnek, editors, *Hybrid Systems Biology*, number 9271 in Lecture Notes in

Computer Science, pages 156–172. Springer International Publishing, September 2015. DOI: 10.1007/978-3-319-26916-0_9. xiii, 9, 11, 121, 165

[BBN13] E. Bartocci, L. Bortolussi, and L. Nenzi. A Temporal Logic Approach to Modular Design of Synthetic Biological Circuits. In *Proc. of CMSB 2013: the 11th International Conference on Computational Methods in Systems Biology, IST Austria, Klosterneuburg, Austria, September 23-25*, volume 8130 of *Lecture Notes in Computer Science*, pages 164–178. Springer-Verlag, 2013. 87

[BBNS13] E. Bartocci, L. Bortolussi, L. Nenzi, and G. Sanguinetti. On the Robustness of Temporal Properties for Stochastic Models. *Electronic Proceedings in Theoretical Computer Science*, 125:3–19, August 2013. arXiv: 1309.0866. xiii, 7, 8, 11, 56, 165

[BBNS15] E. Bartocci, L. Bortolussi, L. Nenzi, and G. Sanguinetti. System design of stochastic models using robustness of temporal properties. *Theoretical Computer Science*, 587:3–25, July 2015. xiii, 7, 8, 11, 56, 165

[BBS14a] E. Bartocci, L. Bortolussi, and S. Sanguinetti. Data-driven statistical learning of temporal logic properties. In *Proc. of FORMATS 2014: the 12th International Conference on Formal Modeling and Analysis of Timed Systems*, volume 8711 of *LNCS*, pages 23–37, 2014. 147

[BBS+14b] S. Bufo, E. Bartocci, G. Sanguinetti, M. Borelli, U. Lucangelo, and L. Bortolussi. Temporal logic based monitoring of assisted ventilation in intensive care patients. In B. Steffen and T. Margaria, editors, *Proc. of ISoLA 2014: 6th International Symposium On Leveraging Applications of Formal Methods, Verification and Validation*, volume 8803 of *LNCS*, pages 391–403, 2014. 147

[BCDŠ13] L. Brim, M. Ceska, S. Drazan, and D. Šafránek. Exploring Parameter Space of Stochastic Biochemical Systems using Quantitative Model Checking. In *Proc. of CAV 2013: the 25th International Conference on Computer Aided Verification, Saint Petersburg, Russia, July 13-19*, volume 8044 of *Lecture Notes in Computer Science*, pages 107–123. Springer-Verlag, 2013. 88

[BCHG+97] C. Baier, E.M. Clarke, V. Hartonas-Garmhausen, M.Z. Kwiatkowska, and M. Ryan. Symbolic model checking for probabilistic processes. In *Proc. of ICALP '97, the 24th International Colloquium on Automata, Languages and Programming, Bologna, Italy, July 7-11*, volume 1256 of *Lecture Notes in Computer Science*, pages 430–440. Springer Berlin Heidelberg, 1997. 4, 56

[BDL12] R. Brochenin, S. Demri, and E. Lozes. On the almighty wand. *Inf. Comput.*, 211:106–137, 2012. 45

[BGK+11] E. Bartocci, R. Grosu, P. Katsaros, C. Ramakrishnan, and S. A. Smolka. Model Repair for Probabilistic Systems. In *Proc. of TACAS 2011: the 17th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, volume 6605 of *Lecture Notes in Computer Science*, pages 326–340. Springer Berlin / Heidelberg, 2011. 87

[BHHK03] C. Baier, B. Haverkort, H. Hermanns, and J.-P. Katoen. Model-checking algorithms for continuous-time markov chains. *IEEE Trans. Softw. Eng.*, 29(6):524–541, 2003. 4, 56

[BHLM13] L. Bortolussi, J. Hillston, D. Latella, and M. Massink. Continuous approximation of collective systems behaviour: a tutorial. *Performance Evaluation*, 70(5):317–349, May 2013. 16, 19

[BHMU11] A. T. Bittig, F. Haack, C. Maus, and A. M. Uhrmacher. Adapting rule-based model descriptions for simulating in continuous and hybrid space. In *Proceedings CMSB'11*, pages 161–170, 2011. 6

[Bil99] P. Billingsley. *Convergence of probability measures*. Wiley, 1999. 23, 58, 59, 65

[Bil12] P. Billingsley. *Probability and measure*. Wiley, 2012. 15, 60

[Bis06] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006. 48, 68, 74

[BK08] C. Baier and J. Katoen. *Principles of model checking*. MIT Press, Cambridge, Mass., 2008. 4

[BLB05] M. L. Bujorianu, J. Lygeros, and M. C. Bujorianu. Bisimulation for General Stochastic Hybrid Systems. In *Proc. of HSCC 2005: the 8th International Workshop on Hybrid Systems: Computation and Control*, volume 3414 of *Lecture Notes in Computer Science*, pages 198–214. Springer-Verlag, 2005. 3

[BLMP12] E. Bartocci, P. Liò, E. Merelli, and N. Paoletti. Multiple Verification in Complex Biological Systems: The Bone Remodelling Case Study. *T. Comp. Sys. Biology*, 14:53–76, 2012. 86

[BM12] K. Bae and J. Meseguer. A rewriting-based model checker for the linear temporal logic of rewriting. *Electron. Notes Theor. Comput. Sci.*, 290:19–36, December 2012. 6, 45

[BMS15] L. Bortolussi, D. Milios, and G. Sanguinetti. U-check: Model checking and parameter synthesis under uncertainty. In *Proc. of QEST 2015: 12th International Conference on Quantitative Evaluation of SysTems*, volume 9259 of *LNCS*, pages 89–104, 2015. 10, 70, 137, 142, 145

[BMS16] L. Bortolussi, D. Milios, and G. Sanguinetti. Smoothed model checking for uncertain Continuous-Time Markov Chains. *Information and Computation*, 2016. 9, 11, 47, 52, 53, 54, 121, 122, 124, 147

[BP08] L. Bortolussi and A. Policriti. Hybrid Approximation of Stochastic Process Algebras for Systems Biology World Congress. In *Proc. of IFAC WC 2008: The 17th World Congress of the International Federation of Automatic Control, COEX, Korea, South*, volume 17, pages 12599–12606, 2008. 8, 70, 82

[BP09] S. Bensalem and D. A. Peled. *Runtime Verification: 9th International Workshop, RV 2009, Grenoble, France, June 26-28, 2009, Selected Papers*. Springer Science & Business Media, September 2009. 4

[BP10] L. Bortolussi and A. Policriti. Hybrid Dynamics of Stochastic Programs. *Theoretical Computer Science*, 411(20):2052–2077, 2010. 8, 70, 82

[BP13] L. Bortolussi and A. Policriti. Hybrid) automata and (stochastic) programs. The hybrid automata lattice of a stochastic program. *J. Log. Comput.*, 23(4):761–798, 2013. 21, 22

[BS13] L. Bortolussi and G. Sanguinetti. Learning and Designing Stochastic Processes from Logical Constraints. In *Proc. of QEST 2013: the 10th International Conference on Quantitative Evaluation of Systems, Buenos Aires, Argentina, August 27-30*, volume 8054 of *Lecture Notes in Computer Science*, pages 89–105. Springer-Verlag, 2013. 68, 69, 142

[BS14] L. Bortolussi and G. Sanguinetti. A statistical approach for computing reachability of non-linear and stochastic dynamical systems. In *Proc. of QEST 2014: the 11th International Conference on Quantitative Evaluation of Systems*, volume 8657 of *Lecture Notes in Computer Science*, pages 41–56. Springer, 2014. 87, 147

[BS15] L. Bortolussi and G. Sanguinetti. Learning and designing stochastic processes from logical constraints. *LOGICAL METHODS IN COMPUTER SCIENCE*, 11:1–24, 2015. 68, 69, 87, 142, 147

[BYWB07] G. Batt, B. Yordanov, R. Weiss, and C. Belta. Robustness analysis and tuning of synthetic gene networks. *Bioinformatics*, 23(18):2415–2422, 2007. 85

[Cai04] L. Caires. Behavioral and Spatial Observations in a Logic for the π-Calculus. In Igor Walukiewicz, editor, *Foundations of Software Science and Computation Structures*, number 2987 in Lecture Notes in Computer Science, pages 72–89. Springer Berlin Heidelberg, March 2004. DOI: 10.1007/978-3-540-24727-2_7. 43

[Cas01] I. Castellani. Chapter 15 - Process Algebras with Localities A2 - Smolka, J.A. BergstraA. PonseS.A. In *Handbook of Process Algebra*, pages 945–1045. Elsevier Science, Amsterdam, 2001. 38, 41

[CC03] L. Caires and L. Cardelli. A spatial logic for concurrency (part I). *Information & Computation*, 186(2):194–235, 2003. 43

[CDKM11] T. Chen, M. Diciolla, M. Kwiatkowska, and A. Mereacre. Time-bounded verification of ctmcs against real-time specifications. In *Proc. of FORMATS 2011, the 9th International Conference on Formal Modeling and Analysis of Timed Systems, Aalborg, Denmark, September 21–23*, volume 6919 of *Lecture Notes in Computer Science*, pages 26–42, Berlin, Heidelberg, 2011. 4

[CFS06] L. Calzone, F. Fages, and S. Soliman. BIOCHAM: an environment for modeling biological systems and formalizing experimental knowledge. *Bioinformatics*, 22:1805–1807, 2006. 87

[CG98] L. Cardelli and Andrew D. Gordon. Mobile ambients. In *In Proceedings of POPL'98*. ACM Press, 1998. 43

[CG00] L. Cardelli and A. D. Gordon. Anytime, anywhere: Modal logics for mobile ambients. In *Proceedings of the 27th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '00, pages 365–377, New York, NY, USA, 2000. ACM. 6, 38, 43, 44

[CGG02] L. Cardelli, P. Gardner, and G. Ghelli. A spatial logic for querying graphs. In Peter Widmayer, Francisco Triguero Ruiz, Rafael Morales Bueno, Matthew Hennessy, Stephan Eidenbenz, and Ricardo Conejo, editors, *ICALP*, volume 2380 of *Lecture Notes in Computer Science*, pages 597–610. Springer, 2002. 45

[CGG+15] V. Ciancia, S. Gilmore, G. Grilletti, D. Latella, M. Loreti, and M. Massink. Spatio-temporal model-checking of vehicular movement in public transport systems. *Submitted*, 2015. 6, 40

[CGL+14] V. Ciancia, S. Gilmore, D. Latella, M. Loreti, and M. Massink. Data verification for collective adaptive systems: Spatial model-checking of vehicle location data. In *Proc. of SASOW*, 2014. 40

[CGP06] Y. Cao, D. T. Gillespie, and L. R. Petzold. Efficient step size selection for the tau-leaping simulation method. *The Journal of Chemical Physics*, 124(4):044109, January 2006. 16

[CH09] Federica Ciocchetta and Jane Hillston. Bio-pepa: A framework for the modelling and analysis of biological systems. *Theor. Comput. Sci.*, 410(33-34):3065–3084, 2009. 142

[CJGP99] E. M. Clarke Jr., Orna Grumberg, and Doron A. Peled. *Model Checking*. MIT Press, Cambridge, MA, USA, 1999. 4

[CL10] F. Calzolai and M. Loreti. Simulation and analysis of distributed systems in klaim. volume 6116 of *Lecture Notes in Computer Science*. Springer, 2010. 43

[CLLM14] V. Ciancia, D. Latella, M. Loreti, and M. Massink. Specifying and verifying properties of space. In *Proc. of IFIP-TCS*, 2014. 6, 9, 39, 40, 92, 99, 146

[CM98] L. Caires and L. Monteiro. Verifiable and executable logic specifications of concurrent objects in lpi. In Chris Hankin, editor, *ESOP*, volume 1381 of *Lecture Notes in Computer Science*, pages 42–56. Springer, 1998. 43

[CML08] V. Ciancia, M. Massink, and D. Latella. Logics of space and time, 2014-01-08. QUANTICOL Technical Report. 38

[CMS07] G. Conforti, D. Macedonio, and V. Sassone. Static bilog: a unifying language for spatial structures. *Fundam. Inform.*, 80(1-3):91–110, 2007. 45

[Col08] P. Collins. Computability and representations of the zero set. *Electronic Notes in Theoretical Computer Science*, 221:37–43, 2008. 65

[Dav93] M.H.A. Davis. *Markov Models and Optimization*. Chapman & Hall, 1993. 21, 58

[DFG+11] A. Donzé, E. Fanchon, L. M. Gattepaille, O. Maler, and P. Tracqui. Robustness analysis and behavior discrimination in enzymatic reaction networks. *PLoS One*, 6(9):e24246, 2011. 87

[DFM13] A. Donzé, T. Ferrer, and O. Maler. Efficient Robust Monitoring for STL. In *Proc. of CAV 2013: the 25th International Conference on Computer Aided Verification, Saint Petersburg, Russia, July 13-19*, volume 8044 of *Lecture Notes in Computer Science*, pages 264–279. Springer-Verlag, 2013. 5, 30, 33, 34, 65, 95, 99

[DG08a] A. Degasperi and S. Gilmore. Sensitivity Analysis of Stochastic Models of Bistable Biochemical Reactions. In *Formal Methods for Computational Systems Biology*, volume 5016 of *Lecture Notes in Computer Science*, pages 1–20. Springer-Verlag, 2008. 71

[DG08b] R. Donaldson and D. Gilbert. A Model Checking Approach to the Parameter Estimation of Biochemical Pathways. In *Proc. of CMSB 2008: the 6th International Conference on Computational Systems Biology, Rostock, Germany, October 12-15*, volume 5307 of *Lecture Notes in Computer Science*, pages 269–287. Springer-Verlag, 2008. 87

[DM10] A. Donzé and O. Maler. Robust satisfaction of temporal logic over real-valued signals. In *Proc. of FORMATS 2010, the 8th International Conference on Formal Modeling and Analysis of Timed Systems, Klosterneuburg, Austria, September 8–10*, volume 6246, pages 92–106, 2010. 5, 29, 30, 31, 33, 57, 65, 87, 99, 143

[Don10] A. Donzé. Breach, A Toolbox for Verification and Parameter Synthesis of Hybrid Systems. In *Proc. of CAV 2010: the 22nd International Conference on Computer Aided Verification, Edinburgh, UK*, volume 6174 of *Lecture Notes in Computer Science*, pages 167–170. Springer-Verlag, 2010. 30, 87, 113

[DOS10] V. Dewar, M. A .and Kadirkamanathan, M. Opper, and G. Sanguinetti. Parameter estimation and inference for stochastic reaction-diffusion systems: application to morphogenesis in d. melanogaster. *BMC Systems Biology.*, page 21, 2010. 125

[Dur12] R. Durett. *Essentials of stochastic processes*. Springer, 2012. 3, 16

[ea31] V. Galpin et al. A preliminary investigation of capturing spatial information for CAS, 2014-03-31. QUANTICOL Deliverable D 2.1. 24, 26, 27

[EL00] M.B. Elowitz and S. Leibler. A synthetic oscillatory network of transcriptional regulators. *Nature*, 403:335–338, 2000. 7, 70, 80

[ESS02] A.J. Elowitz, M.B.and Levine, E.D. Siggia, and P.S. Swain. Stochastic gene expression in a single cell. *Science*, 297(5584):1183, 2002. 2

[FB74] R. A. Finkel and J. L. Bentley. Quad trees a data structure for retrieval on composite keys. *Acta Informatica*, 4(1):1–9, March 1974. 6, 38, 41

[FGM12] C. Fricker, N. Gast, and H. Mohamed. Mean field analysis for inhomogeneous bike sharing systems. *DMTCS Proceedings*, pages 365–376, 2012. 00009. 1

[FH14] C. Feng and J. Hillston. PALOMA: A process algebra for located markovian agents. In *Proceedings of QEST'14*, number 8657 in LNCS, pages 265–280, 2014. 6

[FI14] P. Fried and D. Iber. Dynamic scaling of morphogen gradients on growing domains. *Nat. Comm.*, 2014. 114

[fly] Flyex database. `http://urchin.spbcas.ru/flyex/`. 130

[FP07] G. Fainekos and G. Pappas. Robust Sampling for MITL Specifications. In *Proc. of FORMATS 2007: the 5th International Conference on Formal Modeling and Analysis of Timed Systems*, volume 8044 of *Lecture Notes in Computer Science*, pages 264–279. Springer-Verlag, 2007. 87

[FP09] G. E. Fainekos and G. J. Pappas. Robustness of temporal logic specifications for continuous-time signals. *Theor. Comput. Sci.*, 410(42):4262–4291, 2009. 5, 29, 32, 33, 57, 87

[FS11] R. Frei and Giovanna Di Marzo Serugendo. Concepts in complexity engineering. *Int. J. Bio-Inspired Comput.*, 3(2):123–139, April 2011. 1

[Gal99] A. Galton. The mereotopology of discrete space. In Christian Freksa and DavidM. Mark, editors, *Spatial Information Theory. Cognitive and Computational Foundations of Geographic Information Science*, Lecture Notes in Computer Science. Springer Berlin Heidelberg, 1999. 6, 39

[GB00] M. A. Gibson and J. Bruck. Efficient Exact Stochastic Simulation of Chemical Systems with Many Species and Many Channels. *The Journal of Physical Chemistry A*, 104(9):1876–1889, March 2000. 16

[GBB14] E. Aydin Gol, E. Bartocci, and C. Belta. A formal methods approach to pattern synthesis in reaction diffusion systems. In *Proc. of CDC*, 2014. 6, 114, 146

[GBF+11] R. Grosu, G. Batt, F. Fenton, J. Glimm, C. Le Guernic, S. Smolka, and E. Bartocci. From Cardiac Cells to Genetic Regulatory Networks. In *Proc. of CAV 2011: the 23rd International Conference on Computer Aided Verification*, volume 6806 of *Lecture Notes in Computer Science*, pages 396–411. Springer Berlin / Heidelberg, 2011. 86

[GCPDI05] R. Gunawan, Y. Cao, L. Petzold, and F.J. Doyle III. Sensitivity analysis of discrete stochastic systems. *Biophysical Journal*, 88(4):2530, 2005. 7, 70

[Gil77] D.T. Gillespie. Exact stochastic simulation of coupled chemical reactions. *J. of Physical Chemistry*, 81(25), 1977. 16, 28, 47, 70, 76, 113

[GSC+09] R. Grosu, S.A. Smolka, F. Corradini, A. Wasilewska, E. Entcheva, and E. Bartocci. Learning and detecting emergent behavior in networks of cardiac myocytes. *Communications of the ACM*, 52(3):97–105, 2009. 6, 41, 83

[Hen98] T. A. Henzinger. It's about time: Real-time logics reviewed. In *Proceedings of the 9th International Conference on Concurrency Theory*, CONCUR '98, London, UK, UK, 1998. Springer-Verlag. 3

[HJK+15] I. Haghighi, A. Jones, J. Z. Kong, E. Bartocci, Grosu R., and C. Belta. SpaTeL: A Novel Spatial-Temporal Logic and Its Applications to Networked Systems. In *Proc. of HSCC*, 2015. 6, 38, 41, 87, 114, 146, 147

[HKM08] T. Han, J. Katoen, and A. Mereacre. Approximate Parameter Synthesis for Probabilistic Time-Bounded Reachability. In *Proc. of RTSS 2008: the 29th IEEE Real-Time Systems Symposium*, pages 173–182, Nov 2008. 88

[HKMkS00] H. Hermanns, J.P. Katoen, J. Meyer-kayser, and M. Siegle. Towards model checking stochastic process algebra, 2000. 43

[Jae10] J. Jaeger. The gap gene network. *Cellular and Molecular Life Sciences*, pages 243–274, 2010. 127

[JCL+09a] S. K. Jha, E. M. Clarke, C. J. Langmead, A. Legay, A. Platzer, and P. Zuliani. A Bayesian Approach to Model Checking Biological Systems. In P. Degano and R. Gorrieri, editors, *Computational Methods in Systems Biology*, number 5688 in Lecture Notes in Computer Science, pages 218–234. Springer Berlin Heidelberg, January 2009. 48

[JCL+09b] S. K. Jha, E. M. Clarke, C. J. Langmead, A. Legay, A. Platzer, and P. Zuliani. Statistical model checking for complex stochastic models in systems biology. In *Proceedings of CMSB'09*, 2009. 4, 10, 47, 70, 113

[JDDS13] X. Jin, A. Donzé, J. V. Deshmukh, and Sanjit A. Seshia. Mining Requirements from Closed-loop Control Models. In *Proc. of HSCC 2013: the 16th International Conference on Hybrid Systems: Computation and Control*, pages 43–52. ACM, 2013. 83, 84

[JMA09] J. Jaeger and A. Martinez-Arias. Getting the Measure of Positional Information. *PLoS Biology*, 2009. 128

[JMEK07] A. Jilkine, A. F. M. Marée, and L. Edelstein-Keshet. Mathematical model for spatial segregation of the rho-family GTPases based on inhibitory crosstalk. *Bulletin of Mathematical Biology*, 69(6):1943–1978, 2007. 2

[Kal10] O. Kallenberg. *Foundations of Modern Probability*. Springer, 2010. 65

[KCRS11] M. Komorowski, M. J. Costa, D. A. Rand, and M. PH Stumpf. Sensitivity, robustness, and identifiability in stochastic chemical kinetics models. *PNAS USA*, 108(21):8645–8650, 2011. 32

[KJMA+14] Z. Kong, A. Jones, A. Medina Ayala, Ebru Aydin G., and Calin 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. 147

[KNP04] M. Kwiatkowska, G. Norman, and D. Parker. Probabilistic symbolic model checking with PRISM: a hybrid approach. *Int. J. Softw. Tools Technol. Transf.*, 6(2):128–142, 2004. 4, 87

[KNP11] M. Kwiatkowska, G. Norman, and D. Parker. PRISM 4.0: Verification of Probabilistic Real-time Systems. In *Proceedings of the 23rd International Conference on Computer Aided Verification*, CAV'11, pages 585–591, Berlin, Heidelberg, 2011. Springer-Verlag. 142

[Kur70] T. G. Kurtz. Solutions of ordinary differential equations as limits of pure jump markov processes. *Journal of Applied Probability*, 7(1):49–58, 1970. 19

[Kur81] T. Kurtz. *Approximation of Population Processes*. CBMS-NSF Regional Conference Series in Applied Mathematics. Society for Industrial and Applied Mathematics, January 1981. 18

[LLW12] J. Ladyman, J. Lambert, and K. Wiesner. What is a complex system? *European Journal for Philosophy of Science*, 3(1):33–67, June 2012. 2

[LMST07] R. Lanotte, A. Maggiolo-Schettini, and A. Troina. Parametric Probabilistic Transition Systems for System Design and Analysis. *Form. Asp. Comput.*, 19:93–109, 2007. 87

[LS14] A. Legay and S. Sedwards. Statistical abstraction boosts design and test efficiency of evolving critical systems. In *Proc. of ISoLA, Part I: 6th International Symposium on Leveraging Applications of Formal Methods, Verification and Validation Technologies for Mastering Change*, volume 8802 of *LNCS*, pages 4–25. Springer, 2014. 147

[LV08] E. Lozes and J. Villard. A spatial equational logic for the applied π-calculus. In *CONCUR*, pages 387–401, 2008. 43

[MBR+12] L. Mari, E. Bertuzzo, L. Righetto, R. Casagrandi, M. Gatto, I. Rodriguez-Iturbe, and A. Rinaldo. Modelling cholera epidemics: the role of waterways, human mobility and sanitation. *Journal of The Royal Society Interface*, 9(67):376–388, February 2012. 7, 9, 107, 108

[Mes08] J. Meseguer. The temporal logic of rewriting: A gentle introduction. In *Concurrency, Graphs and Models*, volume 5065 of *Lecture Notes in Computer Science*, pages 354–382. Springer, 2008. 45

[Min13] T. P. Minka. Expectation Propagation for approximate Bayesian inference. *arXiv:1301.2294 [cs]*, January 2013. arXiv: 1301.2294. 54

[MN04] O. Maler and D. Nickovic. Monitoring temporal properties of continuous signals. In *Proc. of Joint International Conferences on Formal Modeling and Analysis of Timed Systmes, FORMATS 2004, and Formal Techniques in Real-Time and Fault -Tolerant Systems, FTRTFT 2004, Grenoble, France, September 22-24*, volume 3253 of *Lecture Notes in Computer Science*, pages 152–166, 2004. 5, 10, 28, 30, 31, 32, 34, 35, 41, 95, 97, 98

[MNP08] O. Maler, D. Nickovic, and A. Pnueli. Checking temporal properties of discrete, timed and continuous behaviors. In Arnon Avron, Nachum Dershowitz, and Alexander Rabinovich, editors, *Pillars of computer science*. Springer-Verlag, 2008. 10, 28

[MWB+12] P. K. Maini, T. E. Woolley, R. E. Baker, E. A. Gaffney, and S. S. Lee. Turing's model for biological pattern formation and the robustness problem. *Interface focus*, 2012. 114

[NB14] L. Nenzi and L. Bortolussi. Specifying and monitoring properties of stochastic spatio-temporal systems in signal temporal logic. In *8th International Conference on Performance Evaluation Methodologies and Tools, VALUETOOLS 2014, Bratislava, Slovakia, December 9-11, 2014*, 2014. xiii, 8, 11, 89, 165

[NBC+15] L. Nenzi, L. Bortolussi, V. Ciancia, M. Loreti, and M. Massink. Qualitative and Quantitative Monitoring of Spatio-Temporal Properties. In E. Bartocci and R. Majumdar, editors, *Runtime Verification*, number 9333 in Lecture Notes in Computer Science, pages 21–37. Springer International Publishing, 2015. xiii, 8, 11, 89, 165

[NFP98] R. D. Nicola, G. L. Ferrari, and R. Pugliese. Klaim: A kernel language for agents interaction and mobility. *IEEE Trans. Software Eng.*, 24, 1998. 42

[NKL+07] R. D. Nicola, J.P. Katoen, D. Latella, M. Loreti, and M. Massink. Model checking mobile stochastic logic. *Theor. Comput. Sci.*, 382, 2007. 6, 42, 43

[NL04] R. D. Nicola and M. Loreti. Momo: A modal logic for reasoning about mobility. In Frank S. de Boer, Marcello M. Bonsangue, Susanne Graf, and Willem P. de Roever, editors, *FMCO*, volume 3657 of *Lecture Notes in Computer Science*, pages 95–119. Springer, 2004. 42

[NLK+] R. De Nicola, M. Loreti, J.P. Katoen, R. Aachen, M. Massink, and D. Latella. Klaim and its stochastic semantics. 42

[NM07] D. Nickovic and O. Maler. AMT: A property-based monitoring tool for analog systems. In *Formal Modeling and Analysis of Timed Systems*, pages 304–319. Springer, 2007. 30

[Øks03] B. Øksendal. *Stochastic Differential Equations: An Introduction with Applications.* Springer., 2003. 22

[Olv14] P. J. Olver. *Introduction to Partial Differential Equations*. Springer International Publishing, 2014. 114, 146

[OMS13] A. Ocone, A. J. Millar, and G. Sanguinetti. Hybrid regulatory models: a statistically tractable approach to model regulatory network dynamics. *Bioinformatics*, 29(7):910–916, 2013. 22, 80, 81, 82, 147

[ORS10] M. Opper, A. Ruttor, and G. Sanguinetti. Approximate inference in continuous time Gaussian-Jump processes. In *Proc. of NIPS 2010: the 4th Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, 6–9 December*, pages 1831–1839, 2010. 22

[PBXB08] S. Pllana, S. Benkner, F. Xhafa, and L. Barolli. Hybrid Performance Modeling and Prediction of Large-Scale Computing Systems. In *International Conference on Complex, Intelligent and Software Intensive Systems, 2008. CISIS 2008*, pages 132–138, March 2008. 142

[PKT09] R. Phillips, J. Kondev, and J. Theriot. *Physical biology of the cell*. Garland Science, 2009. 130

[Pnu77] Amir Pnueli. The temporal logic of programs. *Foundations of Computer Science, IEEE Annual Symposium on*, 0:46–57, 1977. 3, 29

[RBFS08] A. Rizk, G. Batt, F. Fages, and S. Soliman. On a continuous degree of satisfaction of temporal logic formulae with applications to systems biology. In *Proc. of CMSB 2008, the 6th International Conference on Computational Methods in Systems Biology, Rostock, Germany, October 12–15*, volume 5307 of *Lecture Notes in Computer Science*, pages 251–268, 2008. 5, 29, 33, 57, 84, 87

[Ric97] D. Richardson. How to recognize zero. *Journal of Symbolic Computation*, 24(6):627–645, 1997. 65

[RR08] D. Riley and K. Riley. Simulation of stochastic hybrid systems with switching and reflecting boundaries. In *Simulation Conference, 2008. WSC 2008. Winter*, pages 804–812, December 2008. 22

[RS85] J. H. Reif and A.P. Sistla. A multiprocess network logic with temporal and spatial modalities. *Journal of Computer and System Sciences*, 30(1):41–53, February 1985. 45, 91

[RW06] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006. 11, 47, 48, 50, 54, 68, 74, 78

[SKKS12] N. Srinivas, A. Krause, Sham M. Kakade, and M. W. Seeger. Information-theoretic regret bounds for gaussian process optimization in the bandit setting. *IEEE Transactions on Information Theory*, 58(5):3250–3265, 2012. 8, 11, 47, 51, 52, 68, 69, 124

[Ste02] I. Steinwart. On the influence of the kernel on the consistency of support vector machines. *J. Mach. Learn. Res.*, 2:67–93, March 2002. 50

[TBP+12] M. Tizzoni, P. Bajardi, C. Poletto, J. J. Ramasco, D. Balcan, B. Goncalves, N. Perra, V. Colizza, and A. Vespignani. Real-time numerical forecast of global epidemic spreading: case study of 2009 A/H1N1pdm. *BMC Medicine*, 10(1):165, December 2012. 3

[Tur52a] A. Turing. The chemical basis of morphogenesis. *Phil. Trans. R. Soc. Lond.*, pages 37–72, 1952. 113

[Tur52b] A. M. Turing. The Chemical Basis of Morphogenesis. *Philosophical Transactions of the Royal Society of London B: Biological Sciences*, 1952. 9, 144

[WMR07] Y. F. Wu, E. Myasnikova, and J. Reinitz. Master equation simulation analysis of immunostained bicoid morphogen gradient. *BMC systems biology.*, page 52, 2007. 125, 126, 130, 132, 133

[Wol68] L. Wolpert. The french flag problem: A contribution to the discussion on pattern development and regulation. *Towards a theoretical biology.*, pages 125–133, 1968. 127

[WTA15] L. Wolpert, C. Tickle, and A. M. Arias. *Principles of development*. Oxford university press, 2015. 127

[YKNP04] H. L. S. Younes, M. Z. Kwiatkowska, G. Norman, and D. Parker. Numerical vs. statistical probabilistic model checking: An empirical study. In *Proc. of 2004, the 10th International Conference on Tools and Algorithms for the Construction and Analysis of Systems, Barcelona, Spain, March 29 - April 2*, 2004. 4, 10, 30, 47

[YS06] H. L. S. Younes and R. G. Simmons. Statistical probabilistic model checking with a focus on time-bounded properties. *Information and Computation*, 204(9):1368–1409, 2006. 4, 10, 47

[ZPC10] 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*, HSCC '10, pages 243–252, New York, NY, USA, 2010. ACM. 58