THE UNIVERSITY
of EDINBURGH

Continuous-Time Temporal Logic Specification and Verification for Nonlinear Biological Systems in Uncertain Contexts

Thomas Wright

Doctor of Philosophy

Laboratory for Foundations of Computer Science

School of Informatics

University of Edinburgh

2022

Abstract

In this thesis we introduce a complete framework for modelling and verification of biological systems in uncertain contexts based on the bond-calculus process algebra and the $\mathcal{LBUC}$ spatio-temporal logic. The bond-calculus is a biological process algebra which captures complex patterns of interaction based on *affinity patterns*, a novel communication mechanism using pattern matching to express multiway interaction affinities and general kinetic laws, whilst retaining an agent-centric modelling style for biomolecular species. The bond-calculus is equipped with a novel continuous semantics which maps models to systems of Ordinary Differential Equations (ODEs) in a compositional way.

We then extend the bond-calculus to handle uncertain models, featuring interval uncertainties in their species concentrations and reaction rate parameters. Our semantics is also extended to handle uncertainty in every aspect of a model, producing non-deterministic continuous systems whose behaviour depends either on *time-independent* uncertain parameters and initial conditions, corresponding to our partial knowledge of the system at hand, or *time-varying* uncertain inputs, corresponding to genuine variability in a system's behaviour based on environmental factors.

This language is then coupled with the $\mathcal{LBUC}$ spatio-temporal logic which combines Signal Temporal Logic (STL) temporal operators with an uncertain context operator which quantifies over an uncertain context model describing the range of environments over which a property must hold. We develop model-checking procedures for STL and $\mathcal{LBUC}$ properties based on verified signal monitoring over flowpipes produced by the Flow* verified integrator, including the technique of *masking* which directs monitoring for atomic propositions to time regions relevant to the overall verification problem at hand. This allows us to monitor many interesting nested contextual properties and frequently reduces monitoring costs by an order of magnitude. Finally, we explore the technique of *contextual signal monitoring* which can use a single Flow* flowpipe representing a functional dependency to complete a whole tree of signals corresponding to different uncertain contexts. This allows us to produce refined monitoring results over the whole space and to explore the variation in system behaviour in different contexts.

Acknowledgements

I would firstly like to thank my supervisor Ian Stark for his support, feedback, and encouragement throughout my PhD. I would also like to thank my secondary supervisor Paul Jackson for suggesting the use of Flow* and for many interesting discussions and feedback concerning formal methods for continuous systems. Thanks also go to Kristjan Liiva for providing insight on the inner workings of Flow* and for his patches to Flow*'s interval arithmetic code, and to Andrew Sogokon for discussions of his work on continuous systems verification.

I would also like to thank the PEPA club and its attendees for providing a great environment to meet and discuss language-based modelling and verification including Chris Banks, Jane Hillston, Stephen Gilmore, Vashti Galpin, Ludovica Vissat, Anastasis Georgoulas, and Paul Piho. Furthermore, I would like to thank everyone in LFCS whose discussions kept my time in the department interesting including but by no means limited to Simon Fowler, Daniel Hillerström, Wen Kokke, James McKinna, and Sam Lindley. Thanks also go to Murray Cole for his consistent support for PPAR CDT students, especially during the Covid-19 pandemic.

Finally, I would like to thank the anonymous reviewers of my papers for their useful feedback and corrections. I would also like to thank Juliet Cooke, Jos Gibbons, and Julie Wright for reading and providing feedback on drafts of this thesis.

This work was supported in part by the EPSRC Centre for Doctoral Training in Pervasive Parallelism, funded by the UK Engineering and Physical Sciences Research Council (grant EP/L01503X/1) and the University of Edinburgh.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

An early version of the bond-calculus was first introduced in the author's MRes dissertation [358]. The semantics included in Chapter 3 is substantially different than that featured therein in order to achieve compositionality whilst the presentation of the language has significantly evolved. During the course of this thesis, the results from Chapter 4 have been published in the paper [359], whilst parts of Chapters 6 and 7 have been published in [360]. The extension of the bond-calculus to uncertain models is entirely new, as is $\mathcal{LBUC}$.

(*Thomas Wright*)

Lay Summary

In this thesis we introduce a complete framework for modelling the behaviour of biological models under uncertainty. We introduce the bond-calculus, a high-level modelling language for biological systems. This language is capable of handling uncertainty in both the initial reactant concentrations of a model and its reaction rate parameters. We have developed tools for converting bond-calculus models to systems of differential equations, enabling simulation of models and worst-case analysis of their behaviour under uncertainty.

This is coupled with the *Logic of Behaviour in Uncertain Contexts (*$\mathcal{LBUC}$*)*, a specification language for describing properties which we expect a model to satisfy. These properties include temporal properties, specifying the behaviour of agents over time, and contextual properties, specifying the types of uncertain environments or *contexts* in which we expect these properties to hold. This makes it possible to express a wide range of interesting biological properties including the robustness of properties under uncertainties in concentration, reaction rate parameters, and contexts. Moreover, we are able to specify experimental protocols which observe the response of the model to a range of different external interventions.

Throughout this thesis we develop a range of algorithms to check whether a bond-calculus model satisfies a $\mathcal{LBUC}$ property. These include exact formal verification methods which offer guaranteed handling of uncertainty and numerical errors as well as techniques to explore the variation in a model's behaviour for different choices of parameter values.

Our methods have been implemented in an interactive notebook environment, allowing practitioners to experiment with different models and properties. We evaluate these methods and tools through a range of biological models including gene regulatory networks and an ecological model of predator-prey role reversal in a marine environment.

Table of Contents

Chapter 1

Introduction

Over the last two decades there has been increasing interest in applying formal modelling techniques originally developed for software systems to modelling and analysing the behaviour of biological systems. This led first to the use of formal languages such as process algebras and rule-based languages to model components of biochemical systems, and then to the development of increasingly sophisticated modelling frameworks to capture distinctive features of biological systems such as continuous agent concentrations, stochasticity, and complex reaction rate laws. Equally important have been efforts to analyse the behaviour of models starting with simulation of model behaviour and expanding to automated verification of model behaviour against temporal logic properties.

Many aspects of biological systems still pose significant challenges to formal modelling and analysis, motivating the development of new languages, semantics, and verification methods. Firstly, the patterns of communication in biological systems are remarkably diverse even compared to computer systems, so a general purpose framework should be capable of succinctly describing intermolecular reactions between multiple molecules, allosteric interactions, regulatory interactions between genes, intercellular communication, and even interactions between whole organisms in an ecological setting. Secondly, such a framework should be capable of determining the quantitative rates of reactions in a compositional manner whilst separating the reactive behaviour of agents from their quantitative interaction capacities. Thirdly, unlike the discrete non-deterministic systems which form the traditional domain of formal modelling and verification, biological systems usually involve a mixture of continuous evolution and discrete jumps, taking us into the challenging world of continuous and hybrid systems verification. And finally, one usually has only partial knowledge about the system being modelled due to uncertainty arising from any and all of: the concentrations of agents and reaction rates, the system's ever

varying environment, and the simplifications and omissions required to build tractable models.

In this thesis we present our approach to formal modelling and verification of biological systems under uncertainty, based on combining a novel biological process algebra, the bond-calculus, together with a spatio-temporal specification logic, $\mathcal{LBUC}$, for the temporal behaviour of bond-calculus models under uncertainty. To verify these properties, we develop a range of exact model-checking techniques including: verified signal monitoring over flowpipes produced by the Flow* verified integrator [110], property-directed monitoring using *masks* to restrict our efforts to relevant time regions, and *contextual* monitoring and refinement of spatio-temporal signals for functional dependencies.

1.1 The bond-calculus

We propose the bond-calculus as a high-level modelling formalism for biological systems at the molecular and network levels focused around chemical species whose interactions are governed by affinity patterns specifying multi-way, multi-level interactions.

The bond-calculus adopts a process algebraic approach, modelling biochemical species as agents which communicate at reaction sites whilst the quantitative interaction capacities of these sites are specified separately via pattern matching against a network of *affinity patterns*. That is, a bond-calculus model $\mathcal{M} \triangleq (\Pi_0, \mathcal{A})$ consists of two parts: a *mixture*

$$\Pi_0 \triangleq [X_1]\, X_1 \parallel \ldots \parallel [X_n]\, X_n$$

consisting of real-valued concentrations $[X_1], \ldots, [X_n] \in \mathbb{R}$ of different biochemical species represented as process-algebraic processes $X_1, \ldots, X_n$ and an *affinity network*

$$\mathcal{A} \triangleq \{\, \boldsymbol{\gamma}_1 \mathbin{@} \mathrm{L}_1, \ldots, \boldsymbol{\gamma}_m \mathbin{@} \mathrm{L}_\mathrm{m} \,\}$$

which defines the forms of interactions which can occur. Each class of interactions is specified by an *affinity pattern* $\boldsymbol{\gamma}$ @ L which combines a multi-level pattern $\boldsymbol{\gamma} \triangleq (p_{1,1} | \ldots | p_{1,j_1}) \parallel \ldots \parallel (p_{k,1} | \ldots | p_{k,j_k})$, which specifies a combination of compatible interaction sites $p_{i,j}$, and a general kinetic law L, which specifies the quantitative rate at which this interaction proceeds. The bond-calculus is coupled with a compositional continuous semantics which translates models into systems of Ordinary Differential Equations (ODEs); the full syntax and semantics of the bond-calculus are introduced in Chapter 3.

This communication model was originally conceived as an extension of the Continuous π-calculus' [234] binary affinity network based communication mechanism, which

encodes low-level molecular bonding via passing of mobile site names. The bond-calculus incorporates multiway interactions and general kinetic laws to enable high-level modelling of biological networks whilst multi-level pattern matching allows us to model low-level molecular reactions including contextual interactions between different molecular sites. Affinity patterns can be seen as an intermediate abstraction between purely agent-based languages, where the dynamics are driven by agents' use of predefined actions or names, and purely rule-based languages, where agents define molecular structure and the dynamics are driven by reaction rules. This gives us a rather different view of interactions leading to distinctive modelling styles at the molecular and network levels. The bond-calculus' compositional continuous semantics also allows us to independently explore composition of processes and composition of affinity networks.

1.2 $\mathcal{LBUC}$: A Logic for Uncertain Models in Uncertain Contexts

Whilst most biological modelling frameworks require initial species parameters and reaction rate parameters to be fully specified, in real biological systems the existing literature and imperfect experimental data can only give us partial knowledge of these quantitative model parameters. Moreover, in order to scalably model the variety and complexity of living systems, models must abstract over wide ranges of parameter values arising from variations between instances or states of a system, variations in the external environment of a system, or from system components omitted from a model due their complexity. This motivates us to consider uncertain bond-calculus models $\widehat{\mathcal{M}} \triangleq \left(\widehat{\Pi}_0, \widehat{\mathcal{A}}\right)$ featuring interval uncertainty in their initial conditions

$$\widehat{\Pi}_0 \triangleq [a_1, b_1]X_1 \parallel \ldots \parallel [a_n, b_n]X_n$$

and in their affinity network $\widehat{\mathcal{A}}$ via kinetic laws $\mathrm{L}_{[a_1,b_1],\ldots,[a_m,b_m]}$ with interval rate parameters $[a_1, b_1], \ldots, [a_m, b_m]$. These uncertain models can either directly represent uncertainty in the system itself, or the range of uncertain environments or *contexts* which indirectly influence a system's behaviour.

This moves us from considering the deterministic behaviour of a closed continuous system to nondeterministic and contextual behaviour, motivating the development of specification languages capable of describing this contingent contextual behaviour under uncertainty and of formal methods capable of verifying that a system's behaviour

conforms to these specifications. We do this via the Logic of Behaviour in Uncertain Contexts ($\mathcal{LBUC}$). This follows an approach introduced by Banks and Stark's $\mathcal{LBC}$ logic [19] in combining the temporal operators of the Signal Temporal Logic (STL) [253] with a form of spatial guarantee operator

$$\mathcal{C} \triangleright \varphi,$$

inspired by Cardelli and Gordon's spatial logics for concurrent systems [88, 103], which quantifies a property φ over a *context* $\mathcal{C} \triangleq \left(\widehat{\Pi}, \widehat{\mathcal{A}}\right)$ consisting of an uncertain bond-calculus model. Such a property captures the notion of *contextually robust satisfaction*; the satisfaction of φ is invariant under perturbation by the class of context models defined by $\mathcal{C}$.

The interactions between context operators and temporal operators make it possible to express *spatio-temporal experiments* which study the behaviour of the system by observing its reaction to timed perturbations. As a first step, we are able to apply STL temporal specifications such as $\mathcal{F}_{[a,b]}\,\varphi$ or $\mathcal{G}_{[a,b]}\,\varphi$ to uncertain bond-calculus models, allowing us to observe the behaviour of an uncertain system in isolation. Then the use of context operators makes it possible to express timed perturbations such as

$$\mathcal{G}_{[a,b]}(\mathcal{C} \triangleright \varphi)$$

which specifies that φ should hold immediately after the introduction of any bond-calculus model consistent with $\mathcal{C}$ at any time point between a and b time units in the future. The bond-calculus lets us separately consider contexts of the form $\widehat{\Pi} \triangleright \varphi$ which instantaneously introduce uncertain concentrations of new reactants into the system and *affinity network contexts* of the form $\widehat{\mathcal{A}} \triangleright \varphi$ which introduce new affinity patterns to the system, inducing additional reactions in the system at time-varying uncertain rates. These timed perturbations are similar to timed discrete events or hybrid system jumps, and are able to describe discrete jumps in system state caused by either external environmental changes or experimental interventions. However, $\mathcal{LBUC}$ allows arbitrary nesting of contextual and temporal properties, allowing for much richer logical properties which express contingent contextual behaviour. In this thesis we will explore some of these possibilities as well as developing automated methods for $\mathcal{LBUC}$ verification.

1.3 Verified Monitoring over Flow* flowpipes

Much of the work on continuous-time temporal logic verification has focused on signal monitoring over approximate numerical trajectories building upon the approach of Maler and

Nickovic's Signal Temporal Logic (STL) [253]. This pragmatic approach to verification has led to the development of powerful monitoring algorithms for a wide range of interesting temporal and spatio-temporal logics for quantitative systems. However, numerical trajectories give an inherently imperfect view of a system's dynamics so are unable to guarantee monitoring results have not been influenced by numerical errors. Moreover, individual numerical trajectories do not take into account the many uncertainties which arise when modelling real systems. Indeed, in an uncertain model, there are often uncountably many trajectories which are equally consistent with our knowledge of the system, each of which could be essential to the overall truth of a proposition. Finite length numerical trajectories are also unable to verify many interesting unbounded temporal properties which describe the long term behaviour of a system. These factors motivate us to consider exact methods which soundly account for all possibilities, allowing us to perform worst-case analysis of a system's behaviour under uncertainty.

Flow* [110] is a leading formal verification tool which uses preconditioned Taylor model flowpipes [45, 251] to tightly over-approximate the dynamics of continuous and hybrid systems over time. This offers a powerful approach to formal verification under uncertainty, with Flow*'s sophisticated flowpipe representation allowing us to track our uncertain knowledge of the system state at each point of time. Flow* has historically been limited to verifying a restricted class of reach-avoidance properties, calling for the development of new methods to expand its scope to general purpose temporal logic verification.

We develop techniques for exact verification of STL and $\mathcal{LBUC}$ properties based on verified signal monitoring over Flow* flowpipes. We employ interval-based monitoring methods to translate the uncertain picture of the system state contained in a Flow* flowpipe into a Three-Valued Signal representing our uncertain knowledge of the truth of propositions over time. By adopting an adaptive symbolic monitoring approach which tightly integrates the monitoring process with Flow*'s flowpipe representation, we are able to obtain more precise monitoring results for complex atomic propositions compared to existing "black-box" verified monitoring approaches which handle flowpipes uniformly as interval functions [208].

We subsequently introduce a number of techniques to significantly improve the effectiveness and scope of monitoring. We propose how three-valued signals may be extended to verify unbounded-time properties using suitable invariants, following an approach introduced by Sogokon, Jackson, and Johnson [333]. We then enable property-directed monitoring by augmenting the usual bottom-up signal monitoring with top-down com-

putation of masks, a special type of signal representing the regions of interest in a given proposition. This allows us to direct the monitoring of atomic propositions to only those time regions relevant to the overall verification problem, avoiding expensive symbolic flowpipe operations over much of the time domain. Finally, we develop contextual monitoring which uses a Flow* flowpipe to represent a functional dependency of a system's behaviour on its initial conditions. This allows us to efficiently monitor spatio-temporal signals tracking the truth of propositions over the "context space" of different contexts to a system, allowing us to give refined verification results and to explore variations in system behaviour over the context space.

1.4 Contributions

The main contributions of this thesis are the following:

1. **A new compositional semantics and ODE extraction algorithm for the bond-calculus** This thesis includes a revised semantics and implementation of the bond-calculus, which was first introduced in the author's Masters dissertation [358]. The language's original graphical communication mechanism based on affinity hypergraphs has been replaced with a simpler formulation based on pattern matching. We have also introduced a new semantics for the language to make it possible to extract ODEs in a compositional way. This compositionality is important in the implementation of context operators which dynamically introduce new species into an existing system, and our new semantics shows how it may be achieved in the presence of non-mass action rates laws. The semantics of the language is also extended to allow interval uncertainty in any part of a model.

2. **The $\mathcal{LBUC}$ logic and its formal semantics** We introduce $\mathcal{LBUC}$, a distinctive spatio-temporal logic combining STL temporal operators with an uncertain context operator $\widehat{\mathcal{C}} \triangleright \varphi$. This leads to a comprehensive framework to rigorously handle uncertainty in all aspects of a system including the initial concentrations and reaction rates of the underlying model and the uncertain environment in which it operates. As we will explore in Section 5.3, the addition of uncertain contexts makes it possible to capture a wide range of additional interesting biological behaviours, supporting its role as a core logic for modelling behaviour under uncertainty.

3. **An exact model-checking algorithm for $\mathcal{LBUC}$ based on verified monitoring of Flow* flowpipes under *masks*** We introduce an algorithm to verify the

conformance of bond-calculus models to $\mathcal{LBUC}$ properties based on verified monitoring. We also introduce masks, which make it possible to restrict the monitoring of atomic propositions to those time regions required for the property at hand. This addresses a limitation of existing attempts to extend continuous and hybrid systems reachability computation to temporal logic verification, as our approach introduces the first STL monitoring algorithm over flowpipes we are aware of which is *property-directed* in the sense that it is able to adapt its verification algorithm for atomic propositions based on their relevance to the overall property at hand. We demonstrate several cases in which this substantially increases monitoring performance, especially in the case of context operators.

4. **Adaptive symbolic extensions of these model-checking techniques to handle large uncertain contexts and spatially inhomogeneous behaviour** We introduce an application of spatio-temporal signal monitoring to verifying $\mathcal{LBUC}$ properties involving large uncertain regions. Our previous model-checking algorithm required constructing a Flow* flowpipe and giving a monitoring result covering the whole range of uncertainty, limiting its applicability to relatively small ranges of uncertainty. We use trees of successive subdivisions of a context to monitor spatio-temporal signal trees for properties over the *context space* generated by an uncertain context. We demonstrate how, when Flow* succeeds in flowpipe construction, its symbolic flowpipe format makes it possible to compute a signal tree covering all subregions of a given context from a single reachability computation, and introduce a hybrid algorithm, switching from repeated flowpipe computation to symbolic subdivision as necessary to handle a given context. These techniques are also combined with masks to introduce a form of spatio-temporal masking.

5. **Case studies and benchmarks demonstrating these techniques** We also investigate example models and properties to demonstrate the applicability and performance of the techniques developed in this thesis. We investigate bond-calculus modelling styles for gene regulation at both the molecular level and the network level, which exemplify the difference in our modelling style from both agent-based process algebra models and rule-based models. Furthermore, we apply our model-checking techniques to an ecological model of Predator-Prey Role Reversal and a 9-dimensional Genetic Oscillator benchmark model, demonstrate our ability to verify interesting STL and $\mathcal{LBUC}$ properties, and evaluate monitoring performance.

A key secondary component of this work involved implementing the techniques of

this thesis. This firstly involved extending the bond-calculus implementation (originally developed in the author's Masters project) to handle uncertainty (by producing parametric systems of ODEs with interval rates and initial condition) and Chemical Reaction Network extraction. This then involved implementing the $\mathcal{LBUC}$ as an embedded DSL in a Python/Jupyter interactive environment, and implementing all of the techniques discussed herein on top of Flow*'s C++ API. These implementations enable concrete tests and benchmarks of our methods and have frequently influenced their development.

1.5 Thesis Structure

The structure of this thesis is as follows:

Chapter 1: Introduction In this chapter we introduce the aims and content of this thesis.

Chapter 2: Background and Literature In this chapter we outline background material and discuss the literature relevant to this thesis. Firstly in Section 2.1 we introduce necessary mathematical definitions and notation. Then in Section 2.2 we introduce existing process-algebraic formalisms relevant to this thesis and their distinct modelling domains and abstractions. In Section 2.3 we discuss existing temporal and spatio-temporal logics. Finally, in Section 2.4 we discuss existing approaches to continuous and hybrid systems verification under uncertainty and introduce Flow* and Taylor models.

Chapter 3: Bond-Calculus In this chapter we introduce the syntax and semantics of the bond-calculus process algebra. Firstly, in Section 3.2 we introduce the formal syntax of the language. In Section 3.3 we introduce its semantics by defining the transition matrix of a mixture and the concentration dependent rate vector of an affinity network. Finally, in Section 3.4 we outline the continuous dynamics of bond-calculus models as a vector field, enabling extraction of differential equations or stochastic chemical reaction models.

Chapter 4: Modelling Patterns of Gene Regulation in the Bond-Calculus In this chapter we explore different styles of bond-calculus modelling for gene regulation. Firstly, in Section 4.1 we revisit Kuttler and Niehren's classic process algebra model of cooperative regulation at the λ-switch and see how the bond-calculus' multiway communication and

mobility lead to a direct model of the molecular interactions in the switch. Then in Section 4.2 we move to the network level and see how we may model gene regulatory networks in a systematic way, combining a general purpose agent-based model of transcription and translation with an affinity network encoding the regulatory interactions in a particular network.

Chapter 5: Uncertain Models and Uncertain Contexts In this chapter we move beyond simulating fixed bond-calculus models to verifying their behaviour in uncertain time-varying contexts. Firstly in Section 5.1 we introduce uncertain bond-calculus models, their semantics in terms of bond-model trajectories, and how we perform interval ODE extraction. Then in Section 5.2 we introduce the $\mathcal{LBUC}$ logic and its semantics over uncertain bond-calculus models. Finally, in Section 5.3 we explore how different types of uncertainty in biological systems may be modelled in $\mathcal{LBUC}$.

Chapter 6: Verified monitoring with Flow* In this chapter we introduce a model-checking algorithm for $\mathcal{LBUC}$ properties of uncertain bond-calculus models based on verified monitoring over Flow* flowpipes. Firstly, in Section 6.1 we introduce our monitoring algorithm for STL properties, and see how the symbolic nature of Flow* flowpipes may be exploited to increase the precision and efficiency of monitoring. In Section 6.2 we extend this to a full model-checking algorithm for $\mathcal{LBUC}$ by introducing a monitoring algorithm for context operators. In Section 6.3 we discuss how our monitoring algorithm can make use of continuous invariants to verify unbounded temporal operators. Then in Section 6.4 we apply it to a genetic oscillator and investigate monitoring performance and in Section 6.5 we demonstrate the application of our monitoring algorithm to an ecological model of predator-prey role reversal.

Chapter 7: Masks In this chapter we introduce our masked monitoring algorithm. In Section 7.1 we define masks and their operations. In Section 7.2 we introduce monitoring contexts, which capture the context of a STL formula within the overall monitoring process, and define the sufficiency and optimality of a mask for a given context. In Section 7.3 we introduce the masks for each operator of STL, which we prove to be optimal and sufficient, whilst in Section 7.4 we show a mask may be applied to monitoring an atomic proposition. In Section 7.5 we extend this to a full masked monitoring algorithm for $\mathcal{LBUC}$ including context operators. Finally, in Section 7.6 we explore the performance impact of masked monitoring for a range of properties.

Chapter 8: Contextual Signals In this chapter we introduce contextual monitoring, which extends signal monitoring to take into account variations in the truth of a proposition over different regions of an uncertain context. In Section 8.1 we introduce *contextual signals*, which add a spatial domain to signals to capture contextual uncertainty, and their concrete representation as *signal trees*. In Section 8.2 we explore different methods to monitor signal trees based on Flow* flowpipes and define a signal tree monitoring algorithm for $\mathcal{LBUC}$. In Section 8.3 we introduce *mask trees*, allowing us to combine masking and contextual monitoring. Finally in Section 8.4 we demonstrate contextual monitoring as applied to our predator-prey role reversal model and explore the performance of contextual monitoring.

Chapter 9: Implementation Overview In this chapter we give a brief overview of our tools implementing the techniques developed in this thesis and their usage. In Section 9.1 we describe our implementation of the bond-calculus language. In Section 9.2 we describe our integration of the bond-calculus and $\mathcal{LBUC}$ with Sagemath and Python and their use in an interactive browser-based Jupyter environment. In Section 9.3 we describe our interface to Flow*'s C++ API and our use of Cython. Finally, in Section 9.4 we briefly describe our methodology for benchmarking and testing.

Chapter 10: Conclusion This chapter forms the conclusion of the thesis. In Section 10.1 we discuss and evaluate the techniques developed throughout the thesis and consider their relationship to related work. Then, finally, in Section 10.2 we discuss future work.

Chapter 2

Background and Literature

In this chapter we review the relevant background and literature for this thesis. In Section 2.1 we introduce some necessary definitions and notation. In Section 2.2 we survey process algebraic modelling formalisms for biological systems. In Section 2.3 we introduce temporal and spatial logic and survey their application to biological and quantitative systems. Finally, in Section 2.4 we introduce continuous and hybrid systems verification including the Flow* verified integrator [110].

2.1 Mathematical Preliminaries

In this section we briefly recall some mathematical definitions and notation concerning three-valued logic, interval arithmetic, and order theory which are used throughout this thesis.

2.1.1 Three-Valued Logic

Kleenian three-valued logic extends traditional Boolean logic whose propositions must be either true (**T**) or false (**F**) by adding a third possibility unknown (**U**) which expresses our uncertainty as to the truth or falsity of a proposition. The operators of three-valued logic match those of Boolean logic, except that **U** must soundly account for the possibility of either Boolean truth value giving the truth-tables,

	$\neg$
T	**F**
U	**U**
F	**T**

$\wedge$	**T**	**U**	**F**
T	**T**	**U**	**F**
U	**U**	**U**	**F**
F	**F**	**F**	**F**

$\vee$	**T**	**U**	**F**
T	**T**	**T**	**T**
U	**T**	**U**	**U**
F	**T**	**U**	**F**

$\Rightarrow$	**T**	**U**	**F**
T	**T**	**U**	**F**
U	**T**	**U**	**U**
F	**T**	**T**	**T**

The above definition of implication follows the equivalence $x \Rightarrow y \equiv \neg x \vee y$ as in Boolean logic.

Remark 2.1.1. Whilst natural, this Kleenian definition of implication has the somewhat surprising consequence that $\mathbf{U} \Rightarrow \mathbf{U} \equiv \mathbf{U}$. There are alternative three-valued logics such as Łukasiewicz logic in which $\mathbf{U} \Rightarrow \mathbf{U} \equiv \mathbf{T}$, each of which corresponds to a different interpretation of the third value $\mathbf{U}$.

We denote the set of Boolean truth values by $\mathbb{B} \triangleq \{\mathbf{T}, \mathbf{F}\}$, and the set of three-valued truth values by $\mathbb{T} \triangleq \{\mathbf{T}, \mathbf{U}, \mathbf{F}\}$.

2.1.2 Interval Arithmetic

Interval arithmetic provides a way of handling uncertainty in numerical computations by extending the rules of arithmetic on real numbers to intervals representing our range of uncertainty in a given numerical value. The definitions in this section are mostly standard and implemented in many popular libraries; a standard reference for interval arithmetic and interval numerical methods is [271] whilst [256] formalises extended interval arithmetic.

We define interval arithmetic over the space $\mathbb{IR}$ of closed real intervals $I = \left[l_I, u_J \right] \triangleq \left\{ x \in \mathbb{R} : l_I \leq x \leq u_I \right\}$ with lower and upper endpoint $l_I, u_I \in \overline{\mathbb{R}}$ in the extended real line $\overline{\mathbb{R}} \triangleq \mathbb{R} \cup \{-\infty, \infty\}$. Whilst this in general allows unbounded intervals $(-\infty, b]$, $[a, \infty)$, etc, we will often restrict our attention to the *bounded intervals* I having finite endpoints $l_I, u_I \in \mathbb{R}$. Then arithmetic operations on intervals are defined by

$$\begin{aligned} I + J &\triangleq \left\{ \, x + y \;\middle|\; x \in I, y \in J \, \right\} = \left[\, l_I + l_J, u_I + u_J \, \right] \\ I - J &\triangleq \left\{ \, x - y \;\middle|\; x \in I, y \in J \, \right\} = \left[\, l_I - u_J, u_I - l_J \, \right] \\ IJ \quad &\triangleq \left\{ \, xy \quad \;\middle|\; x \in I, y \in J \, \right\} = \left[\, \min S, \max S \, \right] \end{aligned}$$

where $S = \{ \, l_I l_J, l_I u_J, l_J u_I, u_I u_J \, \}$, whilst division is partially defined by

$$I/J \quad \triangleq \left\{ \, x/y \quad \;\middle|\; x \in I, y \in J \, \right\} = x \left[\, 1/u_J, 1/l_J \, \right]$$

whenever $0 \notin J$.

We can also define the lattice operations of interval intersection and interval hull by

$$\begin{aligned} I \cap J &= \left[\, \max(l_I, l_J), \min(u_I, u_J) \, \right] \\ I \sqcup J &= \left[\, \min(l_I, l_J), \max(u_I, u_J) \, \right] \end{aligned}$$

and the *inner subtraction* of intervals by

$$I \dot{-} J = \bigcap_{j \in J} (I - j) = \mathbb{R} \setminus (\mathbb{R} \setminus I - J) = \Big[l_I - l_J, u_I - u_J \Big].$$

giving an inner-approximation of the result of subtraction[1].

Given a real-valued function $f : D \to \mathbb{R}$ over interval domain D, an interval-valued function $F : D \to \mathbb{R}$ is an *interval extension* of f is $f(x) \in F(X)$ for every point $x \in D$ and interval $X \subseteq D$ such that $x \in D$.

One limitation of standard interval arithmetic is that the quotient I/J is rather narrowly defined since if we allow $0 \in J$ the set of quotients x/y for $x \in I, y \in J$ will often not be connected and hence will not form an interval. This motivates us to define *multi-interval arithmetic* [340] which allows us to operate simultaneously on different interval branches of a set.

Definition 2.1.2. A *multi-interval* is a set $I = I_1 \cup \ldots \cup I_n$ formed as a finite union of disjoint interval components $I_1, \ldots, I_n$.

Each of the standard interval operations extend to multi-intervals distributively, so, for example, $(I_1 \cup \ldots \cup I_n) + (J_1 \cup \ldots \cup J_m) = \bigcup_{i=1}^{n} \bigcup_{j=1}^{m} (I_i + J_j)$. Then we can extend the definition of the quotient to handle zeros so

$$I/J \triangleq \begin{cases} I/J & \text{if } 0 \notin J \\ \Big(-\infty, l_I/u_J \Big] & \text{if } 0 = l_J < u_J \\ \Big(-\infty, l_I/u_J \Big] \cup \Big[l_I/u_J, \infty \Big) & \text{if } l_I < 0 < u_J \\ \Big[l_I/u_J, \infty \Big) & \text{if } l_I = 0 = u_J \\ \Big(-\infty, \infty \Big) & \text{otherwise} \end{cases}$$

where we still exclude the case of $l_J = 0 = u_J$.

The move to multi-intervals is especially important in interval root finding methods, as we often need to handle intervals which may contain multiple roots of a function. For example, the *extended interval Newton-Raphson method* [208, 271, 340] shown in Algorithm 1 is an extension of the standard Newton-Raphson root finding method using multi-interval arithmetic and is able to simultaneously bound all roots of a continuously differentiable function f based on interval extensions F, F' of f and its derivative f' respectively.

[1]This inner-approximated subtraction operation corresponds to applying the normal inverse Minkowski sum to their complements and variants have been studied in extended versions of interval arithmetic [221, 256, 257]

$\texttt{roots}(F, F', X, X_{\text{old}}, \tau) \triangleq$

if $\texttt{diameter}(X_{\text{old}}) - \texttt{diameter}(X) < \tau$ **then**

  return X

else

  $x := \texttt{midpoint}(X)$

  $X' := \left(x - \frac{F(x)}{F'(X)}\right) \cap X$

  return $\bigcup_{Y \in \texttt{components}(X')} \texttt{roots}(F, F', Y, X, \tau)$

Algorithm 1: Extended Interval Newton-Raphson method to compute a multi-interval enclosing all roots of an interval function F with derivative enclosure F' in interval X. The tolerance parameter τ determines the minimum difference in enclosing interval before the algorithm stops recursing whilst X_{old} contains the interval bound from the previous step.

We are also interested in n-dimensional vectors of intervals $\mathbf{I} \in \mathbb{IR}^n$, representing n-dimensional orthogonal parallelopipeds, for which we define the vector interval operations

$$(\mathbf{I} + \mathbf{J})_j \triangleq \mathbf{I}_j + \mathbf{J}_j \qquad (I\mathbf{J})_j \triangleq I\mathbf{J}_j$$
$$(\mathbf{I} \cap \mathbf{J})_j \triangleq \mathbf{I}_j \cap \mathbf{J}_j \qquad (\mathbf{I} \sqcup \mathbf{J})_j \triangleq \mathbf{I}_j \sqcup \mathbf{J}_j.$$

Then a vector interval function $\mathbf{F} : \mathbf{D} \to \mathbb{IR}^n$ is an interval extension of a vector function $\mathbf{f}$ if $\mathbf{f}(\mathbf{x}) \in \mathbf{F}(\mathbf{X})$ for any $\mathbf{x} \in \mathbf{D}$ and $\mathbf{X} \in \mathbb{IR}^m$ such that $\mathbf{x} \in \mathbf{X}$.

2.1.3 Orders and Trees

We firstly recall some basic notions from order theory. A *partially ordered set* $(S, \leq)$ consists of a set S together with a relation $\leq$ which is *reflexive* ($x \leq x$ for all $x \in S$), *anti-symmetric* ($(x \leq y) \wedge (y \leq x) \implies (x = y)$ for all $x, y \in S$), and *transitive* ($(x \leq y) \wedge (y \leq z) \implies (x \leq z)$ for all $x, y, z \in S$). A *totally ordered set* is partially ordered set $(S, \leq)$ such that either $x \leq y$ or $y \leq x$ for all $x, y \in S$. A totally ordered set $(S, \leq)$ is *well-ordered* if every subset of S has a least element. We will need the following two important orderings on truth values:

Example 2.1.3 (Truth Order)**.** The *truth order* is defined by $x \leq y$ iff $x \vee y \equiv y$. Under this ordering both $\mathbb{B}$ and $\mathbb{T}$ are totally ordered and $\mathbf{F} \lneq \mathbf{U} \lneq \mathbf{T}$.

Example 2.1.4 (Information Order)**.** The set $\mathbb{T}$ of three-valued truth values is a partially ordered set under the *information order* or *approximation order* [36] defined by $\mathbf{T} \sqsupseteq \mathbf{U} \sqsubseteq \mathbf{F}$.

We define the *down-set* $x{\downarrow}$ of an element $x \in S$ as $x{\downarrow} = \left\{y \in S : y \leq x\right\}$ and the *up-set* $x{\uparrow}$ of $x \in S$ as $x{\uparrow} = \left\{y \in S : x \leq y\right\}$.

The order-theoretic definition of a tree is then given by,

Definition 2.1.5. A *tree* is a partial order $(T, \leq)$ which has a least element root(T) and for which every down-set $x{\downarrow}$ is well-ordered.

In particular, we will be interested in *trees of sets* $(\mathcal{T}, \supseteq)$, which consist of a collection of sets $\mathcal{T}$ which is a tree under the superset relation $\supseteq$. We note that in trees of sets the order is somewhat counter-intuitive since we consider the root to be the largest set.

Other tree related concepts may then be defined order-theoretically. Thus the descendants of a node x (inclusively of x itself) in the tree are given by the up-set $x{\uparrow}$ whilst the ancestors of x are given by the down-set $x{\downarrow}$. A node y is a child of x if $x < y$ and for any z such that $y \leq z \leq x$, we must have either $x = z$ or $z = y$, and we denote the set of all children of x as children(x). Finally, a leaf node of $(S, \leq)$ is any node $x \in S$ such that children$(x) = \varnothing$, and we denote the set of all leaf nodes of S by leaves(S).

2.2 Formal Modelling Languages for Biology

Whilst formal languages have played a key role in computer science as abstract models for computation and concurrency, they have also come to increasing prominence in computational biology as formalisms for modelling different types of interactions in biological systems. In contrast to lower-level mathematical formalisms such as differential equations which directly encode system dynamics, the formal language-based approach aims first to model the intensional structure and interactions of agents, and then to provide a choice of semantics which may be applied to translate models into corresponding mathematical models at different levels of abstraction. These semantics can then be used as the basis for simulating and analysing system behaviour.

Over the last two decades a profusion of formal languages have been applied to biological modelling, whose diversity mirrors the living systems they aim to model. In this section we will survey this ecosystem, with an emphasis on the process-algebraic languages and modelling abstractions which have influenced the bond-calculus' approach. To this end we will first introduce some of the main modelling formalisms in Section 2.2.1, before focusing on specific developments for different domains of biological modelling in Section 2.2.2, and then take stock of the distinct abstractions which these frameworks have brought to biochemical modelling in Section 2.2.3.

2.2.1 Modelling Formalisms

Process algebras such as CCS [265], CSP [201], ACP [44], and the π-calculus [269, 270] first emerged as formal frameworks for modelling reasoning about interactions between concurrent processes in computer systems. Each employs a different model of concurrent communication reflecting the diversity of patterns of interaction in concurrent systems, with CSP modelling multiway coordination on shared actions and CCS modelling binary communication between matching names x and conames $\bar{x}$ representing two halves of a communication channel, whilst the π-calculus extends this to mobile communication, allowing processes to send names along channels. The application of process algebra to modelling biological systems has revealed an even greater range of patterns of interaction in biological systems, leading to a wide range of different languages and communication mechanisms. These languages have also featured a range of semantics ranges from discrete non-deterministic qualitative semantics to stochastic or discrete and continuous quantitative semantics.

2.2.1.1 π-Calculus and Extensions

Regev, Silverman, and Shapiro [310] applied the π-calculus as a formalism for modelling biochemical systems, representing individual molecules as processes in the calculus, and producing a model of the RTK MAPK signal transduction pathway as a prototypical model. This then motivated the application and development of extensions of the π-calculus to better model different aspects of these systems. Hierarchical compartmental structures play a key role in biological systems, and are modelled in the Regev et al.'s BioAmbients [311] which models biological compartments including cells as *ambients* in the style of Cardelli and Gordon's ambient calculus [104]. β-binders [304] uses another compartmental model placing π-processes within "boxes" defining their external communication interface. The π@ [353] extends the π-calculus with polyadic synchronization and priorities, which enables the encoding of spatial calculi such as BioAmbients [353] and β-binders [95]. In SPiCO, π processes are extended to concurrent objects, to provide more structured modelling of biological processes [229]. Attributed π [215, 216], and its imperative extension Imperative π [212], extend the π-calculus with a communication mechanisms based on constraints over attribute stores, which offers another approach to encoding compartmental models. A similar imperative approach is followed by ℓ to provide a domain specific language for biochemical modelling [364]. Extensions of the π-calculus have also considered the motion of biological agents in space including Space Pi [211] which uses real space and time, 3π

based on affine geometry [100], and $\mathcal{L}\pi$[335] which is based on a network of connected locations.

Whilst the application of process algebra to biology first focused on modelling qualitative behaviour via a discrete non-deterministic semantics, languages soon branched out to investigate quantitative stochastic semantics. The Stochastic π-calculus [303] is a stochastic extension of the π-calculus which assigns quantitative rates to channels. Priami et al. [305] applied Stochastic π to biochemical modelling using stochastic simulation based on Gillespie's Stochastic Simulation Algorithm [180] with overall reaction rates derived according to the law of mass action. An abstract machine and improved stochastic simulator for the calculus in [291] whilst Cardelli and Mardare [105] introduced an improved semantics based on measure theory. Many extensions of the π-calculus have since developed support for stochastic simulation including [212, 290]. Stochastic β-binders [143] extended β-binders with a stochastic semantics using affinity between pairs of sites to capture quantitative reaction rates. BlenX [145] is a β-binders inspired modelling language for biological systems including functional reaction rates, whilst BlenXT [121] extends BlenX with biological transactions to encode general multiway reactions as atomic sequences of binary interactions.

Finally, a number of works have considered continuous ODE semantics for variants of π-calculus including translation of a restricted Chemical Ground Form (CGF) (which excludes ν-binders) to Chemical Reaction Networks (CRNs) in [97], an alternative direct continuous semantics for the CGF in [335], and the compositional continuous semantics of Continuous π [234] which we discuss in Section 2.2.1.3.

2.2.1.2 PEPA, Bio-PEPA, and Extensions

The Performance Evaluation Process Algebra (PEPA) was originally introduced by Hillston [197] as a stochastic process algebra aimed at quantitative analysis in computer systems based on a compositional stochastic semantics based on Continuous Time Markov Chains (CTMCs). In contrast to the binary communication of CCS and the π-calculus, it uses CSP-style multiway coordination on shared actions with quantitative rates. Calder, Gilmore, and Hillston [91] applied PEPA to the modelling of biochemical signalling pathways, exploring *reactant-centric* and *pathway-centric* abstractions to model the influence of RKIP on the ERK signalling pathway. These biological applications inspired a more appropriate quantitative rate semantics for the communication operator based on the generalized law of mass action [177] rather than *bounded capacity* [199]. This extension

developed alongside with a continuous population semantics, abstracting the collective behaviour of large populations of processes as systems of ODEs; the result of this *fluid approximation* [196, 346] to capture the limit behaviour of the mass action stochastic semantics for large populations.

Bio-PEPA [126] provides a more comprehensive reimagining of PEPA-style communication for modelling biochemical networks. It introduces an action based communication prefix capable of modelling different roles of a species in a reaction (reactant, product, activator, etc) and stoichiometries, along with support for general (non-mass action) kinetic laws modelled via functional rates. Bio-PEPA directly supports a population-level continuous semantics with processes representing biochemical species, and support for both a continuous ODE semantics and a stochastic semantics. Bio-PEPA also supports modelling molecular compartments or more general nested locations [123] and has been extended with discrete events [120].

Whilst this work represents a development from individual-level to population-level semantics, many real biological systems involve the interactions of components at each of these levels, motivating the use of *multi-level models* which explicitly model scales of structure in a system and the communication between them. This has been considered via the Process Algebra with Hooks (PAH) which has been applied to modelling pattern formation [144] and in the Process Algebra with Layers (PAL) which models communication between Bio-PEPA processes at the population and individual levels and has been applied to a multi-scale cell cycle and DNA damage repair model [330]. The earlier PEPA Nets [181] models mobility of PEPA processes between different locations with different context processes and combines process-level transitions with population level transitions altering network structure. Many biological systems also feature disparities in timescales or interaction with discrete components. HYPE [174] is a process algebra with full support for modelling hybrid systems based on composition of continuous flows in a style influenced by the continuous semantics of PEPA, whilst [172] introduces a hybrid semantics for Bio-PEPA based on a translation into HYPE.

An additional point of focus has been handling systems with uncertain parameters. ProPPA extends the syntax of Bio-PEPA to model uncertain parameters as distribution and introduces a FuTS-style [280] semantics to interpret them as non-deterministic stochastic systems, that is, Probabilistic Constraint Markov Chains (PCMDP). The language supports forwards stochastic simulation for given parameter values and backwards inference of model parameters from sample traces based on Approximate Bayesian Computation. This work has also considered inference based on fluid approximation and linear

noise approximations of model behaviour [178], however, a formal continuous semantics for ProPPA was not included.

2.2.1.3 Continuous π and the bond-calculus

The Continuous π-calculus [234] is a π-calculus based process algebra for biomolecular systems, featuring continuous mixtures of processes, and a continuous ODE semantics based on vector fields. It uses a global *affinity network* which assigns quantitative affinities to pairs of sites as well as local affinity networks defined via an extended form of ν binder. Continuous π semantics represents continuous mixtures of species as a vector Π and uses an *interaction tensor* $\oplus_M$ to compositionally define a vector field over species (or equivalently, a symbolic system of ODEs) deriving the continuous evolution of the system according to the law of mass action. This style of continuous semantics brings the benefit of compositionality, with new reactions derived incrementally as new species are added to the system. The Continuous π-calculus and its semantics have also been been formalized in the Lean theorem prover [130].

The bond-calculus was originally conceived as an extension of the Continuous π-calculus in the following directions:

- Extending binary mass action affinities to multiway interactions with general kinetic laws whilst preserving compositionality in the continuous semantics.
- Capturing *contextual affinities* which depend on interactions between multiple sites in a single species.
- Enable compositional affinity networks which can be imposed upon existing processes.

It also takes some inspiration from the join-calculus' [166] use of join-patterns in moving from affinity networks to *affinity patterns*. An early version of the bond-calculus which appeared in the author's masters dissertation featured a related graphical communication mechanism based on labelled hypergraphs [358].

2.2.1.4 Rule-Based Languages

An alternative approach to biochemical modelling is given by rule-based languages which use a simpler structural description of molecules with reactions specified separately via rewriting rules. A pioneering instance of this approach was the bio-calculus [272] which

uses a join-calculus–style [166] pattern matching based communication mechanism to model chemical reactions. BioNetGen [51] uses an extended rewriting based approach to generate Chemical Reaction Networks from a smaller set of reaction rules. Kappa [134] provides a rule-based language for protein interactions using a process algebra inspired approach involving agent-based descriptions of molecular structures being acted on by reaction rules. BIOCHAM [107] is a rule-based modelling language for biochemical systems which provides a number of different qualitative and quantitative semantics coupled with automated Temporal Logic verification techniques. The Calculus of Looping Sequences (CLS) [24] and its extensions [26, 27] are a somewhat different rewriting based formalism for biological modelling with reactants represented as recurring sequences. Subsequently, the Language for LBS [286] and LBS-κ [285] extended the rule-based approach with modules, allowing large modules to be constructed in a compositional way. The Bio-Chemical Space Language (BCSL) [142, 347] aims at extending the rule-based approach with a higher level of abstraction and a semi-formal description of chemical reactions allowing partial specification of biochemical structures.

There has long been a fruitful overlap between rule-based and process algebra-based modelling languages with Kappa taking particular inspiration from process algebra. The rule-based language React(C) combines Kappa with hyperedges and constraints to close any expressiveness gap with variants of the π-calculus and to enable a transparent encoding of the Stochastic π-calculus [213]. As with quantitative process algebras, most of the above languages offer a choice of non-deterministic, continuous, and stochastic semantics, with, for example, KADE [94] recently providing a tool to translate Kappa models to reduced systems of ODEs.

2.2.1.5 Others

There have been a large number of other approaches to modelling biological systems. The early approach of Fontana and Buss [165] used the λ-calculus as a model of molecular evolution. Another early precursor to the development of biological process algebras was the application of the stochastic discrete-time process algebra Weighted Stochastic CCS (WSCCS) to modelling insect behaviour [344]. Meanwhile, the stochastic fusion calculus uses a symmetric form of communication based on global fusion of names and has been applied to modelling gene regulation [116] and fungus/root symbiosis [119]. In Stochastic Concurrent Constraint Programming (sCCP) [61] agents communicate indirectly via a global store of constraints; this communication model has been applied to modelling a

wide range of biological systems and supports stochastic, continuous [69], and hybrid semantics [66].

A number of calculi have focused around the structure and dynamics of biological membranes including Cardelli's Brane Calculus [96] and Bitonal and Atonal Calculi. Bigraphical languages such as the $\mathcal{C}$-calculus [133], Bio-β [15], Biological Bigraphs [16], and Stochastic Bigraphs [226] have been applied to modelling protein and membrane interactions [16]. In an alternative approach, the link-calculus uses an open multiway communication mechanism based on the formation of link-chains and is able to transparently encode compartmental calculi [55].

The Strand Algebra [99] focuses on DNA strand interactions and can be used as a model for DNA computing. The Beacon Calculus [57] is a general purpose process calculus with a flexible guarded communication mechanism, which can concisely model DNA replication and damage repair as well as multisite protein phosphorylation. In a different direction, the Bonding Calculus [10] uses a distinct form of concurrent communication to directly model the creation and dissolution of covalent chemical bonds.

Finally, there have been a number of attempts to give a unified semantics of different process calculi. One approach involves encodings between different calculi, some of which have been listed above. Milner's Bigraphical Reactive Systems [266] seeks to unify the treatment of mobility of agents and ambient structures based on Bigraph rewriting. PRISMA [79] gives a general framework for qualitative calculi which encompasses both π-calculus–style name passing and CSP-style multiway communication. FuTS [280] and SGSOS [223] have both been proposed as uniform frameworks for defining the semantics of quantitative languages using extended forms of quantitative transition systems.

2.2.2 Modelling Domains

The biological process calculi surveyed in the previous subsection cover a wide range of different biological modelling domains. The abstract nature of process calculi allows a given language to span multiple different modelling domains, however, the development of models in specific domains has frequently inspired the development of new modelling styles and formalisms. These developments have also inspired unifying abstractions for biological systems with vastly different contexts and scales. In particular, the bond-calculus aims to support a range of modelling domains, some of whose development is outlined below.

2.2.2.1 Biomolecular Interactions and Bonding

Early applications of biological process algebra focused on molecular biochemistry and modelling molecular bonding. Models in the π-calculus from [310] onwards used the passing and sharing of names to represent molecular bonding, with two components of a molecule linked by a shared name (ν binder). This domain is also directly targeted by rule-based languages such as Kappa [134], BioNetGen [51], and LCLS [27] which use the creation of links to form molecular bonds. β-binders and BlenX[145] use boxes with typed interaction sites as a model for the interaction interfaces of molecules corresponding to sites or binding domains on the surface of proteins. In [234] a Continuous π model is developed for a post-translational genetic circadian clock and the language's continuous semantics is used to generate associated ODEs for simulation; this model is used in [232, 233] to study the molecular evolution of the clock. It is also possible for dynamic bonding to result in infinite families of species as a model of polymerization; this phenomenon has been studied in Stochastic π by [106] and in BlenX by [235].

2.2.2.2 Gene Regulation

Gene Regulation has been studied in a variety of mathematical modelling formalisms at the molecular and network levels; a recent survey of formal models is [25]. Gene Regulation has also been a core case study for stochastic process calculi starting with Stochastic π [303]. Kuttler and Niehren's λ-switch model [227] is a classical model of cooperative gene regulation in the Stochastic π-calculus and has served as a case study for several stochastic process algebras [117, 118, 215, 229] and rule-based languages [35]. Subsequent molecular-level models have investigated the mechanics of transcription and translation [227], whilst in their model of transcriptional attenuation at the Typ operon, Kuttler, Lhoussaine, and Nebut [228] argue the advantages of a rule-based approach due to the difficulty of capturing multiway interactions. Blossey and Cardelli [52] proposed a general compositional approach to network-level modelling of gene regulation in Stochastic π, where processes represent regulatory interactions rather than individual genes; extensions of this model include [53, 204, 316]. In [69] a similar compositional model is presented for representing gene regulatory networks in sCCP, whilst Bio-PEPA is well suited to modelling gene regulation at the network level [125]. Following a different approach, [284] presents a general translation of the Process Hitting framework for gene regulatory networks to Stochastic π.

2.2.2.3 Population Dynamics

Populations level modelling focuses on modelling the overall behaviour of large groups of individual organisms and encompasses ecological models focused on the growth of species in an ecosystem as well as epidemiological models focusing on the spread of disease through a population; an introduction to the use of mathematical modelling in these areas is [75] whilst [345] surveys the use of individual-based models [345]. Both [281] and [260] investigated stochastic WSCCS models of population dynamics for epidemiological models whilst [261] looked at host-parasite population dynamics. In [259] McCaig, Norman, and Shankland investigated methods of generating population level ODE models from WSCCS models. Bradley, Gilmore, and Hillston [73] analysed internet worm transmission application using a PEPA model and fluid flow approximation. The models of [40, 42] further evaluated the suitability of PEPA for epidemiological modelling, leading to a proposed improvement to the fluid flow approximation of PEPA to better model population dynamics [41] whilst [124] presents a version of Bio-PEPA tailored to epidemiological models. In [329], an ecological model of Pacific oysters was presented based on a translation of Dynamic Energy Budget (DEB) models to Bio-PEPA. Spatial distribution of agents also plays an key role in epidemiology and has been investigated in spatial process algebras including PLAPS [289], MELA [355], PALOMA [161], and CARMA [71, 288].

2.2.3 Modelling Abstractions

Whilst the huge variety of models and modelling formalisms for biological systems can appear somewhat of a jungle [278], we can discern a few distinct core modelling abstractions to make sense of the overall landscape:

- Regev's original π-calculus models advocated the "process as a molecule" approach, which was furthered by Stochastic π's use of individual-based stochastic simulation.
- PEPA [92] and Bio-PEPA [126] models use a "process as a species" or *reagent-centric* abstraction where processes represent the characteristic individual of a chemical species.
- Rule-based languages such as Kappa can be said to take a "processes as molecular structure" approach and leave dynamics to the rules.
- Spatial languages such as BioAmbients and the Brane Calculus use a "ambient as cell" or more generally "ambient as compartment" abstraction.

Whilst other languages and models have proposed alternative[2] abstractions including:

- Duchier and Kuttler proposed an object-oriented abstraction of "concurrent objects as molecules" [154] which is developed in SPiCo [229].
- The Gene Gate model uses a "process as a regulatory interaction" abstraction to give a compositional encoding of Gene Regulatory Networks in the Stochastic π-calculus [52].
- In [92] an alternative "process as a subpathway" or *pathway-centric* abstraction is used for modelling biochemical signalling pathways in PEPA; this model is shown to be bisimilar to a reagent-centric model of the same system.
- In β-binders [304] and BlenX [145], boxes represent the external communication interfaces of molecules at typed sites, whilst π-processes govern their internal behaviour.
- Attribute-based languages such as Attributed π [215] and the Beacon Calculus [57] use an "attributes as molecular state" abstraction.
- sCCP [61] uses a "variables as species concentrations" abstraction, combined with a "processes as interactions" encoding of population dynamics via concurrent constraints, whilst [210, Section 4.2.2] shows how this style of population-based modelling can be replicated in Imperative π via reaction constraints.
- BlenXT [121] uses a "reactions as transactions" abstraction to model general reactions as transactions atomically combining multiple binary interactions.

We can also make a broad distinction dating back to CCS and CSP between languages such as the π-calculus and its variants which use binary communication based on names and conames and languages such as PEPA and its descendants which use multiway synchronisation coordinated on synchronisation sets.

The bond-calculus fits in between many of these classes. It uses a form of symmetric multiway synchronisation which is capable of *internal* mobility of names in the style of the πI-calculus [322] and of modelling general multiway interactions similarly to PEPA and Bio-PEPA. This is driven by affinity patterns, which, like the rules of rule-based languages, determine the valid patterns of interaction between processes and their quantitative rates by pattern matching. However, the results of each reaction are determined by the processes

[2]This distinction depending on the vantage point of the beholder.

themselves, retaining an agent-based style similar to the π-calculus. This is also influenced by the use of affinity in Continuous π [234] and BlenX [145], however, we generalise the notion of affinity from binary mass action reactions to multiway interactions with general kinetic laws[3] and our use of pattern matching captures *context-dependent affinities*.

2.2.3.1 Openness and Contexts

Traditionally, mathematical models such as ODEs and Chemical Reaction Networks describe biological systems as *closed systems* where all of the where all of the reactants in the system are explicitly modelled. The use of process algebra models substantially improves upon this by enabling compositional modelling whereby the behaviour of an overall system can be derived from models of smaller components. For example, in a π-calculus model new reactants can be added to the system, leading both to new reactions with existing reactants, and to the dynamic generation of new species due to dynamic complexation. The bond-calculus takes this further, permitting the following forms of openness within a model:

- **Open Multiway Interactions** The bond-calculus allows interactions involving a combination of different sites to gradually build up to an overall reaction until a complete *affinity pattern* is reached, and for the overall reaction rate to change in a nonlinear manner when new concurrent components provide additional reactants at any of these sites. Thus the bond-calculus models open interactions in the sense of [54].

- **Open Affinity** In the bond-calculus the reactions in a given mixture of processes are determined by the affinity of sites and are not fixed, since putting the mixture in parallel composition with a new affinity network may lead to additional interactions between existing species.

These features are supported by the compositional semantics of the bond-calculus which is compositional in both the species in a mixture and the patterns in an affinity network. This extends to the dynamic continuous semantics of models allowing new reactions and ODEs to be computed incrementally as new components and affinity patterns are added to the system. These features are crucial to the use of the bond-calculus with a spatial logic since the context operator $\mathcal{C} \triangleright \varphi$ relies on the ability to reinterpret a model in a new

[3]Whilst BlenX does support functional reaction rate laws, these rates are dependent on global species concentrations [145], whereas the bond-calculus calculates reaction rates compositionally based on the concentrations of the sites matched by an affinity pattern.

context by composing it with a process representing its broader environment or contexts. Whereas for the Continuous π-calculus these contexts could only add new reactants to the system, for the bond-calculus they can also complete unfinished multiway interactions and add new reaction rules to the system, substantially increasing the scope of contextual properties.

2.3 Temporal and Spatial Logics

In this section we introduce Temporal and Spatial Logics and their roles in specifying and verifying behavioural properties of systems.

2.3.1 Linear and Branching Temporal Logics for Discrete Systems

The logical treatment of time dates back to ancient philosophy with Aristotle and his medieval commentators' considerations of contingency and necessity [282], however, the start of formal modal temporal logic comes with Prior's Tense Logic [306] which uses modal temporal operators Future / Eventually $\mathcal{F}\varphi$ (*it will be the case that* φ *holds*) and Globally / Always $\mathcal{G}\varphi$ (*it will always be the case that* φ *holds*) as well as their time reversed duals Past $\mathcal{P}\varphi$ and Historically $\mathcal{H}\varphi$. Temporal logic has since come to wide spread prominence in formal verification with the introduction of verification techniques based on temporal sequencing of system states and events [84]. This approach was formalised by Pnueli's seminal Linear Temporal Logic (LTL) [170, 300], which adapted a variant of Tense Logic to traces of events in a computer system. The syntax of LTL is given by the grammar,

$$\varphi, \psi \ ::= \ \rho \ \mid \ \varphi \wedge \psi \ \mid \ \neg\varphi \ \mid \ \mathcal{X}\,\varphi \ \mid \ \varphi\,\mathcal{U}\,\psi$$

where the atomic propositions ρ of the logic consist of events of the system. Here the neXt operator $\mathcal{X}\varphi$ asserts that φ will hold at the next discrete time instant, whilst the temporal Until operator $\varphi\,\mathcal{U}\,\psi$ states that φ holds until some time instant at which ψ holds. The until operator [219] can be considered a generalization of the $\mathcal{F}$ operator which may be encoded as $\mathcal{F}\,\varphi \equiv \mathbf{T}\,\mathcal{U}\,\varphi$, whilst $\mathcal{G}$ can also be encoded as the De Morgan dual $\mathcal{G}\,\varphi \equiv \neg\,\mathcal{F}\,\neg\varphi$ of $\mathcal{F}$. This application of temporal logic has played a wide ranging and effective role in formal verification since future temporal modalities correspond with the main classes of verification problems, with $\mathcal{G}$ capturing *safety properties* of systems (things which must always be the case) and $\mathcal{F}$ capturing *liveness properties* (things which must eventually happen).

2.3.1.1 Branching and Metric Time

Linear Temporal Logic is based on a discrete linear model of time which, given the state of the system at one time instant, admits only one possible successor state at the next time instant. However, there is an alternative *branching* model of time in which each state leads to multiple possible futures, as first noted in Kripke's 1958 letters to Prior [299] which suggested a tree-structured notion of time was necessary to capture indeterminacy. This approach became central to the verification of nondeterministic computer systems with the introduction of the Computational Tree Logic (CTL) [129], which extends LTL with a branching model of time and adds path quantifiers $\mathcal{A}\varphi$ (φ *holds on All paths*) and $\mathcal{E}\varphi$ (*there Exists a path on which* φ *holds*). The other main limitation of LTL and CTL is their discrete model of time, ignoring real-time aspects of systems such as *punctuality*. The Metric Interval Temporal Logic (MITL) [8] uses variants of the LTL temporal operators bounded by intervals $I = [a, b]$, so that, $\mathcal{F}_{[a,b]}\,\varphi$ states that φ should hold within at some point within a and b time units in the future, whilst, $\mathcal{G}_{[a,b]}\,\varphi$ states that φ should hold at every point within a and b time units in the future.

2.3.1.2 Model Checking

Much of the power of temporal logics in verification of discrete finite state systems has come from their synergistic relationship to model-checking techniques which evaluate the truth of formulae based on a model of all possible state transitions in the system (a *Kripke structure* [225] in the case of finite state systems). Model checking was introduced with CTL in [129] and independently in [308]. A major challenge which model checking must overcome is the "state space explosion" problem wherein the number of states grows exponentially with the number of system components. This motivated the introduction of *symbolic model checking* which uses symbolic representations of system states to drastically increase the scalability of model checking [83, 262]. For stochastic systems there has been widespread interest in the probabilistic model-checking problem [238] of determining whether a system satisfies a property φ with a probability greater than a given threshold. This has led to mature probabilistic model checkers such as PRISM [231] which has been applied to modelling checking behaviours of signalling pathways in [93].

2.3.1.3 Modelling Checking and Biological Modelling Languages

A number of process algebraic and rule-based frameworks have coupled biological modelling languages with model-checking techniques to automatically verify temporal properties of

biological systems. BIOCHAM [107] provides an integrated environment for analysing temporal logic properties of rule-based models, enabling model checking of CTL properties by applying the NuSMV model checker [114] to its non-deterministic discrete semantics and model checking of LTL($\mathbb{R}$) properties (which combine quantitative state constraints with LTL temporal operators) over numerical traces produces via its continuous ODE semantics [158]. Bio-PEPA supports a translation to PRISM models [126], making it possible to apply probabilistic model checking to verify probabilistic properties of a model's behaviour.

Some of this work has focused on handling uncertainty. Fages and Rizk [157] proposed techniques for temporal logic constraint solving over BIOCHAM models to find model parameters consistent a given temporal property. The u-Check tool [63] implements a number of Gaussian Process based techniques for the analysis of uncertain systems and supports Bio-PEPA models as an input format. The Three-Valued Spatio-Temporal Logic (TSTL) [249] is a spatio-temporal logic for properties of stochastic spatial systems modelled in the MELA [355] process algebra, which uses a three-valued semantics to handle stochastic uncertainty based on confidence intervals.

2.3.2 Signal Temporal Logic

Signal Temporal Logic (STL) [253] provides an effective specification languages for temporal behaviour over continuous real time domains. STL-based logics have emerged as a common language for specifying and monitoring quantitative systems, be they electronic [279], biological [30], and cyber-physical [32] in nature. The logic combines assertions, such as $[X] > 3$ or $[X]^2 + 3\,[Y]^2 > 2$, over the state variables of a model with MITL-style interval temporal operators which specify at which time points these assertions are expected to occur. That is, the logic includes as atomic propositions inequalities $\rho \triangleq f(\mathbf{x}) > 0$ over the system state variables $\mathbf{x} = ([X_1], \ldots, [X_n])$[4], whilst the complex propositions of the logic follow the grammar,

$$\varphi, \psi \ ::= \ \rho \ \mid \ \varphi \wedge \psi \ \mid \ \varphi \vee \psi \ \mid \ \varphi \Rightarrow \psi \ \mid \ \neg\varphi \ \mid \ \mathcal{F}_I\,\varphi \ \mid \ \mathcal{G}_I\,\varphi \ \mid \ \varphi\,\mathcal{U}_I\,\psi$$

where $I \in \mathbb{IR}$ is a real interval. Unless otherwise indicated we will usually assume that the functions f are polynomials and that the time intervals I are bounded.

The semantics of the logic is specified by defining when a property φ is satisfied by a system trajectory $\mathbf{x} : T \to \mathbb{R}^n$, defined over a time interval $T \subseteq \mathbb{R}_{\geq 0}$, at each time

[4]We note this encompasses more general forms of inequalities $f(\mathbf{x}) > g(\mathbf{x})$.

instant $t \in T$, denoted $(\mathbf{x}, t) \models \varphi$ or in shorthand $\mathbf{x} \models_t \varphi$. Then the semantics of atomic propositions is given by

$$\mathbf{x} \models_t f > 0 \qquad \text{iff} \qquad f(\mathbf{x}(t)) > 0$$

whilst the semantics of logical operators is defined by

$$\begin{aligned} \mathbf{x} &\models_t \varphi \wedge \psi \qquad \text{iff} \qquad \mathbf{x} \models_t \varphi \text{ and } \mathbf{x} \models_t \psi \\ \mathbf{x} &\models_t \neg\varphi \qquad \text{iff} \qquad \mathbf{x} \not\models_t \varphi \end{aligned}$$

and the semantics of the until operator is defined by

$$\mathbf{x} \models_t \varphi \,\mathcal{U}_I\, \psi \qquad \text{iff} \qquad \begin{cases} \text{for all } s \in I, \\ \quad (\ \mathbf{x} \models_{t+s} \psi \text{ and} \\ \quad\ \text{for all } s' \in [t, t+s'],\ \mathbf{x} \models_{s'} \varphi\ ) \end{cases}$$

The semantics for the other logical operators are defined by the standard equivalences $\varphi \vee \psi \equiv \neg(\neg\varphi \wedge \neg\psi)$ and $\varphi \Rightarrow \psi \equiv \neg\varphi \vee \psi$ whilst the semantics for the other temporal operators is defined in terms of $\mathcal{U}_I$ based on the standard equivalences $\mathcal{F}_I\, \varphi \equiv \mathbf{T}\, \mathcal{U}_I\, \varphi$ and $\mathcal{G}_I\, \varphi \equiv \neg\, \mathcal{F}_I\, \neg\varphi$.

STL is often combined with numerical methods to provide approximate verification of properties based on Boolean signals monitoring [253]. For each atomic proposition $\rho \triangleq f(\mathbf{x}) > 0$, an approximate numerical trajectory $\mathbf{x}_a : T \to \mathbb{R}^n$ of a system over a time interval $T \in \mathbb{IR}_{\geq 0}$ can be used to give a *Boolean signal* $s : T \to \mathbb{B}$ for ρ based on the changes of sign of the function $f(\mathbf{x}_a(t))$ (in fact, a numerical method with *real-root detection* may be used to precisely pinpoint these roots [253]). The Boolean signal monitoring algorithm of [253] then combines Boolean signals for each atomic proposition in order to compositionally compute signals for complex propositions.

2.3.3 Space and Contexts

Whilst temporal logics provide an effective way to specify the behaviour of systems over time, much of this behaviour cannot be understood without considering spatial aspects of a system such as spatial variability in state or a system's interactions with its wider surroundings or *context*. This has motivated the development of frameworks such as spatial logics to describe spatial aspects of a system's state, and of spatio-temporal logics to describe the evolution of this state over time. The abstract nature of these frameworks has allowed their development to take place across a number of different areas targeting diverse

applications including biochemical modelling, ecology, concurrent communication, program verification, security, artificial intelligence, robotics, and cyber-physical systems. We will now survey some of the most relevant works in the field starting with those logics which are explicitly aimed at modelling agents in physical space (of various degrees of abstraction) and those closer to our interests in applying spatial abstractions in more abstract spaces (i.e. the compositional structure of concurrent processes or the concentration space of a population model).

2.3.3.1 Physical Space

Spatial logics make it possible to express properties about the spatial structure of systems based on models and abstractions of the physical spaces in which they reside. One such spatial abstraction, quad-trees [162], represents discrete or continuous 2D space in an efficient hierarchical manner as a tree of successively subdivided regions. This representation is used in the Linear Spatial-Superposition Logic (LSSL) [184], which uses spatial analogues of the LTL operators neXt, Until, and Release to specify spatial patterns as lattices of excitable cells as path properties of a quad-tree representation of their spatial structure. The Tree Spatial-Superposition Logic (TSSL) [31] adopts a similar approach but possesses path quantified operators similarly to the CTL [129] and uses a branching interpretation of quad-trees to express a wide variety of spatial patterns in reaction diffusion networks [349]. The Spatial Logic for Closure Spaces (SLCS) [113] takes a topological view, using closure spaces as a spatial abstraction which extends to discrete spatial structures such as graphs, whilst its spatial Until operator (Surrounded) and spatial neXt operator (Near) make it possible to express topological properties such as boundaries and interiors. An extensive earlier body of work used spatial logics for representing and reasoning with qualitative knowledge about space, including the logic RCC-8 [309] which expresses interconnectivity of spatial regions and can be encoded in the modal logic S4. Such topological approaches draw upon a deep connection between modal logics and topology as covered in [43].

Spatial properties are most interesting in systems whose state changes over time and consequently many spatio-temporal logics have emerged which combine spatial operators with temporal operators in order to express the properties of the overall spatio-temporal behaviour of a system. The SpaTeL [188] logic combines the spatial operators of TSSL with STL temporal operators and applies statistical model checking over time sequences of quad-trees. This has been applied to verify the dynamic evolution of reaction diffusion patterns and power usage in smart neighbourhood grids [188] and for multi-agent motion

planning in a 2D environment via a Mixed Integer Linear Programming (MILP) encoding [245]. The Signal Spatio-Temporal Logic (SSTL) [65] also uses STL temporal operators but employs a bounded *surround* operator and bounded *somewhere* operator which are able to express topological and metric properties. Its implementation has focused of discrete weighted-graph structured population models supporting either stochastic simulation or a continuous fluid approximation of concentrations; this has been applied to pattern formation and bike sharing systems [277]. In [319] a form of discretization is used to apply SSTL to continuous particle-based simulations. On the other hand the STLCS logic [113] combines the spatial operators of SLCS with the temporal operators of CTL to give a spatio-temporal logic interpreted over a discrete time sequence of "snapshots" of a closure space. ImgQL [82] extends STLCS with distance operators to specify metric properties and has been applied to analysis of medical imagery. The Spatio-Temporal Reach and Escape Logic (STREL) generalises SSTL with spatial reach and escape operators which are able to capture topological and metric properties of systems with dynamically changing spatial structures. In [244] GSTL, a graph-based extension of STL with operators for parent and sibling relationships between nodes, is applied to robot motion planning. There have also been a number of spatio-temporal logics based on RCC-8 [361] and S4 [171, 239, 339].

Other logics have combined temporal logic with assertions over symbolic system variables as a means of reasoning about space. The temporal logic S-STL (Spatial Signal Temporal Logic) [287] combines STL operators with quantified inequalities over the variables of a system of PDEs and is able to verify spatio-temporal properties of continuous systems based on finite element methods. In [326] an interval spatio-temporal logic for the motion of shapes in multi-dimensional space is introduced based on an extension of the duration calculus to the spatial variables of a system. The Differential Dynamic Logic (d$\mathcal{L}$) [295] is able to express a wide variety of properties of cyber-physical systems in a dynamic logic including Hybrid Programs which often include spatial variables. Quantified Differential Dynamical Logic ($\mathcal{Q}$d$\mathcal{L}$) [296] expands this logic with location quantifiers to explicitly model distributed hybrid systems.

2.3.3.2 Space and Contexts in Concurrency

There is an alternative tradition of spatial logics for concurrency in which the notion of space does not correspond directly to the locations of objects in the physical world but to the composition of components in a concurrent system. This tradition was pioneered

with the ambient logic of [103] which originally served as a logic for reasoning about the hierarchical locations in the Ambient Calculus. This approach has subsequently been applied more generally to the composition structure of many process calculi [87–89], reasoning about data structures such as trees [102], graphs [101], and heaps [313], and also to expressing properties of Brane Calculus models of biological membranes [264]. Ambient Logic includes a spatial composition operator $\varphi \mid \psi$, asserting a process can be decomposed as a parallel composition of a process satisfying φ and a process satisfying φ, its logical adjoint, the *guarantee* operator[5] $\psi \triangleright \varphi$ specifying that a process satisfies φ when composed with a processes satisfying ψ. The logic also features standard logical operators, a temporal eventually operator, and a range of operators for reasoning about fresh names in the style of nominal logic [293]. The full logic turns out to be very penetrating, being able to discriminate processes essentially up to structural congruence[6] [323], revealing *intensional* information about the structure of agents whereas process logics such as Hennessy-Milner Logic [191] discriminate agents *extensionally* up to their externally observable communications (bisimilarity). Whilst a number of authors have proposed model-checking algorithms for different fragments of these logics [90, 103], interactions between the different operators of the logic frequently prove too difficult to handle with the guarantee operator $\psi \triangleright \varphi$ often leading to undecidability [17, 86, 108]. Spatial logics have also been investigated for quantitative systems with Rounds [318] proposing a spatial logic for concurrent hybrid systems modelled in the Φ-calculus [317], whilst Larsen, Mardare, and Xue [236] proposed the spatial logic Concurrent Weighted Logic (CWL) for reasoning about weighted labelled transition systems.

Various related spatial logics have fruitful applications in other settings, using weaker spatial operators to enable automated verification. In [255] a decidable extension of Hennessy-Milner Logic with a spatial composition operator is presented. The spatial composition and guarantee operators correspond respectively to the bunched conjunction $*$ and implication $-\!\!*$ operations of bunched logics [131] for reasoning about availability of resources, whilst [307] applies this in the MBI spatial logic for reasoning about resource dependent processes in the SCRP process algebra. MoMo [139] is a modal logic for reasoning about mobile tuple-passing agents in the Klaim language [138], which features guarantee-like production and consumption operators which specify the behaviour of a system after producing or consuming a given resource. MoSL$^+$ [140] is a stochastic logic which features MoMo-style resource operators and supports stochastic model checking.

[5] Also referred to as *spatial implication*.

[6] Precisely, the equivalence relation $=_{\mathcal{L}}$ defined by logic is structural congruence + η-equivalence.

SoSL [141] is a stochastic logic for service-oriented systems in a similar vein, which also features a open-ended reactivity operator which models the reaction of an open-ended system to an external stimulus. Rather differently, in [328] it is shown that the migration of the Physarum slime mould can be viewed as a naturally occurring non-modal spatial logic, offering a spatial logic approach to natural computation.

2.3.3.3 Connections

There are natural connections between the two notions of space we have discussed in this section. The hierarchical notion of space present in the ambient calculus is within the scope of the topological framework of closure spaces over which SLCS is defined, as attested by applications of ambient logic variants to trees [102] and graphs [101]. In Chapter 8 of this thesis we will explore another connection, applying a form of subdivision-based spatio-temporal monitoring to model checking context operators.

2.3.4 Logic of Behaviour in Context

The Logic of Behaviour in Context ($\mathcal{LBC}$) is a spatio-temporal logic for biological systems modelled in the Continuous π-calculus. It combines the temporal operators of STL, interpreted over the concentrations of agents, with a spatial *context operator* $\Pi \triangleright \varphi$ which is based on the guarantee operator of Ambient logic but provides a concrete *context process* Π which consists of a Continuous π process modelling an environment into which the process is placed. The formal syntax of $\mathcal{LBC}$ is defined by

$$\varphi, \psi \ ::= \rho \mid \varphi \wedge \psi \mid \varphi \vee \psi \mid \varphi \Rightarrow \psi \mid \neg\varphi \mid \varphi \, \mathcal{F}_I \, \psi \mid \varphi \, \mathcal{G}_I \, \psi \mid \varphi \, \mathcal{U}_I \, \psi \mid \Pi \triangleright \varphi$$

where $I = [a, b]$ is a time interval, ρ is an inequality over species concentrations $[X_1], \ldots, [X_n]$, and Π is a Continuous π process. Banks and Stark developed model-checking techniques for $\mathcal{LBC}$ based first on monitoring properties over discrete numerical traces [19] and then based on continuous signal monitoring [23]. The combination of Continuous π and $\mathcal{LBC}$ offers an *in silico* workbench for modelling *spatio-temporal experiments* which use context operators to introduce new reactants to systems and verify temporal properties of a system's response to these external stimuli. This approach has been applied to a post-translational oscillator [217] to characterise biologically relevant properties such as oscillation, coupling, and phase response [22].

The focus of $\mathcal{LBC}$ on approximate monitoring over numerical simulation traces brings some limitations. The continuous ODE interpretation of models assumes complete knowl-

edge of the system being modelled and that their behaviour is deterministic. The form of contextual information which may be expressed is also significantly restricted compared to Ambient Logic, given the restriction of the left-hand side of the guarantee operator from a logical formula to a concrete context process. However, some important non-determinism is still present in the logic due to the combination of interval temporal operators with context operators. For example, the property

$$\mathcal{G}_{[a,b]}\,(\Pi \triangleright \varphi)$$

specifies that the property should hold if the context process Π is introduced at any point between a and b time units in the future. This allows the logic to express interesting properties such as robustness of a system to the timings of its interactions with its environment and oscillations whilst it still assumes complete knowledge of that environment once it is introduced. This temporal non-determinism accounts for most of the complexity of monitoring $\mathcal{LBC}$ since, whereas STL properties can be monitored based on a single execution of a system, $\mathcal{LBC}$ monitoring requires monitoring multiple executions of the systems covering each time instant at which the context could be introduced, and nesting of context operators entails an exponential increase in the amount of execution and monitoring required. We also note that, due to the form of Continuous π processes, contexts are restricted to adding new reactants to the system and cannot reconfigure the reaction rules of the system.

In [23] an alternative method of $\mathcal{LBC}$ verification was proposed using sensitivity analysis [151] to extend numerical trajectories to cover multiple potential context introduction times. For $\mathcal{LBUC}$ verification we must handle many additional sources of uncertainty including uncertain initial conditions, uncertain rate parameters, and uncertain contexts. These challenges motivate us to develop new verification methods throughout this thesis, using verified monitoring over Flow* flowpipes [110] to achieve sound $\mathcal{LBUC}$ verification under uncertainty.

2.4 Formal Verification of Continuous Systems under Uncertainty

Consider the class of continuous system defined by initial value problems,

$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = \mathbf{f}(\mathbf{x}); \quad \mathbf{x}(0) = \mathbf{x}_0$$

consisting of a system of coupled differential equations defined via Lipschitz continuous function $\mathbf{f}$ and an initial value constraint $\mathbf{x}(0) = \mathbf{x}_0$. More generally, we can consider hybrid systems which combine the continuous evolution of the system with discrete jumps in state or dynamics and may be represented as Hybrid Automata [9]. For both of these classes of systems it can be very challenging to analyse and verify behavioural properties since we do not have explicit solutions for system trajectories making it difficult to automatically reason about the system's temporal evolution.

Traditional numerical methods compute numerical solutions $(\mathbf{x}_i)_i$ for the system at a sampling of time points $t_i \in [0, T]$ over a finite time duration T. Whilst numerical methods have proved tremendously powerful in giving fast and reasonably reliable approximations to the solutions of systems, they bring with them two main drawbacks. Firstly, as approximate solutions they can produce erroneous answers due to either a numerical method's discretization of the continuous time domain of the system or the imprecision of the floating point numbers used in calculations, and moreover, they do not track the degree of uncertainty in the answers they provide. Whilst these sources of uncertainty are often small enough to ignore, this can make for a shaky foundation for a logical formalism, and furthermore, they can cause real problems in chaotic systems where small uncertainties can compound throughout the temporal evolution of the system. More fundamentally, they do not provide a way of representing or handling any of the manifold sources of uncertainty which go into modelling a real physical or biological system. These include both imprecision resulting from our limited ability to perfectly measure the parameters of a real system as well as fundamental uncertainties arising from genuine variability within the behaviours and environments of the many possible instances of real systems which a single model abstracts.

It is unclear how best to extend numerical methods to uncertain continuous and hybrid systems since we must quantify over uncountably many trajectories a system which may arise under different choices of uncertain inputs and parameters. In this section we introduce some methods for continuous systems verification under uncertainty, leading up to Taylor models and the Flow* verified integrator on which the verification methods of this thesis are based.

2.4.1 Uncertain Systems

Before we begin discussing verification methods we must first introduce the classes of systems we are interested in by extending the definition of continuous systems to consider

non-deterministic systems which are under the influence of different types of uncertainty. The first type we consider is uncertain initial conditions:

Definition 2.4.1. An (autonomous) Initial Value Problem with *uncertain initial conditions*

$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = \mathbf{f}(\mathbf{x}); \quad \mathbf{x}(0) \in X_0$$

consists of a system of ODEs and an initial value constraint $\mathbf{x}(0) \in X_0$ given a set X_0 of possible initial values.

These consider the behaviour of the system under the complete range of uncertain initial values $\mathbf{x}_0 \in X_0$; the trajectories of the system consist of all trajectories $\mathbf{x}_{\mathbf{x}_0} : [0, T] \to \mathbb{R}^n$ for concrete system instances $\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = \mathbf{f}(\mathbf{x});\ \mathbf{x}(0) = \mathbf{x}_0$ for some $\mathbf{x}_0 \in X_0$.

We are also interested in parametric systems whose dynamics depend upon *time-invariant* uncertain parameters.

Definition 2.4.2. An (autonomous) Initial Value Problem with *uncertain parameters*

$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = \mathbf{f}(\mathbf{x}, \boldsymbol{\theta}); \quad \mathbf{x}(0) \in X_0; \quad \boldsymbol{\theta} \in \Theta$$

consists of a system of ODEs, a set initial value constraint $\mathbf{x}(0) \in X_0$, and a parameter constraint $\boldsymbol{\theta} \in \Theta$ given a set $\Theta \subseteq \mathbb{R}^m$ of possible parameter values.

The trajectories of the system consist of all trajectories $\mathbf{x}_{\mathbf{x}_0,\boldsymbol{\theta}_0} : [0, T] \to \mathbb{R}^n$ for concrete system instances $\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = \mathbf{f}(\mathbf{x}, \boldsymbol{\theta}_0);\ \mathbf{x}(0) = \mathbf{x}_0$ for some $\mathbf{x}_0 \in X_0$ and some $\boldsymbol{\theta}_0 \in \Theta$.

We are also interested in the harder class of systems with *time-varying* uncertain inputs.

Definition 2.4.3. An (autonomous) initial value problem with *uncertain inputs*

$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = \mathbf{f}(\mathbf{x}, \mathbf{u}(t)); \quad \mathbf{x}(0) \in X_0; \quad \forall t, \mathbf{u}(t) \in \Theta$$

consists of a system of ODEs under a set initial value constraint $\mathbf{x}(0) \in X_0$ and an unknown input function $\mathbf{u} : [0, T] \to \mathbb{R}^m$ which is assumed to be continuously differentiable and satisfy the input condition $\mathbf{u}(t) \in \Theta$ for all $t \in [0, T]$.

The trajectories of the system consist of all trajectories $\mathbf{x}_{\mathbf{x}_0,\mathbf{u}} : [0, T] \to \mathbb{R}^n$ for concrete system instances, $\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = \mathbf{f}(\mathbf{x}, \mathbf{u}(t));\ \mathbf{x}(0) = \mathbf{x}_0$ for some $\mathbf{x}_0 \in X_0$ and some valid input function $\mathbf{u}$. Time-invariant parametric systems can be seen as a special case of Definition 2.4.3 where the unknown input function $\mathbf{u}$ is assumed to be a constant function.

Similarly to our discussion of stochastic and continuous process algebras, these classes of uncertain continuous systems can also be related to corresponding classes of uncertain stochastic systems by appropriate fluid approximation results [62]. This suggests the relevance of these classes of systems to analysing stochastic models [348], although this extension is outside of the scope of this thesis.

Some key problems for uncertain systems analysis include:

- *Reachability analysis*: Find or over-approximate the set of system states covered by all possible trajectories.
- *Reach-avoidance analysis*: Prove no trajectories of a given system intersect with a given set B of "bad states". This is equivalent to the bounded-time safety property $\mathcal{G}_{[0,T]}(\mathbf{x} \notin B)$.
- *Temporal logic model checking*: Prove all trajectories of a system satisfy a temporal logic property φ.
- *Parameter synthesis*: Find a subset $\Sigma \subseteq \Theta$ of parameters for which the system satisfies φ.

There are long recognised connections between these problems, which will be relevant throughout this section. Reachability analysis is frequently used as a means of verifying reach avoidance properties, and both are frequently referred to collectively as *reachability analysis*. Both problems form an subset of Temporal Logic model checking, which itself can be seen as a special case of parameter synthesis.

Beyond continuous systems, there has also been significant interest in hybrid systems verification. The base systems we will consider in this thesis are continuous since a hybrid semantics is beyond the current scope of the bond-calculus, however, $\mathcal{LBUC}$ properties describe the response of a system to discrete jumps induced by context operators, which can be seen as a logical counterpart of mode changes in a non-deterministic hybrid system. As such our approach is influenced both by methods for nonlinear continuous systems verification and by methods for hybrid system verification. We will also restrict our attention to systems with interval uncertain initial values $\mathbf{X}_0 \in \mathbb{IR}^n$ and interval uncertain parameters $\Theta \in \mathbb{IR}^m$.

2.4.2 Verification Methodologies

We will start by briefly introducing some of the main methodologies for systems verification under uncertainty.

2.4.2.1 Trajectory Sampling and Falsification

Whilst it is impossible to separately and precisely compute all of the uncountably many trajectories of a continuous or hybrid system, trajectory sampling methods attempt to compute enough numerical trajectories of a system to give sufficient coverage of its overall behaviour. Donzé and Maler proposed to compute trajectories for a sample of initial conditions to the system and to use *sensitivity analysis* to measure the sensitivity of trajectories to initial conditions making it possible to expand trajectories to tubes covering all possible initial conditions [151]; this method gives precise results for affine systems and approximate error bounds for general nonlinear systems. In [149] this is applied to the parameter synthesis problem of identifying regions of parameter space avoiding a given region of system's phase space. This method is extended by C2E2 which uses a precise symbolic representation based on *discrepancy functions* to perform exact reachability analysis of hybrid systems [155].

An alternative approach is given by continuous quantitative semantics for temporal logics introduced by Rizk et al. [314] and by Fainekos and Pappas [160]. The semantics of Rizk et al. [314] quantifies strongly a proposition is violated based on a metric on propositions whereas the semantics of Fainekos and Pappas [160] gives a signed measure of how strongly a proposition is satisfied or violated based on a metric on trajectories. Both of these measures of robustness increases our confidence in the robustness of results to modelling uncertainties or simulation errors. The continuity of these measures also offers a method of dealing with uncertain parameters and initial conditions by allowing us to explore a property's landscape of satisfaction over the parameter space. This leads to methods of *temporal logic falsification* such as [3, 12, 148, 314] which turn the verification problem on its head by applying optimization methods to search for trajectories which violate a property. Later techniques have also extended the approach of Fainekos and Pappas to monitor quantitative *intervals of satisfaction*, including [146] focusing on online monitoring (uncertainty about the future) and [206] and [363] both focusing on robustness to model and simulation uncertainties.

2.4.2.2 Topological Dynamics, Formal Proof, and Constraint Solving

An alternative approach to analysing continuous systems under uncertainty is to shift the focus of our verification efforts from local analysis of trajectories to global analysis of systems dynamics. This includes the mathematical disciplines of dynamical systems theory and topological dynamics for which a standard reference is [337], however, the

application of these techniques has traditionally required manual reasoning and proof. RoVerGeNe [33] attempts to automate global analysis under uncertainty for *piecewise multi-affine system* models of gene regulatory networks by applying LTL model checking to discrete abstractions of their dynamics; this also enables parameter synthesis via a hierarchical search of the parameter space. This approach has seen many subsequent developments including the Pythia tool [38], which combines a more expressive HUCTL_{P} logic [37] with parallel semi-symbolic model-checking techniques [77], and in the Hidentify tool [58] which applies the SpaceEx model checker to a quantitative Linear Hybrid Automata abstraction of system dynamics.

Other work has tried to more directly generalize mathematical reasoning about topological dynamics through formal logic and theorem proving. The Differential Dynamic Logic ($\text{d}\mathcal{L}$) [295] employs an encoding of hybrid systems in a dynamic logic with an associated proof system; this forms the core of the KeYmaeraX [169] interactive theorem prover for hybrid systems verification. The Differential Temporal Dynamic Logics dTL [294] and dTL^2 [209] combine Differential Dynamical Logic with LTL temporal operators, whilst, $\text{STd}\mathcal{L}$ [4] is a recent dynamic logic for reasoning about a subset of STL properties. A similar proof-based approach applies the Hybrid Hoare Logic to verification of hybrid concurrent systems in the Hybrid CSP (HCSP) process calculus [243]. These approaches still require user involvement in proof construction, however, it is often able to automate proof steps using continuous and differential invariants. Over recent years a collection of increasingly powerful automated continuous invariant generation techniques have emerged including the decision procedure of Liu, Zhan, and Zhao [242] and the combination of methods implemented by [334]. More automatically still, Satisfiability Modulo Theory (SMT) solvers use a constraint solving based approach to handling uncertainty and perform parameter synthesis. The dReach hybrid systems verification tool uses the dReal SMT solver [175] to perform δ-complete safety verification, either returning `safe` if a safety property is robustly satisfied, or δ-`unsafe` if the system is within a δ-bounded perturbation of unsafety. Bae and Lee [18] recently introduced an approach to STL verification which translates properties into SMT problems which can be verified by Z3 [137] for linear systems or by dReal for nonlinear systems. Piazza et al. [292] developed earlier methods for exact temporal logic verification for TCTL properties of semi-algebraic hybrid systems based on real quantifier elimination [338].

2.4.2.3 Flowpipe Computation via State Abstractions

There is a long tradition of state-abstraction based methods which track the local evolution of the system similarly to numerical methods but, instead of using floating point numbers to compute a single system trajectory, carry out computations with computable representations of sets of states in order to quantify over all possible behaviours at every step of its evolution. This form of set-based reachability analysis produces a *flowpipe* $X_1, \ldots, X_n$ consisting of a sequence of representations X_i of sets of states which enclose every system trajectory $\mathbf{x}$ so that $\mathbf{x}(t) \in X_i$ for every $t \in [t_i, t_{i+1}]$. These flowpipes enclose all possible states of the system and so may be applied directly to reach-avoidance problems; the reach-avoidance property $\mathcal{G}_{[0,T]}(\mathbf{x} \notin B)$, is equivalent to verifying the set property $B \cap (X_1 \cup \ldots \cup X_n) = \varnothing$. The effectiveness of these methods relies on a form of state representation which is closed under a suitable range of mathematical operations, which permits efficient computations, and which is able to accurately track the evolution of a system's continuous dynamics.

This approach was pioneered by the HyTech [193] tool which uses a polyhedral state abstraction to calculate exact flowpipes for linear hybrid systems. A related approach is used by timed automata analysis tools such as UPPAAL [39] and KRONOS [72] which use representations such as Difference Bounded Matrices [147] for *zones* tracking the range of clock variables. In order to extend the applicability of hybrid reachability analysis, subsequent tools moved from exact reachability computation to over-approximate reachability computation which is tractably scalable to larger systems and more complex continuous dynamics; this approach was initially pursued by tools such as $\frac{\mathbf{d}}{\mathbf{dt}}$ [14], PHAVer [167], and CheckMate [112]. A substantial leap forward for reachability analysis of linear systems came with the use of a lazy state representation based on Zonotopes (affine transformations of boxes) [182] and support functions [237] which act as a symbolic remainder representation. This approach allows linear reachability analysis to scale to systems with hundreds of variables and is implemented in the SpaceEx tool [168] .

These methods have traditionally focused on linear continuous hybrid systems, and as such they use convex reachset representations which are ill-suited to track the flows of nonlinear systems without significant over-approximation. This over-approximation problem is amplified by the *wrapping effect* [247, 271] which causes over-approximation errors to compound throughout the dynamical evolution of a system. A major focus of recent research has been expanding reachability analysis to nonlinear systems. One approach is *hybridization* [192] which employs linear hybrid abstractions of nonlinear

continuous and hybrid systems which make it possible to apply linear reachability analysis whilst bounding the abstraction error via locality; this approach is employed by CORA [7], which implements a range of state representations and reachability methods as a MATLAB toolbox.

There is an alternative tradition of interval numerical methods [271], which extend traditional numerical methods to achieve sound over-approximations of system trajectories via interval arithmetic. This approach has been applied to hybrid systems verification by HyperTech which employs interval numerical methods to computing reachable sets [194]. However, plain interval arithmetic struggles to handle large uncertain intervals and complex continuous dynamics due to the *dependency problem* and the *wrapping problem*, both of which cause interval approximations (which can be seen as rectangular state abstractions) to blow up over time. These challenges motivated the introduction of symbolic interval methods such as Taylor models [45] which are able to directly represent non-convex sets of states. Implementations of Taylor models for continuous systems include COSY INFINITY [46] and CAPD::DynSys [220], whilst VNODES-LP [275] implements related interval Taylor series methods. Flow* [110] combines Taylor models with a number of other set-based methods to provide a leading reachability analysis tool for continuous and hybrid systems which is able to effectively handle large initial sets and complex nonlinear dynamics.

Most reachability methods have also been limited to bounded-time reachability, however, recent methods have proposed unbounded state abstractions [59, 60] or the combination of reachability analysis with continuous invariants [333] to perform unbounded-time reachability analysis.

2.4.3 Taylor models

We now introduce Taylor model arithmetic as a symbolic extension of interval arithmetic to tackle the *dependency problem*. Useful references include [45], [276], and [109, Chapter 2].

Definition 2.4.4. A k^{th}-order Taylor model with domain $\mathbf{D} = (D_1, \ldots, D_m) \in \mathbb{IR}^m$ is a pair (p, I) consisting of an k^{th}-order m-variable polynomial $p \in \mathbb{R}^n[\mathbf{x}]$ ranging over variables x_j with respective domains D_j and an remainder interval I.

More generally, we may define *Taylor model vectors* which represent n-dimensional vectors of Taylor models.

Definition 2.4.5. A k^{th}-order Taylor model vector with domain $\mathbf{D} = (D_1, \ldots, D_m) \in \mathbb{IR}^m$ is a pair $(\mathbf{p}, \mathbf{I})$ consisting of an n-dimensional vector $\mathbf{p}$ of k^{th}-order m-variable polynomials

$p_j \in \mathbb{R}^n[\mathbf{x}]$ ranging over variables x_j with respective domains D_j and an n-dimensional remainder interval $\mathbf{I}$.

Taylor models may be seen as 1-dimensional Taylor model vectors. Throughout this thesis we will work almost exclusively with Taylor model vectors in order to handle n-dimensional systems and henceforth we will refer to Taylor model vectors simply as Taylor models.

A Taylor model may then be treated as an interval function

$$(\mathbf{p}, \mathbf{I}) : \mathbf{D} \to \mathbb{IR}^n : \mathbf{y} \mapsto \mathbf{p}(\mathbf{y}) + \mathbf{I}.$$

We then say that $(\mathbf{p}, \mathbf{I})$ is a *Taylor model* for a real function $\mathbf{f} : \mathbf{D} \to \mathbb{R}^n$ if $(\mathbf{p}, \mathbf{I})$ is an interval extension of $\mathbf{f}$, or equivalently, if $\mathbf{f}(\mathbf{y}) \in \mathbf{p}(y) + \mathbf{I}$ for all $\mathbf{y} \in \mathbf{D}$. Thus, a Taylor model may be used either to over-approximate a function on its domain or to over-approximate a single set via its range $\mathbf{p}(\mathbf{D}) + \mathbf{I}$.

Example 2.4.6 (Zonotopes as Taylor models)**.** A key class of set representations used in linear reachability analysis are Zonotopes, which consist of affine transformations of the unit interval $[-1, 1]^n$ [182]. Zonotopes correspond exactly with first-order Taylor models [109, Lemma 2.4.8.], so that, for example, the Zonotope

$$\mathcal{Z} = \left\{ \begin{bmatrix} 3 & 2 \\ 1 & 4 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 7 \\ 6 \end{bmatrix} \;\middle|\; x \in [-1, 1], y \in [-1, 1] \right\}$$

(pictured in Fig. 2.1) can be represented as a Taylor model $(\mathbf{p}, \mathbf{I})$ with

$$\mathbf{p}\left(\begin{bmatrix} x \\ y \end{bmatrix} \right) = \begin{bmatrix} 3x + 2y + 7 \\ x + 4y + 6 \end{bmatrix}$$

remainder $\mathbf{I} = [0, 0] \times [0, 0]$ and domain $\mathbf{D} = [-1, 1] \times [-1, 1]$.

Taylor models may be also used to directly represent or over-approximate many other set representations used in reachability analysis [109].

The precision of the Taylor model over-approximation increases as the order of the Taylor model increases and as the size of the remainder interval decreases.

Addition of Taylor models may be defined piecewise, so that if $\mathbf{f}$ has Taylor model $(\mathbf{p}_f, \mathbf{I}_f)$ and $\mathbf{g}$ has Taylor model $(\mathbf{p}_g, \mathbf{I}_g)$ (both of order k with shared domain $\mathbf{D}_f = \mathbf{D} = \mathbf{D}_g$) then $\mathbf{f} + \mathbf{g}$ has Taylor model

$$(\mathbf{p}_f, \mathbf{I}_f) + (\mathbf{p}_g, \mathbf{I}_g) = (\mathbf{p}_f + \mathbf{p}_g, \mathbf{I}_f + \mathbf{I}_g).$$

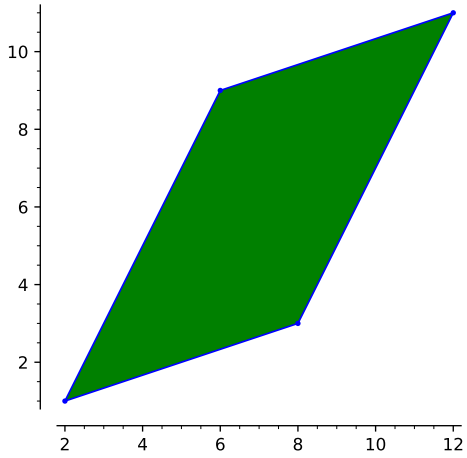


Figure 2.1: The Zonotope $\mathcal{Z}$.

For 1-dimensional Taylor models, we may also define arithmetic operations which generalize the operations of interval arithmetic. However, for these arithmetic operations we must truncate higher-order terms into the remainder interval in order to preserve the order of the Taylor model. For example, where f has Taylor model (p_f, I_f) and g has Taylor model (p_g, I_g), we define a k^{th}-order Taylor model for fg over domain $\mathbf{D}$ by

$$(p_f, I_f)(p_g, I_g) = (p_f p_g - \mathbf{p}_E, I_f g(\mathbf{D}) + f(\mathbf{D}) I_g + I_f I_g + p_E(\mathbf{D}))$$

where p_E contains all terms from $p_f p_g$ of order strictly greater than k.

It is then possible to symbolically apply an arbitrary polynomial p in k variables to a k^{th}-order Taylor model $\mathbf{v}$ by substituting each i^{th} dimension of $\mathbf{v}$ as the i^{th} variable of p. This forms the *symbolic composition* $p \,\square\, \mathbf{v}$ of $\mathbf{v}$ and p and stands in contrast to the standard *functional composition* $p \circ \mathbf{v}$ of a Taylor model $\mathbf{v}$ with a polynomial p as interval functions. This extends naturally to the symbolic composition $(\mathbf{p}_g, \mathbf{I}_g) \,\square\, (\mathbf{p}_f, \mathbf{I}_f)$ and functional composition $(\mathbf{p}_g, \mathbf{I}_g) \circ (\mathbf{p}_f, \mathbf{I}_f)$ of two Taylor model $(\mathbf{p}_f, \mathbf{I}_f)$ and $(\mathbf{p}_g, \mathbf{I}_g)$ provided that $\mathbf{p}_f(\mathbf{D}_f) + \mathbf{I}_f \subseteq \mathbf{D}_g$; both of these composed Taylor model are guaranteed to be an interval extension for the composition $\mathbf{g} \circ \mathbf{f}$ of the underlying functions.

It is further possible to evaluate arbitrary elementary functions over Taylor models using Taylor series expansions as described in [252]; this forms the basis of the application of Taylor model methods to non-polynomial systems.

2.4.4 Verified Integration and Flowstar

Given an uncertain continuous system Definition 2.4.1 with a m-dimensional set of initial conditions given as the range of a l^{th} order Taylor model $X_0 = (\mathbf{p}_0, \mathbf{I}_0)$ with an

l-dimensional interval *space domain* $\mathbf{S} = \mathbf{D}_0 \subseteq \mathbb{IR}^l$, Taylor model verified integrators [47] are able to enclosure all trajectories over a bounded time-horizon $[0, T]$ in a *Taylor model flowpipe* consisting of the Taylor models,

$$(\mathbf{p}_1, \mathbf{I}_1), (\mathbf{p}_2, \mathbf{I}_2), \ldots, (\mathbf{p}_n, \mathbf{I}_n)$$

each of which has domain $\mathbf{D}_k = T_k \times \mathbf{S}$ composed from the interval time domain $T_k = [t_k, t_{k+1}]$ and the space domain $\mathbf{S}$. The overall flowpipe may be treated as an interval function $f : [0, T] \times \mathbf{S} \to \mathbb{IR}^m$ defined by $f(t, \mathbf{u}) = \mathbf{p}_k(t, \mathbf{u}) + \mathbf{I}_k$ where $t \in [t_k, t_{k+1}]$.

Beyond forming a flowpipe for the system comprised of the sets defined by the ranges of each Taylor model, a Taylor model flowpipe in fact gives the more granular guarantee that

$$\mathbf{x}_\mathbf{u}(t) \in f(t, \mathbf{u})$$

where $\mathbf{x}_\mathbf{s}(t)$ is any solution to the system satisfying initial condition $\mathbf{x}_\mathbf{u}(0) \in (\mathbf{p}_0, \mathbf{I}_0)(\mathbf{s})$. That is, a Taylor model flowpipe represents a *functional dependency* of the eventual behaviour of the system on its initial conditions.

Taylor model flowpipe construction has been extended first to uncertain systems with time-invariant uncertain parameters by Lin and Stadtherr [241] and then to time-varying uncertain inputs in Flow* [109, 111]. In Flow* these time-varying uncertain inputs are input as systems of coupled ODEs with intervals appearing in the right-hand side; these intervals are then interpreted as the uncertain inputs to the system. However, the time-varying inputs pose significant additional challenges to tight flowpipe computation due to the fresh non-determinism introduced into the system's behaviour at every timepoint, and in this case we are not given a functional dependency of system's evolution on input parameters.

Advanced Taylor model integrations such as COSY INFINITY [46] and Flow* [110] have extended this basic scheme to increase the precision of verified integration results and to better handle complex continuous dynamics and large uncertain sets.

Preconditioning The technique of *preconditioning* was introduced by Makino and Berz [251] to reduce the wrapping effect on Taylor model flows by decomposing or preconditioning the initial set as a symbolic composition of two Taylor models,

$$X_0 = (\mathbf{p}^{(0)}, \mathbf{I}^{(0)}) = \left(\mathbf{p}^{(0)}_{\text{post}}, \mathbf{I}^{(0)}_{\text{post}}\right) \square \left(\mathbf{p}^{(0)}_{\text{pre}}, \mathbf{I}^{(0)}_{\text{pre}}\right)$$

where the space domain of $\left(\mathbf{p}^{(0)}_{\text{pre}}, \mathbf{I}^{(0)}_{\text{pre}}\right)$ is the unit box $[-1, 1]^m$. Taylor model integration is then extended to produce a *preconditioned Taylor model flowpipe* consisting of a sequence

of preconditioned Taylor models,

$$\left(\mathbf{q}^{(k)}, \mathbf{I}^{(k)}\right) = \left(\mathbf{q}^{(k)}_{\text{post}}, \mathbf{I}^{(k)}_{\text{post}}\right) \square \left(\mathbf{q}^{(k)}_{\text{pre}}, \mathbf{I}^{(k)}_{\text{pre}}\right).$$

This split allows preconditioned Taylor models to more accurately track complex flows, with the first half of the preconditioning acting as a transformed coordinate system for the local dynamics of the system [251]. However, it makes working with flowpipes considerably more computationally challenging. For example, before most kinds of analysis can be carried out (including plotting, reach-avoidance checking, hybrid system jump handling, and interval evaluation of solutions) Flow* must first compose the two halves of the preconditioned Taylor models in a flowpipe or compute a polytope over-approximation of the flowpipe.

Range Bounding and Horner Forms In order to make use of Taylor models bounding the solutions of a system, we need effective ways to bound the range of a Taylor model as an interval or more general polytope. The most obvious way of doing so is interval evaluation of the Taylor model over its domain, however, due to the dependency problem interval evaluation of polynomials can suffer excessive over-approximation errors to a degree which depends heavily on the structure of the polynomial. Flow* attempts to reduce the degree of over-approximation by placing polynomials into a multi-variate Horner form [109] which minimizes the number of interval operations required for interval evaluation; this transformation does, however, come with a significant computational cost.

We note that since the ability of Taylor models to handle the dependency problem is reliant on handling uncertainty symbolically for as long as possible when bounding the range of a composition of Taylor models

$$(\mathbf{q}_g, \mathbf{I}_g) \square (\mathbf{q}_f, \mathbf{I}_f),$$

we will generally get tighter bounds from first performing the symbolic composition and then transforming the overall result to Horner form, than individually bounding each half to compute the functional composition

$$(\mathbf{q}_g, \mathbf{I}_g) \circ (\mathbf{q}_f, \mathbf{I}_f).$$

Symbolic Remainder Estimation The ability of Taylor model integration to handle a given system is limited by its ability to prevent uncertainties from migrating from the symbolic proportion of the flowpipe to the remainder intervals. This becomes a particular

problem in systems featuring time-varying uncertain parameters which are not covered by the space domain of the Taylor model. Flow* has introduced Taylor models with symbolic remainders represented via support functions [109, 111], which are able to significantly reduce the growth of remainder intervals and improve Flow*'s practical support for time-varying uncertain parameters.

Adaptive Step-Sizes and Orders One of the main challenges of applying Taylor model based integration is the number of parameters such as step-size, Taylor model order, and remainder cutoff-threshold which need to be tweaked to allow the process to be applied in a reasonable time to a given system, without the error being so large as to render the result useless. Flow* aids in this by allowing the user to specify a range of step sizes or Taylor model orders. It then dynamically changes the order throughout the integration process in order to maintain a given error tolerance at each step.

2.4.5 From Reachability to Temporal Logic Model Checking

Whilst set-based reachability analysis is effective at verifying reach-avoidance problems, to extend this to general temporal logic model checking one must tackle problems arising from the coarseness of flowpipes compared to individual simulations trajectories. Flowpipes can provide only limited timing information for checking bounded temporal properties whilst their over-approximate nature means that the truth value of a given temporal logic proposition is often underspecified.

Recently, a number of approaches have been proposed to implement exact temporal logic model checking using different types of set-based reachability analysis for continuous and hybrid systems. Bresolin developed a method for encoding LTL properties of hybrid systems as reachability properties of hybrid automata by constructing monitoring automata similarly to the automata theoretic approach of monitoring timed systems [352]. This allows the reuse of existing reachability tools, however, this does not included temporal operators with time intervals and applying this to even relatively simple LTL properties results in automata too large for current nonlinear reachability tools to handle. Cimatti et al. [115] proposed an approach to reducing LTL model checking to reachability analysis via k-liveness. Rather than directly encoding properties as reachability problems, Roehm et al. [315] proposed offline analysis of the flowpipes produced reachability analysis tools, and developed a method of STL verification based on checking a time-sampled version of STL formulae over the polytope flowpipes produced by the CORA [7] reachability analysis

tool. On the other hand, the Sapo tool [152] implements STL model checking and parameter synthesis for discrete-time polynomial dynamical systems based on a parallelotope bundle representation of flowpipes [153].

The work most closely related to our approach is that of Ishii and Goldsztejn, who proposed a method using interval root finding to monitor verified signals over interval extensions of solutions produced via verified integration [206–208]. The methods proposed in Chapter 6 build upon their methods of monitoring atomic propositions whilst our representation of signals is different (representing partial signals using three-valued logic rather than inner/outer approximations of intervals). Our methods also integrate tightly with the specific structure of Flow* flowpipes to improve the efficiency and precision of monitoring, whilst their method targets generic interval extensions of solutions, evaluated in a black-box manner.

The underspecifity of flowpipes poses problems to compositional reasoning when using a Boolean temporal logic semantics since, in a signal monitoring approach, an over-approximate Boolean signal cannot be soundly negated. Moreover, even when uncertainty is represented explicitly, a logical form of the *dependency problem* of interval arithmetic can cause uncertainties to compound throughout the semantics. Another potential pitfall is that in decoupling flowpipe computation and evaluation from the monitoring process, these methods are not able to direct monitoring of atomic propositions to the time regions most relevant to verifying a given property. This is especially relevant in the case of Flow* flowpipes which need to be preprocessed (by symbolically or functionally composing both halves of a preconditioned Taylor model). In this thesis we propose methods to achieve a more *property-directed* monitoring process over Flow* flowpipes using three-valued signals, on-demand symbolic composition of flowpipes, adaptive symbolic and physical subdivision of unknown initial conditions, and the use of *masks* to direct monitoring to relevant regions of the combined space-time domain.

Chapter 3

Bond-Calculus

In this chapter we introduce the *bond-calculus*, a novel process algebra for modelling biological systems. The bond-calculus aims to capture the full range of continuous biological interactions at the molecular and network levels. To this end, the language introduces a novel multi-way communication mechanism which models the concept of affinity between molecular interactions sites by allowing processes to communicate at named sites whose quantitative interaction dynamics are governed by a network of affinity patterns. As we will explore through several case studies in Chapter 4, the flexibility of this communication mechanism allows us to more effectively capture the vast diversity of communication patterns in real biological systems through a range of different modelling styles. At the molecular level, the bond-calculus utilizes mobility of "location names" (identifying individual bonds or linkages within a molecular complex) making it possible to model dynamic bonding similarly to π-calculus–based languages [305, 310] whilst it is able to specify allosteric interactions within a molecule through multi-way communication and pattern matching. Meanwhile, at the network-level the combination of multi-way communication and functional affinity rate laws allows us to capture the full range of biochemical reactions including multi-way interactions and general kinetic laws.

Once we have specified the formal syntax of the bond-calculus, our main focus will be developing a compositional continuous semantics for the language. As part of this we develop a transition semantics to capture interactions within a single molecular species and then on top of this we introduce a continuous vector space semantics for mixtures of species. In order to achieve compositionality, we define a number of operators to extract the multi-molecular interactions between species, as well a novel method of extracting multi-molecular interactions within a single species via an *interaction exponential*. This semantics then allows us to compositionally derive the interaction dynamics of different

mixtures of processes/species, and ultimately, systems of differential equations or Chemical Reaction Networks (CRNs) to allow a system to be simulated.

The structure of this Chapter is as follows. Section 3.1 gives a brief overview of the bond-calculus before Section 3.2 introduces the formal sequences of each of the different language constructs. Section 3.3 covers the semantics of the bond-calculus, focusing first on the transition semantics for individual species/molecules in Section 3.3.1, before providing a continuous semantics for mixtures of species in Section 3.3.2 and for affinity networks in Section 3.3.3 before Section 3.3.4 establishes the compositionality of this semantics. Finally, Section 3.4 shows how this semantics defines the dynamics of a model. We first how to extract systems in the form of CRNs in Section 3.4.1. We then show how the semantics may be used to define a vector field capturing the continuous interaction dynamics in Section 3.4.2, before showing how this allows us to extract systems of differential equations in Section 3.4.3.

An initial version of the bond-calculus was first introduced in the author's master's dissertation [358]. The version presented here has been substantially extended with a reformulated communication mechanism based on multi-way pattern matching, a new semantics, and support for stochastic simulation via Chemical Reaction Network extraction. Later on, Chapter 5 further extends the bond-calculus to add support for uncertain systems using interval arithmetic.

3.1 Language Overview

Before introducing the formal syntax and semantics of the bond-calculus, we will start with a high-level description of the language as it relates to biochemical modelling[1]. The chief component of a bond-calculus model is a *mixture*:

$$\Pi \triangleq \alpha_1 A_1 \parallel \ldots \parallel \alpha_n A_n \,.$$

This represents a chemical solution of different *species* $A_1, \ldots, A_n$ at real-valued concentrations $\alpha_1, \ldots, \alpha_n \in \mathbb{R}_{\geq 0}$ respectively. Species are described by process-algebraic terms that indicate their potential behaviour. In particular, species offer interaction at certain *sites* $s, e, p, \ldots$, which may also be annotated with *locations* $\ell, m, \ldots$ that indicate spatial proximity on a molecular complex.

[1] As we will see in subsequent examples, many of the language's constructs can also be interpreted appropriately in other biological modelling domains such as ecology.

The declaration of which sites are compatible and the quantitative rates of interaction between them appears in a separate *affinity network*,

$$\mathcal{A} \triangleq \left\{ \boldsymbol{\gamma}_1 \mathbin{@} \mathrm{L}_1, \ldots, \boldsymbol{\gamma}_n \mathbin{@} \mathrm{L}_n \right\}$$

which is made up of *affinity patterns* $\boldsymbol{\gamma} \mathbin{@} L$. Each affinity pattern combines a *pattern* $\boldsymbol{\gamma}$ of reaction sites with a general kinetic *law* $\mathrm{L} : \mathbb{R}^m \to \mathbb{R}$ for the corresponding reaction rate. Patterns themselves are structured $\boldsymbol{\gamma} = p_1 \parallel \ldots \parallel p_m$ where each component is either a single site $p_i = s$, or a *cluster* of multiple colocated sites $p_i = (s_1 | \ldots | s_{t_i})$. For example, a simple pattern $a \parallel b \parallel c$ allows chemical reactions to occur between three distinct molecules presenting sites a, b, and c respectively, whilst the pattern $a \parallel (b|c)$ allows reactions between a molecule with site a and another molecule with both sites b and c at a shared location, and a pattern $(a|b|c)$ allows unimolecular reactions involving a, b, and c all at a shared location on a single molecule[2].

The reactants themselves are described by a number of *species* $A, B, C, \ldots$ whose intrinsic behaviour is defined via process-algebraic definitions $A \triangleq \ldots$. For example, we may define two species,

$$A \triangleq a(\ell_1).a^*@\ell_1.A \qquad \text{and} \qquad B \triangleq b(\ell_2).b^*@\ell_2.B.$$

These definitions state that, for example, A may engage in a reaction at site a to become a new species $a^*@\ell_1.A$, involving a site a^* which is located at the new internal molecular location ℓ_1. Reactions between species may involve bonding of sites to dynamically generate new species. In this case, if sites a and b are compatible, the species A and B may react to form a bimolecular complex

$$C \triangleq (\nu\, \ell)(a^*@\ell.A \mid b^*@\ell.B)\,.$$

The complex species is defined dynamically, using the location binder $(\nu\, \ell)(\ldots)$ (similar to name abstraction in the λ-calculus and π-calculus) to denote the shared location which binds the sites a^* and b^* at each half of the complex. To form this new complex, the locations ℓ_1 and ℓ_2 are merged into a single location ℓ in a form of simultaneous agreement inspired by the πI-calculus [322]. This form of communication models the symmetric nature of molecular bonding, and extends naturally to multiway reactions.

Suppose we want to complete this with a description of quantitative reaction rates following the *law of mass action*, defined by

$$\mathrm{MA}_k(x_1, \ldots, x_1) \triangleq k x_1 \ldots x_n.$$

[2]The ability of a cluster of sites at a shared location within a molecule to jointly participate in reactions models composite molecular binding sites and allosteric interactions between sites.

We can use affinity pattern $a \parallel b \mathbin{@} \mathrm{MA}_{k_1}$ (or, $a \parallel b \mathbin{@} k_1$ for short) to declare that sites a and b are compatible and interact at rate $\mathrm{MA}_{k_1}([a], [b]) = k_1[a][b]$ where $[a]$ and $[b]$ are the total concentrations of species carrying sites a and b respectively — the sites in the affinity pattern are associated to the arguments of the rate law in a positional manner indicating that in this case the concentrations of a and b correspond to the first and second arguments of the rate law respectively. These site concentrations are interpreted as unitless site concentrations (so that the reaction rates are independent of cell volume V) and kinetic laws are specified under the assumption that each site is on an independent chemical species; as later demonstrated in Example 3.3.12 the bond-calculus semantics will automatically adjust the rate law to take into account multiple instances of a single site occurring in the same species to apply a suitable combinatoric scaling coefficient similarly to [51, 94].

Finally, we can specify the initial state of the system as a mixture $\Pi \triangleq [A]\, A \parallel [B]\, B$ consisting of concentration $[A]$ of species A and concentration $[B]$ of species B. Thus a complete bond-calculus model is made up of a mixture Π of species, which are defined via a number of species definitions, and an affinity network $\mathcal{A}$ consisting of different affinity patterns, whose rate laws are defined via rate law definitions.

Models in the language can be translated into a number of other different forms of mathematical models based on the semantics of the language. These methods have been implemented in the `bondwb` tool described in Section 9.1 which can perform numerical simulation of extracted ODEs or stochastic simulation of extracted Chemical Reactions Networks via the StochPy [250] library.

For example, in the mixture Π specified above, A offers site a and B offers site b: which are compatible according to the pattern $a \parallel b$; this results in the reaction, $A \parallel B \to^{k_1} C$ which consumes species A and B whilst producing species C at overall rate $\mathrm{MA}_{k_1}([A], [B]) = k_1[A][B]$, or the differential equations

$$\frac{\mathrm{d}[C]}{\mathrm{d}t} = -\frac{\mathrm{d}[A]}{\mathrm{d}t} = -\frac{\mathrm{d}[B]}{\mathrm{d}t} = k_1[A][B] \ .$$

Similarly, we can introduce an unbinding reaction $C \to^{k_{-1}} A \parallel B$ by using the affinity pattern $(a^*|b^*) \mathbin{@} \mathrm{MA}_{k_{-1}}$, where sites a^* and b^* are now colocated on C.

3.2 Syntax

We will now define the formal syntax of the bond-calculus. First we define rate laws which specify the rates governing reactions, and then sites, locations, and affinity networks,

allowing us to specify the types of reactions taking part in a system. Next we define the two levels of the calculus of agents: the species level specifying the behaviour of individual agents, and the mixture level specifying mixtures of different concentrations of each species. Finally, we define a structural congruence relation, prime species, and normal forms, allowing us to identify equivalent species and give a unique representation of a mixture.

3.2.1 Rate Laws

We first define rate laws, which form the basis of our support for general kinetics. A rate law $\mathrm{R} : \mathbb{R}^n \to \mathbb{R}$ gives the rate $\mathrm{R}(x_1, \ldots, x_n)$ of a reaction based on the concentration of each of its arguments $x_1, \ldots, x_n$. We allow arbitrary mathematical functions as kinetic laws and give examples in standard mathematical notation.

Definition 3.2.1 (Rate law)**.** A rate law is a function $\mathrm{R} : \mathbb{R}^n \to \mathbb{R}$ which takes a list of n real-valued concentrations to a real-valued rate.

Definition 3.2.2 (Rate law family)**.** A rate law family is a function $\mathrm{R} : \mathbb{R}^m \to \mathbb{R}^n \to \mathbb{R}$, which takes a list $\mathbf{k}$ of m rate law parameters and returns a rate law $\mathrm{R}_{\mathbf{k}} \triangleq \mathrm{R}(\mathbf{k})$.

A slightly more exotic example is given by the HBr formation process.

Example 3.2.3 (Hydrogen Bromide formation)**.** Hydrogen Bromide (HBr) may be formed from Hydrogen (H_2) and Dibromide (Br_2) via the linear chain reaction shown in Figure 3.1a [56]. This reaction can be modelled as a single ternary reaction (shown in Figure 3.1b) with rate law given by,

$$\mathrm{R}_k([\mathrm{H_2}], [\mathrm{Br_2}], [\mathrm{HBr}]) = \frac{[\mathrm{H_2}] [\mathrm{Br_2}]^{1/2}}{1 + k\frac{[\mathrm{HBr}]}{[\mathrm{Br_2}]}}.$$

We will return to this example throughout this section to illustrate many features of the language and its semantics.

3.2.2 Sites, Locations, and Affinity Networks

As the basis of a model we define the set SITE of *site names*, along with the set LOC of *location names*. Sites represent (chemical) reaction sites and determine when molecules may react, whilst locations refer to internal locations within a molecule. We distinguish a special location $\top \in \mathrm{LOC}$, the *ambient location*, referring to the top level of a mixture. The

$$\begin{aligned} \mathrm{Br_2} &\longrightarrow \mathrm{Br} + \mathrm{Br} \\ \mathrm{Br} + \mathrm{H_2} &\longrightarrow \mathrm{H} + \mathrm{HBr} \\ \mathrm{H} + \mathrm{Br_2} &\longrightarrow \mathrm{Br} + \mathrm{HBr} \\ \mathrm{Br} + \mathrm{Br} &\longrightarrow \mathrm{Br_2} \end{aligned}$$

(a) Linear chain of mass action reactions.

$$\mathrm{H_2} \quad + \quad \mathrm{Br_2} \quad \xrightarrow{\ r\ } \quad 2\,\mathrm{HBr}$$

$$r = \frac{[\mathrm{H_2}]\,[\mathrm{Br_2}]^{1/2}}{1 + k\frac{[\mathrm{HBr}]}{[\mathrm{Br_2}]}}$$

(b) Composite reaction and rate r.

Figure 3.1: Hydrogen Bromide formation process.

$$\begin{array}{ccccc} \textsc{Site} & & \textsc{Cluster} & & \textsc{Pattern} \\ s & \subseteq & \gamma = s_1 | \ldots | s_n & \subseteq & \boldsymbol{\gamma} = \gamma_1 \parallel \ldots \parallel \gamma_n \end{array}$$

Figure 3.2: Sites, clusters, and patterns.

set LocSite consists of located sites $\pi = s@\ell$ where s is a site and ℓ is a location. Then two located sites $s@\ell$ and $t@m$ can interact allosterically if they are at the same location $\ell = m$, however, sites at different molecular locations may still engage in intermolecular reactions. A site $s@\top$ at the ambient location $\top$ is called an *ambient site* and we establish the shorthand $s \triangleq s@\top$; an ambient site is not involved in allosteric interactions, but can be involved in intermolecular reactions.

In order to determine the dynamics of a system we will need to know which sites may react with which, and associate kinetic laws to these reactions. Reactions in the bond-calculus are specified by pattern matching. Clusters $\gamma = s_1 | \ldots | s_n$ match against a collection of sites in the same molecule (and specify allosteric interactions), whilst patterns $\boldsymbol{\gamma} = \gamma_1 \parallel \ldots \parallel \gamma_n$ match against solutions of molecules (in which each molecule must contain the corresponding cluster of sites). Any site can be treated as a trivial cluster, and any cluster can be treated as a trivial pattern as shown in Figure 3.2.

Definition 3.2.4. A *cluster* $\gamma \in$ Cluster consists of a bag $\wr s_1, \ldots, s_n \wr$ of sites $s_1, \ldots, s_n \in$ Site, and we write $\gamma = s_1 | \ldots | s_n$.

Definition 3.2.5. A *pattern* $\boldsymbol{\gamma} \in$ Pattern consists of a bag $\wr \gamma_1, \ldots, \gamma_n \wr$ of clusters $\gamma_1, \ldots, \gamma_n \in$ Cluster, and we write $\boldsymbol{\gamma} = \gamma_1 \| \ldots \| \gamma_n$. An *ordered pattern* $\boldsymbol{\gamma} = \gamma_1 \| \ldots \| \gamma_n \in$ Ordered-Pattern, is a pattern which also records the order of the site patterns, and is represented by a list of clusters $(\gamma_1, \ldots, \gamma_n)$.

Then the reactions which can occur in a given system are specified by an affinity

network $\mathcal{A}$ which specifies a pattern for each reaction which can occur, along with its rate law.

Definition 3.2.6 (Affinity network)**.** An *affinity network*,

$$\mathcal{A} = \left\{ \boldsymbol{\gamma}^{(1)} \mathbin{@} \mathrm{L}_1, \ldots, \boldsymbol{\gamma}^{(n)} \mathbin{@} \mathrm{L}_n \right\} \in \textsc{Aff} \triangleq \mathcal{P}(\textsc{Ordered-Pattern} \times [\mathbb{R}^* \to \mathbb{R}])$$

consists of a set of ordered patterns $\boldsymbol{\gamma}^{(i)} = \gamma_1^{(i)} \parallel \ldots \parallel \gamma_m^{(i)}$ together with rate laws L_i (where $\mathcal{P}(X)$ denotes the powerset of X and $\mathbb{R}^* = \bigcup_{m=0}^{\infty} \mathbb{R}^m$).

In most contexts patterns do not have a fixed order so $\boldsymbol{\gamma} \parallel \boldsymbol{\delta} = \boldsymbol{\delta} \parallel \boldsymbol{\gamma}$, however, in an affinity network the order of the pattern specifies the order of the arguments of the rate law (given many nonlinear rate laws are not commutative). Then each term $\boldsymbol{\gamma} = \gamma_1 \parallel \ldots \parallel \gamma_n \mathbin{@} \mathrm{L}$ of the affinity networks specifies a reaction involving n (not necessarily distinct) species of molecules containing interactions with site clusters $\gamma_1, \ldots, \gamma_n$ at reaction rate $\mathrm{L}([\gamma_1], \ldots, [\gamma_n])$. We also allow the shorthand $\boldsymbol{\gamma} \mathbin{@} k$ (where $k \in \mathbb{R}$) for the Mass-Action pattern $\boldsymbol{\gamma} \mathbin{@} \mathrm{MA}_k$.

Example 3.2.7. The HBr formation reaction has affinity network,

$$\mathcal{A} \triangleq \left\{ h \parallel b \parallel h^* | b^* \mathbin{@} \mathrm{R}_k \right\}$$

where the rate law R_k is as defined in Example 3.2.3.

3.2.3 Species and Abstractions

We will now define the syntax rules for species in the language. In order to handle communication prefixes which bind locations such $x(\ell_1, \ldots, \ell_n).A$, we will present the calculus in an *abstraction-concretion* style following [268] (although, given the symmetry of our communication operator, we only have abstractions) and similarly to Continuous π [234], so in addition to species A we define *abstractions* $F \triangleq (\ell_1, \ldots, \ell_n)S$ where the variables $\ell_1, \ldots, \ell_n$ bind locations in S. Then we can treat prefixes such as $x(\ell_1, \ldots, \ell_n).A$ as syntactic sugar for $x.(\ell_1, \ldots, \ell_n)A$ and thereby restrict our attention to simple communication prefixes of form $x@\ell.F$ to simplify the presentation of the language. So, we start by giving the following grammar for *species*:

$$A, B \ ::= \ \mathbf{0} \ \mid \ \pi_1.F_1 + \ldots + \pi_n.F_n \ \mid \ A \mid B \ \mid \ (\nu\, \ell_1, \ldots, \ell_n)A \ \mid \ D(s_1, \ldots, s_n; \ell_1, \ldots, \ell_m)$$

That is, a species can be any of:

- The *null species* $\mathbf{0}$. This does exactly nothing, and represents an agent which attempts no communication actions.

- A *choice* $\pi_1.F_1 + \ldots + \pi_n.F_n$ of one of n abstractions $F_1, \ldots, F_n$ guarded by the *prefixes* $\pi_1 = s_1@\ell_1, \ldots, \pi_n = s_n@\ell_n \in \textsc{LocSite}$. This means that the species can evolve into one of the abstractions F_i, but only after the synchronisation corresponding to the prefix given by the (potentially) located site $\pi_i = s_i@\ell_i$ has occurred. We define the empty choice as $\mathbf{0}$.

- A *parallel composition* $A \mid B$ of two species A and B. In this species any of the evolutions of A or B can occur in parallel and they can communicate with each other.

- A *location binding* (or *restriction*) $(\nu\, \ell_1, \ldots, \ell_n)A$ of a set $\boldsymbol{\ell} = \{\ell_1, \ldots, \ell_n\}$ of locations in the species A which we may equivalently write as $(\nu\boldsymbol{\ell})A$. This marks the specified locations as local to A.

- A *definition application* $D(x_1, \ldots, x_n; \ell_1, \ldots, \ell_m)$, which applies the definition of the species D, supplying as arguments a list of site names $x_1, \ldots, x_n$ and a list of location names $\ell_1, \ldots, \ell_m$.

The case for definition applications allows us to define the species of a system as a list of mutually recursive definitions,

$$D(x_1, \ldots, x_n; \ell_1, \ldots, \ell_m) \triangleq P$$

where $x_1, \ldots, x_n$ bind site names in P, and $\ell_1, \ldots, \ell_m$ bind location names in P. We will often write $D_{\ell_1,\ldots,\ell_n}$ as shorthand for $D(; \ell_1, \ldots, \ell_n)$. Unlike in many presentations of the π-calculus [267] we use recursive definitions rather than a replication operator ! since this will produce much more readable biological models, and moreover unlike in the full π-calculus where the two operators are equivalent, in calculi with only internal mobility such as πI recursion can be strictly more expressive than replication [85, 283, 322].

Next, we define abstractions according to the following grammar:

$$F, G \;::=\; (\ell_1, \ldots, \ell_n)A.$$

There is only one form of abstraction, since we list all of the abstracted location names at the top level, however, we will now define operators to lift parallel composition and name restriction from the process level to the abstraction level.

Composition of abstractions implements our simultaneous agreement based communication mechanism by unifying and then merging the location names of two abstractions to allow the underlying species to be composed.

Definition 3.2.8. The *parallel composition* or *colocation* of abstractions $(\ell_1, \ldots, \ell_p)A$ and $(\ell_1, \ldots, \ell_q)B$ is defined by,

$$(\ell_1, \ldots, \ell_p)A \mid (\ell_1, \ldots, \ell_q)B = (\ell_1, \ldots, \ell_s)(A \mid B),$$

where $s = \max\{p, q\}$. This defininition is then extended to arbirary pairs of abstractions $(\ell_1, \ldots, \ell_p)A$ and $(m_1, \ldots, m_q)B$ by applying α-renaming to place them in the above form by making their respective location names agree.

Definition 3.2.9. The *restriction* of names $\ell_1, \ldots, \ell_p$ in an abstraction $(m_1, \ldots, m_q)A$ is defined by,

$$(\nu\, \ell_1, \ldots, \ell_p)(m_1, \ldots, m_q)A = (m_1, \ldots, m_q)(\nu\, \ell_1, \ldots, \ell_p)A$$

where location names $\ell_1, \ldots, \ell_p$ and $m_1, \ldots, m_q$ are assumed to be distinct by α-renaming.

We will also freely allow a species A to be embedded as a trivial abstraction $()(A)$, and we can see that in this case these definitions reduce to parallel composition and restriction of species. The parallel composition operation on abstractions is similar to the *pseudo-application* operation in the abstraction-concretion presentation of the π-calculus [234, 268], but for us plays the role of combining the contributions of two parties in a reaction into a complex.

An abstraction represents the products of an open reaction, and is expanded via colocation as more reactants join the reaction. Once all the reactants have joined, an abstraction can be *committed*, giving the resulting species.

Definition 3.2.10. The *committed product* of an abstraction F is the species given by,

$$\mathrm{commit}((\ell_1, \ldots, \ell_n)A) \triangleq (\nu\, \ell_1, \ldots, \ell_n)A.$$

3.2.4 Mixtures

Multiple species may be combined into a mixture, representing a chemical solution of different concentrations of each pure species. Mixtures are specified according to the grammar:

$$P, Q \;::=\; c \cdot S \;\mid\; P \parallel Q$$

that is, a mixture is defined recursively as either:

- $c \cdot S$, meaning species S is present in concentration $c \in \mathbb{R}$.
- The *parallel composition* or *solution* $P \parallel Q$ of two processes P and Q.

As shorthand we may omit the dots and write $c_1 \cdot A_1 \parallel \ldots \parallel c_n \cdot A_n$ as $c_1 A_1 \parallel \ldots \parallel c_n A_n$.

3.2.5 Abstract Syntax and Prime Species

In the preceding part of this section we defined a formal grammar specifying the concrete syntax of the language. However, this syntax allows a lot of redundancy, allowing us to write a given species in many equivalent ways. A simple example of this is α-equivalence: the species $(\nu\,\ell)(S^*_\ell \,|\, E^*_\ell)$ and $(\nu\,m)(S^*_m \,|\, E^*_m)$ since they differ only in the name of the bound location ℓ, and we can express this type of equivalence as $(\nu\,\ell)(S^*_\ell \mid E^*_\ell) \equiv_\alpha (\nu\,m)(S^*_m \mid E^*_m)$. We need some way of identifying these equivalent processes in order to practically implement the system, since separately tracking many equivalent copies of the same species would result in a much larger (and frequently infinite) state space.

We resolve this issue by defining a *structural congruence* relation $\equiv$ on species, abstractions, and mixtures which tells us when two systems are considered syntactically equivalent. This is semantics preserving equivalence relation which includes α-equivalence and a number of other equivalences specific to the language (many of these coincide with the structural congruence relations for other variants of the π-calculus [234, 267, 303]). When two species are structurally congruent they are considered to be different concrete syntax for the same abstract species and they may be interchanged freely; likewise the corresponding structural congruences allow us to interchange equivalent abstractions and mixtures respectively.

Definition 3.2.11. The structural congruence $\equiv$ on species is the least congruence containing α-equivalence and satisfying the following axioms:

$$
\begin{aligned}
\mathbf{0} \mid A &\equiv A \\
A \mid B &\equiv B \mid A \\
(A \mid B) \mid C &\equiv A \mid (B \mid C) \\
\textstyle\sum_{i=0}^{n} \pi_i.A_i &\equiv \textstyle\sum_{i=0}^{n} \pi_{\sigma_i}.A_{\sigma_i} && \text{given a permutation } \sigma \\
(\nu\,\boldsymbol{\ell} \cup \boldsymbol{m})F &\equiv (\nu\,\boldsymbol{\ell})(\nu\,\boldsymbol{m})F \\
(\nu\,\boldsymbol{\ell})F &\equiv F && \text{given } \boldsymbol{\ell} \cap \operatorname{flocs}(F) = \varnothing \\
(\nu\,\boldsymbol{\ell})(A \mid B) &\equiv A \mid (\nu\,\boldsymbol{\ell})B && \text{given } \boldsymbol{\ell} \cap \operatorname{flocs}(A) = \varnothing
\end{aligned}
$$

where flocs(F) denotes the set of free (that is, unbound by ν-binders) locations which occur in a species.

Definition 3.2.12. The structural congruence $\equiv$ on abstractions is the least congruence containing α-equivalence and satisfying the following axioms:

$$(\ell_1, \ldots, \ell_{n-1}, \ell_n)A \equiv (\ell_1, \ldots, \ell_{n-1})A \qquad \text{given } \ell_n \notin \text{flocs}(A)$$
$$(\ell_1, \ldots, \ell_n)A \equiv (\ell_1, \ldots, \ell_n)B \qquad \text{given } A \equiv B$$

Definition 3.2.13. The structural congruence $\equiv$ on processes is the least congruence containing α-equivalence and satisfying the following axioms:

$$\begin{aligned}
(c \cdot \mathbf{0}) \parallel P &\equiv P \\
P \parallel Q &\equiv Q \parallel P \\
(P \parallel Q) \parallel R &\equiv P \parallel (Q \parallel R) \\
(c + d) \cdot A &\equiv (c \cdot A) \parallel (d \cdot A) \\
a \cdot (A \mid B) &\equiv a \cdot A \parallel a \cdot B \\
c \cdot A &\equiv c \cdot B \qquad \text{given } A \equiv B
\end{aligned}$$

One particularly interesting case of this congruence is $a \cdot (A|B) \equiv a \cdot A \parallel a \cdot B$, which says that if a species (molecule) $S = A \mid B$ can be decomposed as a parallel composition of two completely independent subspecies A, B, then it in fact represents a mixture of two different species (two separate molecules). This is key to the interpretation of communication as creation and dissolution of bonds: it means two molecules will bind together as a single molecule when they gain a shared location (as in $S \parallel E \to (\nu\, \ell)(S^*_\ell \mid E^*_\ell)$), and a molecule will break apart when the bonds between its components are broken (as in $(\nu\, \ell)(S^*_\ell \mid E^*_\ell) \to S \parallel E$). In order to see how many distinct species/molecules a species actually contains, we follow Continuous π [234] (and, moreover, abstract algebra) and adopt the notion of *prime species* which cannot be broken down any further.

Definition 3.2.14. A species S is *prime* if, for all species A, B,

$$S \equiv A \mid B \implies A \equiv \mathbf{0} \text{ or } B \equiv \mathbf{0}.$$

Any species S may be decomposed as a unique parallel composition of prime species and we denote by primes(S) the bag of all prime factors of S.

Proposition 3.2.15. *For any species S, we have a unique* prime decomposition. *That is, there is a unique bag of prime species* $\text{primes}(S) = \wr P_1, \ldots, P_n \wr$ *such that,*

$$S \equiv \bigmid_{P \in \text{primes}(S)} P \equiv P_1 \mid \ldots \mid P_n.$$

Proof. This will soon follow from the normal form of Definition 3.2.16/Theorem 3.2.17 in a similar manner to the corresponding result for Continuous π ([234, Theorem 11]). □

Working with structural congruence and prime species requires us to solve two practical problems: how do we test if two species (abstractions, mixtures) are structurally congruent, and how can we concretely represent an equivalence class of species (abstractions, mixtures) up to structural congruence? We answer these problems by defining a *normal form* for species, mixtures, and abstractions which gives a unique representative element to each class of structurally congruent agents. This normal form also decomposes species into compositions of prime species, and so ensures that a mixture is represented as a mixture of prime species, each corresponding to a single type of molecule.

Definition 3.2.16. The normal form for processes, species, and abstractions are defined via the following grammar,

$$\begin{aligned}
\textsc{Mix} &::= \|_{i=1}^{n} \alpha_i \cdot \textsc{Prime} & \textsc{Spec} &::= |_{i=1}^{n} \textsc{Res} \\
\textsc{Res} &::= (\nu\, \ell_1, \ldots, \ell_n)\textsc{Par} & \textsc{Par} &::= |_{i=1}^{n} \textsc{Sum} \\
\textsc{Sum} &::= \textstyle\sum_{i=0}^{n} \pi.\textsc{Abst} & \textsc{Abst} &::= (\ell_1, \ldots, \ell_n)\textsc{Spec},
\end{aligned}$$

where we stipulate the following:

- Prime consists of elements of Spec satisfying the additional condition of primeness.
- The terms in a Proc, Spec, Par, or Sum, and the order of locations in a Res are ordered lexicographically (or according to any other canonical ordering).
- The bound locations in Res or Abst are given standardized names (we can choose canonical names using De Bruijn indices).
- A mixture Mix contains no duplicate species.
- A restriction Res contains no redundant locations, and the last location of Abst is bound in Par.
- There is no partition of locations in a restriction Res which would allow it to be split into a parallel composition of two restrictions.

We can the see that every bond-calculus process may be placed into this unique normal form.

Theorem 3.2.17. *(Unique Normal Forms) Every species (abstraction, mixture) S is structurally congruent to a unique normal form* $\mathrm{nf}(S)$ *of the form described in Definition 3.2.16.*

Proof. (Sketch) This can be shown by constructing an appropriate confluent and terminating rewriting system using directed variants of the structural congruence rules which terminates at the normal form Definition 3.2.16; such rewriting rules have been implemented and extensively tested as part of the bond-calculus implementation. This is similar to existing normal form results for Stochastic π [213, Lemma 3] and Continuous π [234, Theorem 11] (which is expanded in [232, Appendex A]). □

The bond-calculus implementation uses the normal form Definition 3.2.16 as its concrete computational representation for species.

3.2.6 Bond-Calculus Models

We have now defined all of the components of a bond-calculus model,

Definition 3.2.18. A bond-calculus model $(\Pi_0, \mathcal{A})$ consists of

- An initial mixture $\Pi_0 \in \textsc{Mix}$.
- An affinity network $\mathcal{A} \in \textsc{Aff}$.

Here the initial mixtures Π_0 may depend on a number of species definitions, whilst the affinity network may depend on a number of kinetic law definitions. We also embed any isolated mixture as the bond-calculus model $\Pi \equiv (\Pi, \varnothing)$ and any isolated affinity networks as a bond-calculus model $\mathcal{A} \equiv (\mathbf{0}, \mathcal{A})$.

In the spirit of compositional modelling, we are able to build models by combining models of smaller components.

Definition 3.2.19. We define the composition of bond-calculus models $(\Pi, \mathcal{A})$ and $(\Phi, \mathcal{B})$ as the model,

$$(\Pi, \mathcal{A}) \parallel (\Phi, \mathcal{B}) \triangleq (\Pi \parallel \Phi, \mathcal{A} \cup \mathcal{B}).$$

assuming the species definitions and kinetic law definitions of both models are consistent.

The composition of bond-calculus models not only extends the possible reactants by the composition of the mixtures Π and Φ, but allows new affinity patterns to be applied to existing reactants via the composition of the affinity networks $\mathcal{A}$ and $\mathcal{B}$. Thus a bond-calculus model $(\Pi, \mathcal{A})$ may be decomposed as a composition $\Phi \parallel \mathcal{A}$ of its reactants and its affinity network.

$$\frac{}{\sum_{i=0}^{n} s_i@\ell_i.F_i \xrightarrow{s_j}_{\ell_j} F_j}\ \textsc{Choice}_{j,n} \qquad \frac{A \equiv_\alpha B \quad B \xrightarrow{\gamma}_\ell F}{A \xrightarrow{\gamma}_\ell F}\ \textsc{Alpha}$$

$$\frac{A \xrightarrow{\gamma}_\ell F}{A \mid B \xrightarrow{\gamma}_\ell F \mid B}\ \textsc{Par-Left} \qquad \frac{A \xrightarrow{\gamma}_\ell F}{B \mid A \xrightarrow{\gamma}_\ell B \mid F}\ \textsc{Par-Right}$$

$$\frac{A \xrightarrow{\gamma}_\ell F \quad D(\mathbf{x};\mathbf{l}) \triangleq A}{D(\mathbf{y};\mathbf{m}) \xrightarrow[\ell\{\mathbf{m}/\mathbf{l}\}]{\gamma} F\{\mathbf{y}/\mathbf{x},\mathbf{m}/\mathbf{l}\}}\ \textsc{Def} \qquad \frac{A \xrightarrow{\gamma}_\ell F \quad \ell \notin \mathbf{m}}{(\nu\,\mathbf{m})A \xrightarrow{\gamma}_\ell (\nu\,\mathbf{m})F}\ \textsc{Res}$$

$$\frac{A \xrightarrow{\gamma}_\ell F \quad \ell \in \mathbf{m}}{(\nu\,\mathbf{m})A \xrightarrow{\gamma}_\top (\nu\,\mathbf{m})F}\ \textsc{Del} \qquad \frac{A \xrightarrow{\gamma}_\ell F \quad B \xrightarrow{\delta}_\ell G \quad \ell \neq \top}{A \mid B \xrightarrow{\gamma|\delta}_\ell F \mid G}\ \textsc{Com}$$

Figure 3.3: Species multi-transition system rules.

3.3 Semantics

In this section we will define a formal semantics for bond-calculus models. First we will focus on the individuals in each species, giving an operational semantics for the state transitions an individual may undergo upon interaction in the form of a multi-transition system. Then we will turn our attention to continuous mixtures of species, defining the transition matrix $\mathcal{T}(\Pi)$ which captures the "concentration" of each transition $A \xrightarrow{\gamma} F$ within a mixture Π, and see how it may be derived compositionally using the interaction tensor $\odot$. Finally, we will define the reaction rate vector $\mathcal{R}_{\mathcal{A}}$ which determines the rate of each reaction given an affinity network $\mathcal{A}$.

3.3.1 Single Molecule Transition Semantics

We start by defining a semantics for the interactions which may occur in to a single molecule of a chemical species, as a multi-transition systems, with transitions labelled by a cluster of sites, and a shared location. That is, we will represent the ability to interact on site s at location ℓ as a unitary transition,

$$A \xrightarrow{s}_\ell F\ ,$$

whilst internal (allosteric) interactions will build up larger, composite interactions

$$A_1 \mid \ldots \mid A_n \xrightarrow{\gamma}_{\ell} F_1 \mid \ldots \mid F_n$$

labelled with the cluster $\gamma = s_1 | \ldots | s_n$ of all of the sites involved. Whilst operational semantics for non-deterministic programming languages are frequently specified as transition systems in order to capture the fact that the quantitative rate of a chemical reaction depends on the number of copies of a given site which are present in a molecule, we need a multi-transition system, which can include multiple instances of the same reaction (multi-transition systems are also used in other quantitative process algebra semantics [125, 234]). Additionally since our transitions correspond to open reactions, where more interactions may join at any point (either within the same species, or from other reactants in a mixture), transitions go from a species A to an abstraction F representing the partial products of a reaction.

The definition for our multi-transition system is as follows.

Definition 3.3.1 (Multi-transition system)**.** The *multi-transition system* for bond-calculus species consists of a multiset containing the 4-tuples (that is, *transitions*)

$$\left(A \xrightarrow{\gamma}_{\ell} F\right) \triangleq (A, \gamma, \ell, F) \in \textsc{Spec} \times \textsc{Cluster} \times \textsc{Loc} \times \textsc{Abst},$$

which are derivable according to the rules in Figure 3.3 (with multiplicities corresponding to the number of derivations). We will denote the multiset of transitions $A \xrightarrow{\gamma}_{\ell} F$ starting from a particular species A as $\mathrm{trans}(A)$.

This is specified using a small step structural operational semantics [298] according to the transition rules in Figure 3.3. Initial unitary transitions are induced from choices by the Choice rule, whilst the Par-Left and Par-Right rules allow them to propagate unchanged parallel compositions. The Res rule also allows transitions to move through restrictions, provided they are not at a restricted location. The Com rule allows transitions at a shared location ℓ to join together and form a larger interaction. The Del rule provides a way for interactions at a restricted location to move through a restriction by becoming ambient interactions at the location $\top$; this means they may engage in no more internal reactions, but can still be part of reactions with other molecules. Finally, the Dfn rule expands the transitions of named species from their definitions.

Example 3.3.2. The system with species

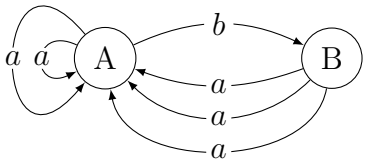
$$A \triangleq a.A + a.A + b.B \qquad\qquad B \triangleq a.A + a.A + a.A$$

has single species transitions,

$$\mathcal{M} = \text{trans}(A) \cup \text{trans}(B) = \left\{\!\!\left\{ 2 \times \left(A \xrightarrow{a}_{\top} A \right),\ 1 \times \left(A \xrightarrow{b}_{\top} B \right),\ 3 \times \left(B \xrightarrow{a}_{\top} A \right) \right\}\!\!\right\},$$

which are illustrated in Figure 3.4. The full transition system also includes multispecies transitions $A|B \xrightarrow{b}_{\top} B|B$, $A|B \xrightarrow{a}_{\top} A|A$, $A|B \xrightarrow{a|a}_{\top} A|A$, etc.

Figure 3.4: The multi-transition system $\mathcal{M}$, visualised as a labelled multi-graph.



3.3.2 Mixture Semantics

We will now move to the second level of the language and give a semantics for mixtures. The semantics we give is inspired by aspects of the vector semantics of Continuous π [234] and the numerical representation of PEPA/Bio-PEPA models [125, 198] and has also been influenced by other numerical representations [200, 312] and fluid approximation techniques [41, 98, 346]. Many of the differences with existing semantics arise from the differences between the bond-calculus' communication mechanism and both binary π-calculus style communication and CSP/PEPA style multi-way cooperation as well as our desire for a compositional continuous semantics.

First we see how we can consider mixtures as a vector space with a basis given by prime species. Given we can decompose any species into primes, we can also decompose any mixture Π into a unique parallel composition of prime species,

$$\Pi \equiv \alpha_1 S_1 \parallel \ldots \parallel \alpha_n S_n \equiv \sum_{S \text{ prime}} [S]_\Pi \, S$$

Then we see that mixtures form a real vector space if we define scalar multiplication of a mixture Π by a scalar $\gamma \in \mathbb{R}$ by

$$\gamma \cdot \Pi = \gamma \sum [S]_\Pi \, S = \sum (\gamma \, [S]_\Pi) \, S$$

and addition of two mixtures Π and Φ by,

$$\Pi + \Phi = \sum [S]_\Pi \, S + \sum [S]_\Phi \, S = \sum ([S]_\Pi + [S]_\Phi) \, S = \Pi \parallel \Phi.$$

Definition 3.3.3. The mixture space $(\mathbb{M}, \cdot, +, \mathbf{0})$ is the real vector space of mixtures with scalar multiplication $\cdot$ and addition $+$ defined above.

If we define the embedding $\langle S \rangle$ of a single species as below, then we see that the vectors $\langle P \rangle$ for prime species P form a basis for $\mathbb{M}$.

Definition 3.3.4. The species embedding is the map $\langle \cdot \rangle : \textsc{Spec} \to \mathbb{M}$ defined by

$$\langle S \rangle = 1 \cdot S = \sum_{P \in \text{primes}\, S} 1 \cdot P.$$

In order to develop our semantics we will need a number of auxiliary vector spaces for patterns, clusters, and transitions allowing us to raise the other elements of our language to the same level as the mixture space.

Definition 3.3.5. We define the following real vector spaces:

- The *pattern space* $\mathbb{P} \triangleq [\textsc{Pattern} \to \mathbb{R}]$ is the space of real vectors indexed by patterns.
- The *cluster space* $\mathbb{G} \triangleq [\textsc{Cluster} \to \mathbb{R}] \subseteq \mathbb{P}$ is the space of real vectors indexed by clusters of sites.
- The *transition space* $\mathbb{T} \triangleq [\textsc{Spec} \times \textsc{Abst} \to \mathbb{R}]$ is the space of real vectors indexed by transitions $A \to F \triangleq (A, F)$.

These are given the bases of indicator functions $\mathbf{I}(\boldsymbol{\gamma})$ (for $\boldsymbol{\gamma} = \gamma_1 \,\|\, \ldots \,\|\, \gamma_n$), $\mathbf{I}(\gamma)$, $\mathbf{I}(A \to F)$ respectively, where the indicator functions $\mathbf{I}(X) : \mathbb{X} \to \mathbb{R}$ are defined by,

$$\mathbf{I}(X)\, Y \triangleq \begin{cases} 1 & \text{if } X = Y \\ 0 & \text{otherwise} \end{cases}$$

The semantics for the language will be defined in terms of linear maps $M \in \mathcal{L}(\mathbb{P}, \mathbb{T})$ (i.e. matrices) which map pattern vectors into transition vectors. These maps correspond to a system of pattern labelled transitions representing the partial reactions arising from a given mixture, with coefficients representing the concentration of the transition. We have a basis for $\mathcal{L}(\mathbb{P}, \mathbb{T})$ consisting of the matrices $\mathbf{E}(A \to F,\, \gamma)$ defined such that

$$\mathbf{E}(A \to F,\, \boldsymbol{\gamma})\, \mathbf{I}(\boldsymbol{\gamma}) \triangleq \begin{cases} \mathbf{I}(A \to F) & \text{if } \boldsymbol{\gamma} = \boldsymbol{\delta} \\ \mathbf{0} & \text{otherwise} \end{cases}$$

for any prime species A and pattern $\boldsymbol{\gamma} = \gamma_1 \| \ldots \| \gamma_n$.

Example 3.3.6. A simple example of a matrix encoding of transition systems is given by multi-transition systems. For example, the multi-transition system $\mathcal{M}$ from Example 3.3.2 may be represented as the matrix,

$$M = 2\mathbf{E}(A \to A,\, a) + \mathbf{E}(A \to B,\, b) + 3\mathbf{E}(B \to A,\, a)\,.$$

If we fix the finite bases, $\mathcal{B} = (A \to A,\, A \to B,\, B \to A)$ and $\mathcal{C} = (a, b)$ for the transition space and site space respectively, then a linear combination of patterns $\varphi = m\mathbf{I}(a) + n\mathbf{I}(b)$ and a linear combination of transitions $\psi = p\mathbf{I}(A \to A) + q\mathbf{I}(A \to B) + r\mathbf{I}(B \to A)$ could be written as column vectors, whilst M could be written as a matrix:

$$\varphi = \begin{bmatrix} m \\ n \end{bmatrix} \qquad \psi = \begin{bmatrix} p \\ q \\ r \end{bmatrix} \qquad M = 2\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}\begin{bmatrix} 1 & 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}\begin{bmatrix} 0 & 1 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}\begin{bmatrix} 1 & 0 \end{bmatrix} = \begin{bmatrix} 2 & 0 \\ 0 & 1 \\ 3 & 0 \end{bmatrix}.$$

Then we can find the total transition after a reaction matching the pattern $\varphi = m \cdot a \parallel n \cdot b$ as,

$$M\varphi = \begin{bmatrix} 2 & 0 \\ 0 & 1 \\ 3 & 0 \end{bmatrix}\begin{bmatrix} m \\ n \end{bmatrix} = m\begin{bmatrix} 2 \\ 0 \\ 3 \end{bmatrix} + n\begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} = m(2\mathbf{I}(A \to A) + 3\mathbf{I}(B \to A)) + n\mathbf{I}(A \to B)\,.$$

To extend this matrix encoding to quantitative transitions between mixtures we will use real-valued concentrations as coefficients, rather than multiplicities.

We will now give a compositional semantics for mixtures in terms of the transition matrix $\mathcal{T}(\Pi)$.

Definition 3.3.7. The *transition matrix* $\mathcal{T}(\Pi) \in \mathcal{L}(\mathbb{P}, \mathbb{T})$ is defined for any mixture Π by

$$\mathcal{T}(\alpha A) \triangleq \alpha \sum \Big(\mathbf{E}(S \to F,\, \gamma) : S \in \text{primes}(A),\, S \xrightarrow[\top]{\gamma} F \Big)$$
$$\mathcal{T}(\Pi \parallel \Phi) \triangleq \mathcal{T}(\Pi) + \mathcal{T}(\Phi)$$

First we note that $\mathcal{T} : \mathbb{M} \to \mathcal{L}(\mathbb{P}, \mathbb{T})$ is a linear map, with entries representing the concentration of the transitions at a given cluster. The transition matrix only contains the transitions which are possible from a prime species and which are labelled by clusters, whilst it omits composite transitions from general species which are labelled by general patterns. However, we will build up all of the possible n-way interactions between the species of Π step by step using the bilinear map $\odot$ defined as follows.

Definition 3.3.8. The *interaction tensor* is the bilinear map $\odot : \mathcal{L}(\mathbb{P}, \mathbb{T}) \times \mathcal{L}(\mathbb{P}, \mathbb{T}) \to \mathcal{L}(\mathbb{P}, \mathbb{T})$ defined by,

$$\mathbf{E}(A \to F, \boldsymbol{\gamma}) \odot \mathbf{E}(B \to G, \boldsymbol{\delta}) \triangleq \mathbf{E}(A|B \to F|G, \boldsymbol{\gamma}\|\boldsymbol{\delta})$$

Then the interaction tensor composes transitions (representing open multi-way interactions) $A \xrightarrow{\gamma} F$ and $B \xrightarrow{\delta} G$ into larger open multi-way interactions, $A|B \xrightarrow{\gamma\|\delta} F|G$ which can be composed again as more agents join. The bilinearity of this map means that the concentration of composite transition is assumed to be the product of the concentrations of the reactants – in the mass action case this is proportional to the reaction rate, and even under general kinetic laws, this measures the availability of the reactants (in the space of possible combinations of reactants), although this may no longer correspond directly to the reaction rate. We note that finite dimensional matrices form a commutative algebra under $\odot$ with identity $\mathbf{1} = \mathbf{E}(\mathbf{0} \to \mathbf{0}, \varnothing)$, which can be completed to form an algebra of infinite dimensional matrices.

Now we see that the full transition system, containing all possible n-way interactions between species in Π can be generated using the exponential function (with respect to $\odot$) as follows

$$\exp_{\odot}(\mathcal{T}(\Pi)) \triangleq \sum_{n=0}^{\infty} \frac{1}{n!} \mathcal{T}(\Pi)^{\odot n} \tag{3.1}$$

$$= \mathbf{1} + \mathcal{T}(\Pi) + \frac{1}{2}\mathcal{T}(\Pi) \odot \mathcal{T}(\Pi) + \frac{1}{6}\mathcal{T}(\Pi) \odot \mathcal{T}(\Pi) \odot \mathcal{T}(\Pi) + \ldots \tag{3.2}$$

That is, the transition matrix $\mathcal{T}(\Pi)$ exponentially generates the transitions of Π (under product $\odot$).

We now see that the transition matrix gives us a practical way of finding the reactions of a system, since $\exp_{\odot}(\mathcal{T}(\Pi))\mathbf{I}(\boldsymbol{\gamma}) = \sum_{n=0}^{\infty} \frac{1}{n!} \mathcal{T}(\Pi)^{\odot n} \mathbf{I}(\boldsymbol{\gamma}) = \sum_j \alpha_j \mathbf{I}(A_j \to F_j)$ gives the vector of concentrations of all transitions matching pattern $\boldsymbol{\gamma}$ and moreover, this sum is finite since $\mathcal{T}(\Pi)^{\odot n}\mathbf{I}(\boldsymbol{\gamma})$ is zero for $n > |\boldsymbol{\gamma}|$. Furthermore, the following result assures us composition of transition matrices is equivalent to composition of transition systems.

Proposition 3.3.9. *For any mixtures* Π *and* Φ *we have that*

$$\exp_{\odot}(\mathcal{T}(\Pi \parallel \Phi)) = \exp_{\odot}(\mathcal{T}(\Pi)) \odot \exp_{\odot}(\mathcal{T}(\Phi)).$$

Proof. This follows by the linearity of $\mathcal{T}$ and the fact that $\exp_{\odot}$ is multiplicative. □

Remark 3.3.10. An efficient implementation of the semantics should only compute the coefficients of the terms of $\exp_{\odot}(\mathcal{T}(\Pi))$ which actually correspond to patterns in the

affinity network $\mathcal{A}$. This can be achieved either by computing $\exp_\odot(\mathcal{T}(\Pi))$ lazily, or by setting $\mathbf{E}(A \to F, \boldsymbol{\gamma}) = \mathbf{0}$ unless $\boldsymbol{\gamma} \subseteq \boldsymbol{\delta}$ for some $\boldsymbol{\delta} \mathbin{@} f \in \mathcal{A}$.

Remark 3.3.11. The transition expansion equation Eq. (3.1) should be compared with the equation [232, Theorem 3.3.15] in the semantics of the Continuous π-calculus, which derives the binary self-interactions for a singleton mixture $c \cdot A$ via an *interaction tensor* $\oplus_M$ in the term $\frac{1}{2}(\partial(c \cdot A) \oplus_M \partial(c \cdot A))$. Our semantics can be seen as an extension of this approach to arbitrary self-interactions with the exponential coefficients of Eq. (3.1) correctly generalizing the combinatorics. This differs from the original semantics of the bond-calculus in [358] which does not take the combinatorics of self-interacting sites into account.

Example 3.3.12. Suppose we have a single species A which can engage in reactions with itself,

$$A \triangleq a.A + b.B.$$

according to the affinity network,

$$\mathcal{A} = \Big\{ a \parallel b \mathbin{@} \mathrm{L},\ b \parallel b \mathbin{@} \mathrm{M} \Big\}.$$

This system has transition matrix,

$$\mathcal{T}([A]\,A) = [A]\,\mathbf{E}(A \to A,\ a) + [A]\,\mathbf{E}(A \to A,\ a)$$

From which we see,

$$\begin{aligned}
\exp_\odot(\mathcal{T}([A]\,A))\mathbf{I}(a \parallel b) &= \mathbf{1}\mathbf{I}(a \parallel b) + \mathcal{T}([A]\,A)\mathbf{I}(a \parallel b) + \frac{1}{2}(\mathcal{T}([A]\,A) \odot \mathcal{T}([A]\,A))\mathbf{I}(a \parallel b) \\
&\qquad + \underbrace{\frac{1}{6}(\mathcal{T}([A]\,A) \odot \mathcal{T}([A]\,A) \odot \mathcal{T}([A]\,A))\mathbf{I}(a \parallel b) + \dots}_{=\mathbf{0}} \\
&= \frac{1}{2}\,[A]^2\,(\mathbf{E}(A|A \to A|B,\ a \parallel b) + \mathbf{E}(A|A \to B|A,\ b \parallel a))\mathbf{I}(a \parallel b) \\
&= [A]^2\,\mathbf{I}(A|A \to A|B)\,.
\end{aligned}$$

whilst

$$\begin{aligned}
\exp_\odot(\mathcal{T}([A]\,A))\mathbf{I}(a \parallel a) &= \frac{1}{2}\,[A]^2\,\mathbf{E}(A|A \to A|A,\ a \parallel a)\,\mathbf{I}(a \parallel b) \\
&= \frac{1}{2}\,[A]^2\,\mathbf{I}(A|A \to A|B)\,.
\end{aligned}$$

This shows that the concentration of the homogeneous self-interaction of the site a with itself is halved, whilst the concentration of interaction between sites a and b is not, in accordance with the combinatorics of the law of mass action.

Example 3.3.13. In the HBr formation process we have species,

$$\mathrm{H}_2 \triangleq h(\ell, m).\Big(\mathrm{H}^{(\ell)} \mid \mathrm{H}^{(m)}\Big) \qquad \mathrm{H}^{(\ell)} \triangleq h^*@\ell.\mathrm{H}^{(\ell)}$$
$$\mathrm{Br}_2 \triangleq b(\ell, m).\Big(\mathrm{Br}^{(\ell)} \mid \mathrm{Br}^{(m)}\Big) \qquad \mathrm{Br}^{(\ell)} \triangleq b^*@\ell.\mathrm{Br}^{(\ell)}$$

and the dynamic complex,

$$\mathrm{HBr} = (\nu\, \ell)\Big(\mathrm{H}^{(\ell)} \mid \mathrm{B}^{(\ell)}\Big)\,.$$

Then a general mixture may be written as

$$\Pi \triangleq [\mathrm{H}_2]\,\mathrm{H}_2 \parallel [\mathrm{Br}_2]\,\mathrm{Br}_2 \parallel [\mathrm{HBr}]\,\mathrm{HBr}$$

We see that interaction matrix of the system is

$$\begin{aligned}\mathcal{T}(\Pi) = [\mathrm{H}_2]\,\mathbf{E}\Big(\mathrm{H}_2 \to (\ell, m)\Big(\mathrm{H}^{(\ell)}|\mathrm{H}^{(m)}\Big),\, h\Big) + [\mathrm{Br}_2]\,\mathbf{E}\Big(\mathrm{Br}_2 \to (\ell, m)\Big(\mathrm{Br}^{(\ell)}|\mathrm{Br}^{(m)}\Big),\, b\Big)\\ + [\mathrm{HBr}_2]\,\mathbf{E}(\mathrm{HBr} \to \mathrm{HBr},\, h^*|b^*)\end{aligned}$$

and the interactions on pattern $h \parallel b \parallel h^*|b^*$ are

$$\begin{aligned}\exp_{\odot}(\mathcal{T}(\Pi))\mathbf{I}(h \parallel b \parallel h^*|b^*) = [\mathrm{H}_2][\mathrm{Br}_2][\mathrm{HBr}]\,\mathbf{I}\Big(\mathrm{H}_2|\mathrm{Br}_2|\mathrm{HBr}\\ \to (\ell, m)\Big(\mathrm{H}^{(\ell)}|\mathrm{H}^{(m)}|\mathrm{Br}^{(\ell)}|\mathrm{Br}^{(m)}|\mathrm{HBr}\Big)\Big).\end{aligned}$$

3.3.3 Rate Semantics for Affinity Networks

We will now define the rate vector $\mathcal{R}_{\mathcal{A}}([\gamma_1]\,\mathbf{I}(\gamma_1) + \ldots + [\gamma_n]\,\mathbf{I}(\gamma_n)) \in \mathbb{P}$. This gives the stoichiometric rate (that is, rate per unit reactant concentration) of the reactions at each pattern, given a vector representing the concentration of each site cluster $\gamma \in \textsc{Cluster}$ in the system.

Definition 3.3.14. The *rate vector* $\mathcal{R}_{\mathcal{A}} : \mathbb{G} \to \mathbb{P}$ for affinity network $\mathcal{A}$, is the (nonlinear) function defined by

$$\mathcal{R}_{\mathcal{A}}\left(\sum_{\gamma \in \textsc{Cluster}} [\gamma]\,\mathbf{I}(\gamma)\right) = \sum_{(\gamma_1 \| \ldots \| \gamma_m @ f) \in \mathcal{A}} \left(\frac{f([\gamma_1], \ldots, [\gamma_m])}{[\gamma_1] \ldots [\gamma_m]}\right)\mathbf{I}(\gamma_1 \| \ldots \| \gamma_m)\,.$$

This vector gives a meaningful semantics for affinity networks, which allows the reaction rates at each pattern to be computed from the site concentrations.

Example 3.3.15. In a system of mass action reactions with affinity network,

$$\mathcal{A} = \Big\{\, \gamma^{(i)} = \gamma_1^{(i)} \parallel \ldots \parallel \gamma_{n_i}^{(i)} \;@\; \mathrm{MA}_{r_i} \;\Big|\; i \in I \,\Big\}$$

we have reaction vector,

$$\mathcal{R}_{\mathcal{A}}\Big(\sum [\gamma]\,\mathbf{I}(\gamma)\Big) = \sum_{i\in I} \frac{\mathrm{MA}_{r_i}\Big(\gamma_1^{(i)},\ldots,\gamma_{n_i}^{(i)}\Big)}{\Big[\gamma_1^{(i)}\Big]\cdots\Big[\gamma_{n_i}^{(i)}\Big]}\mathbf{I}\Big(\gamma^{(i)}\Big) = \sum_{i\in I} r_i\mathbf{I}\Big(\gamma^{(i)}\Big)\,.$$

That is, the reaction rate vector is the vector of stoichiometric rate constants for each reaction.

Example 3.3.16. In the HBr formation process, we had affinity network,

$$\mathcal{A} = \{h \parallel b \parallel h^*|b^* \mathbin{@} \mathrm{R}_k\} \qquad \text{where } \mathrm{R}_k(x_1,x_2,x_3) \triangleq \frac{x_1 x_2^{1/2}}{1+k\frac{x_3}{x_2}}$$

and so the reaction rate vector is equal to,

$$\begin{aligned}\mathcal{R}_{\mathcal{A}}\left(\sum_{\gamma\in\textsc{Cluster}} [\gamma]\,\mathbf{I}(\gamma)\right) &= \frac{\mathrm{R}_k([h]\,,[b]\,,[h^*|b^*])}{[h][b][h^*|b^*]}\ \mathbf{I}(h\|b\|h^*|b^*)\\ &= \frac{[b]^{1/2}}{[h^*|b^*]\,([b]+k\,[h^*|b^*])}\mathbf{I}(h\|b\|h^*|b^*)\,.\end{aligned}$$

Since the rates of a reaction depend on the concentrations of the reaction sites involved, this function takes as an argument a vector of the total concentrations of each site pattern in the system – these concentrations record the total concentrations of all species within a mixture which can interact at a given site, and correspond roughly to the notion of the *apparent rate* of an action in a PEPA process [41, 346]. We can define each of these concentrations inductively as,

Definition 3.3.17. The *concentration of cluster* $\gamma \in$ *Cluster in mixture* Π is the real number $[\gamma]_\Pi$ defined by

$$[\gamma]_{[A]\cdot A} \triangleq [A]\sum_F \mathrm{card}\Big(A \xrightarrow{\gamma}_\top F, \mathrm{trans}(A)\Big) \qquad [\gamma]_{\Pi\|\Phi} \triangleq [\gamma]_\Pi + [\gamma]_\Phi\,.$$

We then define the *site concentration vector of* Π by,

$$\mathcal{C}(\Pi) \triangleq \sum_{\gamma\in\textsc{Cluster}} [\gamma]_\Pi\,\mathbf{I}(\gamma)\,.$$

The site concentration vector $\mathcal{C}(\Pi)$ can be computed compositionally based on the structure of Π since $\mathcal{C}(\Pi \parallel \Phi) = \mathcal{C}(\Pi) + \mathcal{C}(\Phi)$. However, the following proposition shows us how $\mathcal{T}(\Pi)$ already captures the site concentrations in Π.

Proposition 3.3.18. *For any cluster* $\gamma \in$ *Cluster and process* Π *we have that,*

$$[\gamma]_\Pi = \|\mathcal{T}(\Pi)\mathbf{I}(\gamma)\|_1$$

where $\|\cdot\|_1$ *is the* ℓ^1*-norm* $\Big\|\sum_j \alpha_j\mathbf{I}(A_j \to F_j)\Big\|_1 = \sum_j |\alpha_j|$.

3.3.4 Compositionality of Semantics

Between the transition matrix and the rate vector we have a semantics for bond-calculus models which is compositional in both the mixture and the affinity network.

Definition 3.3.19. We define the *denotation map* $[\![\cdot]\!] : \mathit{MIX} \times \mathit{AFF} \to \mathcal{L}(\mathbb{S}, \mathbb{T}) \times (\mathbb{G} \to \mathbb{R})$ by

$$[\![(\Pi, \mathcal{A})]\!] = (\mathcal{T}(\Pi), \mathcal{R}_{\mathcal{A}}).$$

for any mixture Π and affinity network $\mathcal{A}$.

Theorem 3.3.20 (Compositionality of semantics)**.** *For mixtures* Π *and* Φ *and affinity networks* $\mathcal{A}$*,* $\mathcal{B}$*, we have that,*

$$[\![(\Pi, \mathcal{A}) \parallel (\Phi, \mathcal{B})]\!] = (\mathcal{T}(\Pi) + \mathcal{T}(\Phi),\ \mathcal{R}_{\mathcal{A}} + \mathcal{R}_{\mathcal{B}})\,.$$

We end this section with an example of how we can compositionally compute the semantics when a new species is be added to an existing model.

Example 3.3.21. Suppose we introduce a new reactant into the HBr formation process, a radioactive isotope of Dibromide, $^{77}Br_2$ [274]. This will have identical chemical properties as Br_2 and hence can be modelled using the same reaction sites, however, the two can be distinguished since ^{77}Br emits gamma rays during its radioactive decay. We can model $^{77}Br_2$ via the new species,

$$^{77}\mathrm{Br}_2 \triangleq b(\ell, m).\Big(^{77}\mathrm{Br}^{(\ell)} \mid {}^{77}\mathrm{Br}^{(m)}\Big) \qquad\qquad ^{77}\mathrm{Br}^{(\ell)} \triangleq b^*@\ell.{}^{77}\mathrm{Br}^{(\ell)}$$

(for simplicity we will not consider the effects of ^{77}Br decay or any new reactions emerging from new $^{77}Br{-}Br$ and $^{77}Br{-}HBr$ complexes). Then, defining the extended system as $\Phi \triangleq \Pi \parallel \left[^{77}\mathrm{Br}_2\right] {}^{77}\mathrm{Br}_2$, we see the overall transition matrix is,

$$\begin{aligned}\mathcal{T}(\Phi) &= \mathcal{T}(\Pi) + \mathcal{T}\Big(\left[^{77}\mathrm{Br}_2\right] {}^{77}\mathrm{Br}_2\Big) \\ &= \mathcal{T}(\Pi) + \left[^{77}\mathrm{Br}_2\right]\mathbf{E}\Big(\mathrm{Br}_2 \to (\ell, m)\Big(^{77}\mathrm{Br}^{(\ell)}|{}^{77}\mathrm{Br}^{(m)}\Big),\ b\Big)\end{aligned}$$

The interactions at pattern $h \parallel b \parallel h^*|b^*$ in the extended system are now,

$$\begin{aligned}\exp_{\odot}(\mathcal{T}(\Phi))\mathbf{I}(h \parallel b \parallel h^*|b^*) = {}& [\mathrm{H}_2][\mathrm{Br}_2][\mathrm{HBr}]\ \ \mathbf{I}\Big(\mathrm{H}_2|\mathrm{Br}_2|\mathrm{HBr} \to (\ell, m)\Big(\mathrm{H}^{(\ell)}|\mathrm{H}^{(m)}|\mathrm{Br}^{(\ell)}\Big)\Big) \\ &+ [\mathrm{H}_2]\Big[^{77}\mathrm{Br}_2\Big][\mathrm{HBr}]\mathbf{I}\Big(\mathrm{H}_2|{}^{77}\mathrm{Br}_2|\mathrm{HBr} \to (\ell, m)\Big(\mathrm{H}^{(\ell)}|\mathrm{H}^{(m)}|{}^{77}\mathrm{Br}^{(\ell)}\Big)\Big)\,,\end{aligned}$$

and include a new cross reaction involving the populations of H_2 and HBr in Π and the population of ^{77}Br.

The reaction rate vector $\mathcal{R}_{\mathcal{A}}$ is unchanged by the addition of the new species, however, in $\mathcal{C}(\Phi)$ the concentration of site b now also includes the concentration of $^{77}\mathrm{Br}_2$ isotope so,

$$\mathcal{R}_{\mathcal{A}}(\mathcal{C}(\Phi)) = \frac{\Big([\mathrm{Br}_2] + \Big[^{77}\mathrm{Br}_2\Big]\Big)^{1/2}}{[\mathrm{HBr}]\Big([\mathrm{Br}_2] + \Big[^{77}\mathrm{Br}_2\Big] + k\,[\mathrm{HBr}]\Big)}\mathbf{I}(h \parallel b \parallel h^*|b^*)\,.$$

3.4 Dynamics

We will now show how the semantics defined in the previous section may be used to simulate and analyse the dynamics of a system. Firstly we will see how the combination of the transition matrix $\mathcal{T}$ and the rate vector $\mathcal{R}_{\mathcal{A}}$ directly define the dynamics of the system when interpreted as a Chemical Reaction Network. Next we will see how to compositionally derive a more compact representation of the dynamics, in the form of the difference matrix $\mathcal{D}(\Pi) \in \mathcal{L}(\mathbb{P}, \mathbb{M})$ which gives the net change in concentration for each prime species at each pattern, and can be used directly to define the dynamics of the system as a vector field,

$$\frac{\mathrm{d}\Pi}{\mathrm{d}t} \triangleq \mathcal{D}(\Pi)\mathcal{R}_{\mathcal{A}}(\mathcal{C}(\Pi)),$$

over mixtures, specifying the instantaneous evolution of the system starting at any process vector Π (under a given affinity network $\mathcal{A}$). Finally we will see how to symbolically derive a system of coupled differential equations which capture the dynamics of the system starting from a given initial mixture Π_0.

3.4.1 Chemical Reaction Network Extraction

In general a bond-calculus mixture may evolve to produce infinitely many different distinct prime species – such infinite systems have been studied in the Stochastic π calculus and correspond to polymerization reactions [106]. However, given a finite initial mixture Π_0, many models will only generate finitely many prime species, allowing us to interpret the model as a Chemical Reaction Network involving finitely many species. This is possible by considering the evolution of a single generic symbolic mixture of prime species $\Pi \triangleq \alpha_1 A_1 \parallel \ldots \parallel \alpha_n A_n$ (which we assume contains all species reachable from the initial mixture Π_0 as in [20, Section 3.2.4]). To do this we take the product,

$$\exp_{\odot}(\mathcal{T}(\Pi))\mathcal{R}_{\mathcal{A}}(\mathcal{C}(\Pi)) = \sum_j r_j(\boldsymbol{\alpha})\mathbf{I}(A_j \to F_j)$$

which gives the rates $r_j(\boldsymbol{\alpha})$ of each transition $A_j \to F_j$ in the system (which depend on the vector of concentrations $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n)$ of each species). Then we can interpret this

system as a Chemical Reaction Network [186] with reactions,

$$A_{j,1} \quad + \quad \ldots \quad + \quad A_{j,p_j} \quad \xrightarrow{r_j(\boldsymbol{\alpha})} \quad B_{j,1} \quad + \quad \ldots \quad + \quad B_{j,q_j}$$

where $\text{primes}(A_j) = \wr A_{j,1}, \ldots, A_{j,p_j} \int$ and $\text{primes}(\text{commit}(F_j)) = \wr B_{j,1}, \ldots, B_{j,q_j} \int$. This Chemical Reaction Network can then either be used to directly extract a system of ODEs, or interpreted stochastically as a PCTMC (Population Continuous Time Markov Chain). The bond-calculus tool (Section 9.1) supports either ODE extraction and numerical simulation via SciPy [218] or stochastic simulation via StochPy [250].

3.4.2 Difference Matrix and Vector Field

We can also define the dynamics of the model directly, as a vector field which assigns an evolution vector $\frac{\mathrm{d}\Pi}{\mathrm{d}t} = \frac{\mathrm{d}(\Pi,\mathcal{A})}{\mathrm{d}t}$ to each mixture Π of species which determines the instantaneous evolution of the system from the given mixture. To this end we define the difference matrix $\mathcal{D}(\Pi)$, which represents the impact of a transition on the overall system as a difference vector $\mathcal{D}(\Pi)\mathbf{I}(\boldsymbol{\gamma}) = \sum_i \alpha_i \mathbf{I}(S_i) \in \mathbb{P}$, which captures the net change in concentration in each species resulting from the transition (assuming unit stoichiometric rate).

Definition 3.4.1. The *difference matrix* $\mathcal{D}(\Pi) \in \mathcal{L}(\mathbb{P}, \mathbb{M})$ of a mixture Π is defined by,

$$\mathcal{D}(\Pi) = \mathcal{F} \exp_{\odot}(\mathcal{T}(\Pi))$$

where the *finalisation map* $\mathcal{F} \in \mathcal{L}(\mathbb{T}, \mathbb{M})$ is the linear map defined by

$$\mathcal{F}\mathbf{I}(A \to F) = \langle \text{commit}(F) \rangle - \langle A \rangle \,.$$

Example 3.4.2. For the HBr formation process, we can use the transition matrix $\mathcal{T}(\Pi)$ Example 3.3.13 to calculate the difference vector at the pattern $h \parallel b \parallel h^*|b^*$,

$$\begin{aligned}\mathcal{D}(\Pi)\mathbf{I}(h \parallel b \parallel h^*|b^*) &= \mathcal{F}\,[\mathrm{H_2}][\mathrm{Br_2}][\mathrm{HBr}]\,\mathbf{I}\Big(\mathrm{H_2}|\mathrm{Br_2}|\mathrm{HBr} \to (\ell, m)\Big(\mathrm{H}^{(\ell)}|\mathrm{H}^{(m)}|\mathrm{Br}^{(\ell)}|\mathrm{Br}^{(m)}|\mathrm{HBr}\Big)\Big) \\ &= [\mathrm{H_2}][\mathrm{Br_2}][\mathrm{HBr}]\,(2\,\mathrm{HBr} - \mathrm{H_2} - \mathrm{Br_2})\end{aligned}$$

since,

$$\begin{aligned}\Big\langle \text{commit}\Big((\ell, m)\Big(\mathrm{H}^{(\ell)}|\mathrm{H}^{(m)}|\mathrm{Br}^{(\ell)}|\mathrm{Br}^{(m)}|\mathrm{HBr}\Big)\Big)\Big\rangle &= \Big\langle (\nu\,\ell, m)\Big(\mathrm{H}^{(\ell)}|\mathrm{H}^{(m)}|\mathrm{Br}^{(\ell)}|\mathrm{Br}^{(m)}|\mathrm{HBr}\Big)\Big\rangle \\ &= \langle \mathrm{HBr} \mid \mathrm{HBr} \mid \mathrm{HBr} \rangle \\ &= 3\,\mathrm{HBr}\end{aligned}$$

and

$$\langle \mathrm{H_2} \mid \mathrm{Br_2} \mid \mathrm{HBr} \rangle = \mathrm{H_2} + \mathrm{Br_2} + \mathrm{HBr}.$$

We are also able to exploit the compositionality of $\mathcal{T}$ and the linearity of $\mathcal{F}$ to compute $\mathcal{D}$ compositionally (in conjunction with $\exp_\odot(\mathcal{T}(\Pi))$),

Proposition 3.4.3 (Compositionality of $\mathcal{D}$)**.** *For any pair of mixtures* Π, Φ *we have that,*

$$\mathcal{D}(\Pi \parallel \Phi) = \exp_\odot(\mathcal{T}(\Pi)) \oplus \exp_\odot(\mathcal{T}(\Phi)) \tag{3.3}$$

$$= \mathcal{D}(\Pi) + \mathcal{D}(\Phi) + (\exp_\odot(\mathcal{T}(\Pi)) - \mathbf{1}) \oplus (\exp_\odot(\mathcal{T}(\Phi)) - \mathbf{1}) \tag{3.4}$$

where the reaction tensor $\oplus : \mathcal{L}(\mathbb{S}, \mathbb{T}) \times \mathcal{L}(\mathbb{S}, \mathbb{T}) \to \mathcal{L}(\mathbb{S}, \mathbb{M})$ *is the bilinear map defined by,*

$$\mathbf{E}(A \to F, \boldsymbol{\gamma}) \oplus \mathbf{E}(B \to G, \boldsymbol{\delta}) \triangleq \mathbf{E}(\mathrm{commit}(F|G), \boldsymbol{\gamma}\|\boldsymbol{\delta}) - \mathbf{E}(A, \boldsymbol{\gamma}\|\delta) - \mathbf{E}(B, \boldsymbol{\gamma}\|\boldsymbol{\delta})$$

Proof. Firstly we see that the definition of $\oplus$ gives $\mathbf{X} \oplus \mathbf{Y} = \mathcal{F}(\mathbf{X} \odot \mathbf{Y})$ and hence,

$$\begin{aligned}
\mathcal{D}(\Pi \parallel \Phi) &= \mathcal{F}(\exp_\odot(\mathcal{T}(\Pi \parallel \Phi))) \\
&= \mathcal{F}(\exp_\odot(\mathcal{T}(\Pi)) \odot \exp_\odot(\mathcal{T}(\Phi))) \\
&= \exp_\odot(\mathcal{T}(\Pi)) \oplus \exp_\odot(\mathcal{T}(\Phi))
\end{aligned}$$

establishing Equation (3.3). Thus we have that

$$\begin{aligned}
&(\exp_\odot(\mathcal{T}(\Pi)) - \mathbf{1}) \oplus (\exp_\odot \mathcal{T}(\Phi) - \mathbf{1}) \\
&= \exp_\odot(\mathcal{T}(\Pi)) \oplus \exp_\odot(\mathcal{T}(\Phi)) - \mathbf{1} \oplus \exp_\odot(\mathcal{T}(\Phi)) - \exp_\odot(\mathcal{T}(\Pi)) \oplus \mathbf{1} - \mathbf{1} \oplus \mathbf{1} \\
&= \mathcal{F}(\exp_\odot(\mathcal{T}(\Pi)) \odot \exp_\odot(\mathcal{T}(\Phi))) - \mathcal{F} \exp_\odot(\mathcal{T}(\Phi)) - \mathcal{F} \exp_\odot(\mathcal{T}(\Pi)) - \mathbf{0} \\
&= \mathcal{D}(\Pi \parallel \Phi) - \mathcal{D}(\Pi) - \mathcal{D}(\Phi)
\end{aligned}$$

establishing Equation (3.4). □

Remark 3.4.4. Similarly to the case for $\mathcal{T}(\Pi)$, an efficient implementation of the semantics may avoid computing coefficients of $\mathcal{D}(\Pi)$ which do not occur in the affinity network $\mathcal{A}$ by setting $\mathbf{E}(A \to F, \boldsymbol{\gamma}) \oplus \mathbf{E}(B \to G, \boldsymbol{\delta}) = \mathbf{0}$ whenever $\boldsymbol{\gamma} \parallel \boldsymbol{\delta} \mathbin{@} f \notin \mathcal{A}$ for some f.

Example 3.4.5. In the extended HBr example, we can compute the reaction matrix based on the reaction and transition matrices of the original system and restricted to

reactions in the affinity network given in Example 3.2.7 following Remark 3.4.4,

$$\begin{aligned}\mathcal{D}\Big(\Pi \parallel \Big[{}^{77}\mathrm{Br}_2\Big]\,{}^{77}\mathrm{Br}_2\Big) = \quad & \mathcal{D}(\Pi) + \mathcal{D}\Big(\Big[{}^{77}\mathrm{Br}_2\Big]\,{}^{77}\mathrm{Br}_2\Big) \\ & + \Big(\exp_{\odot}(\mathcal{T}(\Pi)) - \mathbf{1}\Big) \oplcirc \Big(\exp_{\odot}\Big(\mathcal{T}\Big(\Big[{}^{77}\mathrm{Br}_2\Big]\,{}^{77}\mathrm{Br}_2\Big)\Big) - \mathbf{1}\Big) \\ = [\mathrm{H}_2][\mathrm{HBr}]\Big(& \quad [\mathrm{Br}_2]\Big(\quad 2\mathbf{E}(\mathrm{HBr},\, h\|b\|h^*|b^*) \\ & \qquad\qquad - \mathbf{E}(\mathrm{H}_2,\, h\|b\|h^*|b^*) \\ & \qquad\qquad - \mathbf{E}(\mathrm{Br}_2,\, h\|b\|h^*|b^*)\Big) \\ & + \Big[{}^{77}\mathrm{Br}_2\Big]\Big(\quad 2\mathbf{E}\Big(\mathrm{H}^{77}\mathrm{Br},\, h\|b\|h^*|b^*\Big) \\ & \qquad\qquad - \mathbf{E}(\mathrm{H}_2,\, h\|b\|h^*|b^*) \\ & \qquad\qquad - \mathbf{E}\Big({}^{77}\mathrm{Br}_2,\, h\|b\|h^*|b^*\Big)\Big)\Big),\end{aligned}$$

where

$$\mathrm{H}^{77}\mathrm{Br} = (\nu\,\ell)\Big(\mathrm{H}^{(\ell)} \mid {}^{77}\mathrm{Br}^{(\ell)}\Big)$$

is a newly created dynamic complex.

Finally, we may define the vector field representing the dynamics of the system.

Definition 3.4.6. The *dynamics of a model* $(\Pi, \mathcal{A})$ are defined as the vector field,

$$\frac{\mathrm{d}\Pi}{\mathrm{d}t} = \frac{\mathrm{d}(\Pi, \mathcal{A})}{\mathrm{d}t} \triangleq \mathcal{D}(\Pi)\mathcal{R}_{\mathcal{A}}(\mathcal{C}(\Pi)).$$

based on the *evolution vector* $\frac{\mathrm{d}(\Pi,\mathcal{A})}{\mathrm{d}t}$ of the model $(\Pi, \mathcal{A})$.

This has a similar form to the dynamical equation of a Chemical Reaction Network [11, Equation 7] or to the ODE extraction equation for Bio-PEPA [125, Section 8.3], however, a key difference is that the rate laws $\mathcal{R}_{\mathcal{A}}$ are given as stoichiometric rates and depend on cluster (i.e. reaction site) concentrations rather than species concentrations whilst the matrix $\mathcal{D}(\Pi)$ contains not only the stoichiometric constants but the concentrations of each party involved in a reaction. This is what allows $\mathcal{D}$ to be computed compositionally using the multilinear functions $\odot$ and $\oplcirc$ (since now all of the essential nonlinearity of the system is contained in $\mathcal{R}_{\mathcal{A}}$) and is what makes it possible for a single role in a reaction (at a given cluster) to be shared between multiple species as in Example 3.4.8 (this corresponds to the so called *Stirling amendment* to the PEPA fluid approximation [41]).

Example 3.4.7. In the HBr formation process we obtain the vector field,

$$\begin{aligned}\frac{\mathrm{d}\Pi}{\mathrm{d}t} &= \frac{\mathrm{d}(\Pi,\mathcal{A})}{\mathrm{d}t} = \mathcal{D}(\Pi)\mathcal{R}_{\mathcal{A}}(\mathcal{C}(\Pi)) \\ &= \frac{[b]_{\Pi}^{1/2}}{[h^*|b^*]_{\Pi}\left([b]_{\Pi} + k[h^*|b^*]_{\Pi}\right)}\,[\mathrm{HBr}][\mathrm{H_2}][\mathrm{Br_2}]\,(2\,\mathrm{HBr} - \mathrm{H_2} - \mathrm{Br_2}) \\ &= \frac{[\mathrm{H_2}]\,[\mathrm{Br_2}]^{1/2}}{1 + k\frac{[\mathrm{HBr}]}{[\mathrm{Br_2}]}}\,(2\,\mathrm{HBr} - \mathrm{H_2} - \mathrm{Br_2})\,.\end{aligned}$$

Example 3.4.8. When we extend the system with the radioactive isotope $^{77}\mathrm{Br_2}$ as in Example 3.3.21, we get the overall evolution vector,

$$\begin{aligned}\frac{\mathrm{d}\Phi}{\mathrm{d}t} &= \frac{\mathrm{d}(\Phi,\mathcal{A})}{\mathrm{d}t} = \mathcal{D}(\Phi)\mathcal{R}_{\mathcal{A}}(\mathcal{C}(\Phi)) \\ &= \frac{\left([\mathrm{Br_2}] + \left[^{77}\mathrm{Br_2}\right]\right)^{1/2}}{[\mathrm{HBr}]\left([\mathrm{Br_2}] + \left[^{77}\mathrm{Br_2}\right] + k\,[\mathrm{HBr}]\right)}\,[\mathrm{H_2}][\mathrm{HBr}]\Bigg(\,[\mathrm{Br_2}](2\,\mathrm{HBr} - \mathrm{H_2} - \mathrm{Br_2}) \\ &\qquad\qquad + \left[^{77}\mathrm{Br_2}\right]\left(2\,\mathrm{H}^{77}\mathrm{Br} - \mathrm{H_2} - {}^{77}\mathrm{Br_2}\right)\Bigg) \\ &= \frac{[\mathrm{H_2}]\left([\mathrm{Br_2}] + \left[^{77}\mathrm{Br_2}\right]\right)^{1/2}}{1 + k\frac{[\mathrm{HBr}]}{[\mathrm{Br_2}]+\left[^{77}\mathrm{Br_2}\right]}}\Bigg(\frac{[\mathrm{Br_2}]}{[\mathrm{Br_2}] + \left[^{77}\mathrm{Br_2}\right]}\,(2\,\mathrm{HBr} - \mathrm{H_2} - \mathrm{Br_2}) \\ &\qquad\qquad + \frac{\left[^{77}\mathrm{Br_2}\right]}{[\mathrm{Br_2}] + \left[^{77}\mathrm{Br_2}\right]}\left(2\,\mathrm{H}^{77}\mathrm{Br} - \mathrm{H_2} - {}^{77}\mathrm{Br_2}\right)\Bigg).\end{aligned}$$

This shows how the coefficients of the difference matrix and the reaction rates combine to give the rate for each individual transition: the rate of the reaction depends on the overall concentration of each site regardless of the species carrying it, whereas each of the species with a given site's involvement in the reaction is determined by what proportion of the total site concentration it comprises. This is in contrast to the mass action case considered by Continuous π where we can calculate the reaction rate for each species with a given site separately (based on the species concentration) and add up the effects due to the multilinearity of the rate law [234].

3.4.3 Symbolic ODE Extraction

We have now characterized the dynamics of a system as a vector field, specifying the evolution vector $\frac{\mathrm{d}\Pi}{\mathrm{d}t} = \frac{\mathrm{d}(\Pi,\mathcal{A})}{\mathrm{d}t}$ given any vector $\Pi \triangleq \alpha_1 A_1 \parallel \ldots \parallel \alpha_n A_n$ of species concentrations. This description involves an uncountable infinity of evolution vectors (one for each combination of species concentrations), which, whilst no problem on a theoretical

level, is somewhat daunting from an implementation perspective. One way to alleviate this is abstracting Π to a symbolic mixture $\widetilde{\Pi} \triangleq [A_1]\, A_1 \parallel \ldots \parallel [A_n]\, A_n$, with variables $[A_1], \ldots, [A_n]$ for the concentration of each prime species, and calculate a symbolic evolution vector $\frac{\mathrm{d}\widetilde{\Pi}}{\mathrm{d}t} = \frac{\mathrm{d}(\widetilde{\Pi}, \mathcal{A})}{\mathrm{d}t}$ covering the whole vector field. However, in order to capture the evolution of the system over time, the general symbolic mixture $\widetilde{\Pi}$ must include all species reachable from the initial mixture Π; if species are contained in a finite dimensional subspace of $\mathbb{M}$, we can describe the dynamics of a model as a system of (finitely many) coupled differential equations. To extract this system of differential equations we first define the set of species supporting a given initial mixture.

Definition 3.4.9. For a mixture $\Pi = \sum [A]\, A \in \mathbb{M}$, the *support of* Π is the set of prime species defined by

$$\operatorname{supp} \Pi \triangleq \Big\{\, A \;\Big|\; [A] \neq 0 \,\Big\}.$$

We now use this to define the prime species reachable from a set of species $\mathcal{S}$.

Definition 3.4.10. Given a set of prime species $\mathcal{S} \subseteq \textsc{Prime}$, we define the set of prime species reachable from $\mathcal{S}$ in n steps inductively as

$$\operatorname{reach}_0(\mathcal{S}) \triangleq \mathcal{S} \qquad \operatorname{reach}_{n+1}(\mathcal{S}) \triangleq \operatorname{reach}_n(\mathcal{S}) \cup \bigcup_{\Phi \in \operatorname{span}(\operatorname{reach}_n(\mathcal{S}))} \operatorname{supp}\left(\frac{\mathrm{d}(\Phi, \mathcal{A})}{\mathrm{d}t}\right)$$

The *set of prime species reachable from* $\mathcal{S}$ may then be defined as,

$$\operatorname{reach}(\mathcal{S}) \triangleq \bigcup_{n=0}^{\infty} \operatorname{reach}_n(\mathcal{S}).$$

We are now ready to define the system of ODEs (with initial conditions) corresponding to a bond-calculus model.

Definition 3.4.11. Given a model $(\Pi_0, \mathcal{A})$ with $\Pi_0 \triangleq \alpha_1 A_1 \parallel \ldots \parallel \alpha_n A_n$ given constant initial conditions $\alpha_1, \ldots, \alpha_n \in \mathbb{R}_{\geq 0}$, with finite set of reachable species $\operatorname{reach}(\langle \Pi \rangle) = \{A_1, \ldots, A_n, B_1, \ldots, B_m\}$, we abstract Π_0 to the symbolic mixture,

$$\widetilde{\Pi} \triangleq [A_1]\, A_1 \parallel \ldots \parallel [A_n]\, A_n \parallel [B_1]\, B_1 \parallel \ldots \parallel [B_m]\, B_m.$$

We then define a symbolic initial value problem corresponding to $(\Pi_0, \mathcal{A})$ as,

$$\frac{\mathrm{d}\widetilde{\Pi}}{\mathrm{d}t} = \frac{\mathrm{d}(\widetilde{\Pi}, \mathcal{A})}{\mathrm{d}t}; \quad \widetilde{\Pi}_0 = \Pi_0.$$

This may be expanded as a system of coupled ODEs,

$$\frac{\mathrm{d}[A_i]}{\mathrm{d}t} = f_i([A_1], \ldots, [A_n], [B_1], \ldots, [B_m]) \qquad [A_i]_0 = \alpha_i \qquad (i = 1, \ldots, n)$$
$$\frac{\mathrm{d}[B_j]}{\mathrm{d}t} = g_j([A_1], \ldots, [A_n], [B_1], \ldots, [B_m]) \qquad [B_j]_0 = 0 \qquad (j = 1, \ldots, m)$$

where

$$\frac{\mathrm{d}(\widetilde{\Pi}, \mathcal{A})}{\mathrm{d}t} = \sum_{i=1}^{n} f_i([A_1], \ldots, [A_n], [B_1], \ldots, [B_m])\, A_i + \sum_{j=1}^{m} g_j([A_1], \ldots, [A_n], [B_1], \ldots, [B_m])\, B_j.$$

In fact, our compositional semantics gives Algorithm 2, an efficient procedure for incrementally deriving the extended symbolic process $\widetilde{\Pi}$ and its evolution vector.

Input: Initial symbolic mixture $\widetilde{\Pi} \triangleq [A_1]\, A_1 \parallel \ldots \parallel [A_n]\, A_n$ and affinity network $\mathcal{A}$.
Output: A pair $\left(\widetilde{\Pi}, \frac{\mathrm{d}(\widetilde{\Pi}, \mathcal{A})}{\mathrm{d}t}\right)$ of a symbolic mixture $\widetilde{\Pi}$ and its evolution vector $\frac{\mathrm{d}(\widetilde{\Pi}, \mathcal{A})}{\mathrm{d}t}$.
while $\mathrm{supp}\left(\frac{\mathrm{d}\widetilde{\Pi}}{\mathrm{d}t}\right) \not\subseteq \mathrm{supp}\left(\widetilde{\Pi}\right)$ **do**
  Pick some $B \in \mathrm{supp}\left(\frac{\mathrm{d}\widetilde{\Pi}}{\mathrm{d}t}\right) \setminus \mathrm{supp}\left(\widetilde{\Pi}\right)$
  $\widetilde{\Pi}' := \widetilde{\Pi} \parallel [B]\, B$
  $\exp_{\odot}(\mathcal{T}(\widetilde{\Pi}')) := \exp_{\odot}(\mathcal{T}(\widetilde{\Pi})) \odot \exp_{\odot}(\mathcal{T}([B]\, B))$
  $\mathcal{D}(\widetilde{\Pi}') := \mathcal{D}(\widetilde{\Pi}) + \mathcal{D}([B]\, B) + (\exp_{\odot}(\mathcal{T}(\widetilde{\Pi})) - \mathbf{1}) \oplus (\exp_{\odot}(\mathcal{T}([B]\, B)) - \mathbf{1})$
  $\mathcal{C}(\widetilde{\Pi}') := \mathcal{C}(\widetilde{\Pi}) + \mathcal{C}([B]\, B)$
  $\frac{\mathrm{d}(\widetilde{\Pi}', \mathcal{A})}{\mathrm{d}t} := \mathcal{D}(\widetilde{\Pi}')\mathcal{R}_{\mathcal{A}}(\mathcal{C}(\widetilde{\Pi}'))$

Algorithm 2: Mixture expansion algorithm. We utilize Propositions 3.4.3 and 3.3.9 to compute only the new terms of $\mathcal{D}$ at each step. Provided this terminates we obtain the complete symbolic vector field $\frac{\mathrm{d}\widetilde{\Pi}}{\mathrm{d}t} = \frac{\mathrm{d}(\widetilde{\Pi}, \mathcal{A})}{\mathrm{d}t}$.

Finally we demonstrate the symbolic ODE extraction algorithm in the special case of our enzyme system.

Example 3.4.12. In the HBr system we have $\mathrm{reach}(\mathrm{supp}(\Pi)) = \mathrm{supp}(\Pi) = \{\mathrm{H_2}, \mathrm{Br_2}, \mathrm{HBr}\}$ and so this model corresponds to a system of three ODEs,

$$\frac{\mathrm{d}[\mathrm{H_2}]}{\mathrm{d}t} = \frac{\mathrm{d}[\mathrm{Br_2}]}{\mathrm{d}t} = -\frac{[\mathrm{H_2}]\,[\mathrm{Br_2}]^{1/2}}{1 + k\frac{[\mathrm{HBr}]}{[\mathrm{Br_2}]}} \qquad \frac{\mathrm{d}[\mathrm{HBr}]}{\mathrm{d}t} = \frac{2\,[\mathrm{H_2}]\,[\mathrm{Br_2}]^{1/2}}{1 + k\frac{[\mathrm{HBr}]}{[\mathrm{Br_2}]}}.$$

Example 3.4.13. Whilst previously we included the dynamic complex HBr $\triangleq (\nu\,\ell)\left(\mathrm{H}^{(\ell)} \mid \mathrm{B}^{(\ell)}\right)$ explicitly in the definition of the Hydrogen Dibromide formation process, this was not in fact necessary. Suppose instead we had,

$$\Phi \triangleq [\mathrm{H_2}]_0\,\mathrm{H_2} \parallel [\mathrm{Br_2}]_0\,\mathrm{Br_2}$$

then HBr appears dynamically in $\frac{\mathrm{d}\Phi}{\mathrm{d}t}$ giving

$$\begin{aligned}
\mathrm{reach}_0(\mathrm{supp}(\Phi)) &= \mathrm{span}\{\mathrm{H_2}, \mathrm{Br_2}\} \\
\mathrm{reach}_1(\mathrm{supp}(\Phi)) &= \mathrm{span}\{\mathrm{H_2}, \mathrm{Br_2}, \mathrm{HBr}\} \\
\mathrm{reach}_2(\mathrm{supp}(\Phi)) &= \mathrm{span}\{\mathrm{H_2}, \mathrm{Br_2}, \mathrm{HBr}\} \\
\vdots \qquad &= \qquad \vdots
\end{aligned}$$

and hence

$$\mathrm{reach}(\mathrm{supp}(\Phi)) = \mathrm{span}\{\mathrm{H_2}, \mathrm{Br_2}, \mathrm{HBr}\}.$$

Therefore $\widetilde{\Phi} = \widetilde{\Pi}$ and we get the same ODEs as in the previous example with initial condition $[\mathrm{HBr}]_0 = 0$.

Chapter 4

Modelling Patterns of Gene Regulation in the Bond-Calculus

In this chapter we explore different modelling styles for capturing complex patterns of interaction through models of gene regulation. Firstly, in Section 4.1 we apply the bond-calculus to molecular-level modelling of cooperative interactions at the λ-switch based on the classic Stochastic π-model of Kuttler and Niehren [230], demonstrating how affinity patterns are able to transparently encode context-dependent interactions. Then in Section 4.2 we turn to network-level modelling and demonstrate the ability of the bond-calculus to combine a general purpose agent-centric model of gene regulation with affinity patterns specifying the structure of a particular network. Finally in Section 4.2.3, we give an example of style applied to a compact model of the complex plant circadian clock [135, 136].

The results of this chapter have been published in the paper [359].

4.1 Molecular-Level Modelling: Cooperativity at the λ-switch

In this section we will demonstrate how the bond-calculus can be used to build detailed mechanistic models of gene regulation, through the running example of the lysis-lysogenesis decision circuit of λ-phage infected E. Coli (the *λ-switch*). The λ-phage is a bacteriophage which infects E. Coli cells by inserting the λ-switch genetic circuit into their DNA, placing them into one of two growth phases: the *lysogenic phase* in which the viral DNA is passively reproduced by the normal reproduction of the E. Coli cells, and the *lytic phase*

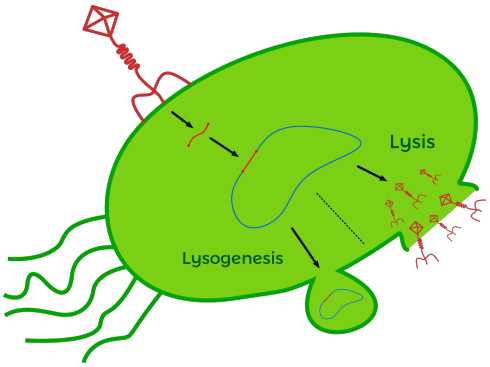


Figure 4.1: Schema of E. Coli infection by phage λ.

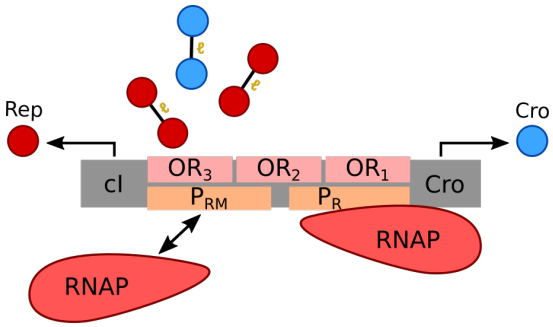


Figure 4.2: Gene regulation at the λ-switch.

when the virus reprograms the cell to produce many copies of the phage which are released upon the initiation of *lysis* breaking down the cell wall (see Fig. 4.2).

The λ-switch has been the subject of a number of mathematical models including [2, 13, 189, 190, 258, 325, 331, 342], and has became a standard benchmark for modelling gene regulation. Kuttler and Niehren's model of the λ-switch is a classic model of cooperative gene regulation in the Stochastic π-calculus [230] and provides the basis of our model. Unlike individual level stochastic process calculi, the bond-calculus takes a continuous view of biological systems, modelling the state of a system via the concentration of each species of agents. Hence, a bond-calculus model allows us to analyse a system not only via stochastic simulation, but also by extracting a system of differential equations: in this section we will focus on these differential equations to compare them to the stochastic simulations from the original model [230]. Since π-based process calculi rely on a binary communication mechanism, they require cooperative interactions involving multiple sites of the operator region to be modelled as internal state updates with instantaneous reaction rates (modelled via update channels [117, 118, 215, 230] or the visitor pattern [229]) — other potential methods of modelling such interactions include transactions [127] or priorities [353]. This raises a question of how to model the switch in continuous process calculi such as Continuous π and the bond-calculus whose differential equation semantics do not include instantaneous rates, however, we will see that *affinity patterns* are expressive enough to capture cooperativity directly as a type of multiway synchronisation. This is somewhat similar to how rule-based languages capture cooperativity using schemes of reaction rules spanning multiple sites [35, 228], however, whilst rules specify the whole effect of the reaction, affinity patterns are more narrowly focused on the interaction capacities of sites, leaving the effect of reactions to be determined by the agents involved,

similarly to traditional process calculi.

4.1.1 Model Preliminaries

The mechanisms of transcription regulation at the λ-switch underlying Kuttler and Niehren's model are described in depth in [230]; here we will recall some of the key features of the switch. The dynamics of gene regulation at the λ-switch should implement a bistable switch [325] based on the levels of the two proteins, Rep and Cro, exhibiting either high levels of Rep and the exclusion of Cro (leading to *lysogeny*), or high levels of Cro and the exclusion of Rep (leading to *lysis*). The protein Rep is produced from the gene cI by the binding of RNA Polymerase (RNAP) to its promoter region $\mathrm{P_{RM}}$, whilst the protein Cro is produced by the binding of RNAP to its promoter region $\mathrm{P_R}$ (Fig. 4.2). The proteins Rep and Cro form into dimers which act as repressors by binding to each of the three operator regions $\mathrm{OR_1}$, $\mathrm{OR_2}$, and $\mathrm{OR_3}$ which overlap with the promoter sites for $\mathrm{P_R}$ and $\mathrm{P_{RM}}$, so that the binding of repressors at $\mathrm{OR_3}$ and RNAP at $\mathrm{P_{RM}}$ are mutually exclusive, and the binding of repressors at $\mathrm{OR_1}$ and $\mathrm{OR_2}$ and RNAP at $\mathrm{P_R}$ are mutually exclusive. The final key component of the switch is *cooperative binding*: the binding of a Rep dimer at $\mathrm{OR_1}$ significantly increases the affinity for the Rep dimer at $\mathrm{OR_2}$; in this way a Rep dimer at $\mathrm{OR_1}$ recruits another at $\mathrm{OR_2}$.

All values of the model's rate parameters are taken from [230, Fig. 4], with the exception of two new derived parameters defined as the following arithmetic combinations of rate parameters,

$$\begin{aligned} \texttt{Kd\_OR2\_boost} &\triangleq \texttt{Kd\_OR2\_rep} - \texttt{Kd\_OR2\_rep\_coop} = 3.835 \\ \texttt{Kf\_prm\_boost} &\triangleq \texttt{Kf\_prm\_promoted} - \texttt{Kf\_prm} \qquad = 0.081 \end{aligned}$$

and the following parameter for Rep degredation, which is set to zero,

$$\texttt{Kd\_rep} \triangleq 0.0.$$

To allow direct comparison between our model and the existing stochastic model, we will assume throughout that our units of concentration have been rescaled to coincide with copy numbers; under these units the macroscopic and stochastic rate parameters coincide.

4.1.2 Modelling Autoreactive Sites: Repressor Dimerization

A key feature of λ-switch is the binding of pairs of Rep proteins (or equally pairs of Cro proteins) to form dimers, which then act as repressors. The schema of reaction among Rep

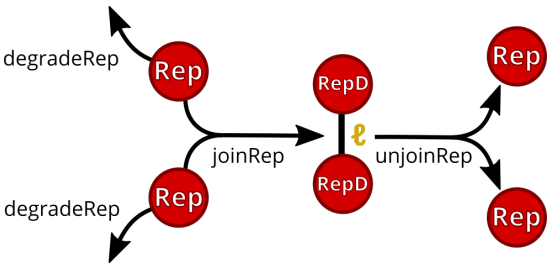


Figure 4.3: Schematic of dimerization.

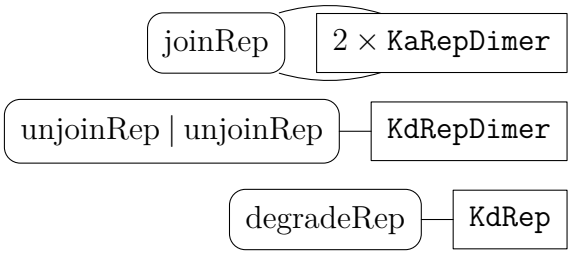


Figure 4.4: Affinity network $\mathcal{A}_{\text{REP}}$ for dimerisation reactions.

proteins is shown in Fig. 4.3. The unbound Rep protein can either degrade by interacting on the degradeRep site, causing it to decay into nothing, or bind to another copy of itself by interacting on the joinRep site. This type of homodimerization reaction in gene regulation, plays an important role in controlling noise, and can be one source of cooperativity at binding sites [81]. Despite the ubiquity of this mechanism, it is not a particularly natural fit for the communication mechanisms of traditional π-based calculi [230], since the symmetrical nature of the reaction needs to be broken into two halves modelling Rep by an agent such as

$$\text{REP} \triangleq \text{joinRep!}.\text{REP}_2 + \text{joinRep?}.\mathbf{0}$$

which offers itself a choice between sending on the joinRep! site or receiving on the joinRep? site. This departs from the underlying chemistry firstly in breaking the symmetry of the underlying reaction, but also, once we add in the quantitative reaction rate `KaRepDimer`, splitting the reaction site into two might be expected to result in twice the reaction rate expected from the law of mass action [213, 214, 291][1].

In contrast, in the bond-calculus we are able to model the Rep protein as agents representing the bound/unbound state of the protein,

$$\text{REP} \triangleq \text{degradeRep}.\mathbf{0} + \text{joinRep}(\ell).\text{REPD}(\ell)$$
$$\text{REPD}(\ell) \triangleq \text{unjoinRep@}\ell.\text{REP} + \text{bindRep@}\ell.\mathbf{0}$$

On dimerization each REP agent and its binding partner will agree on a shared location ℓ and each will transition into the bound state $\text{REPD}(\ell)$: overall this will result in a dynamic complex, $\text{REP}_2 \triangleq (\nu\,\ell)(\text{REPD}(\ell) \mid \text{REPD}(\ell))$, as a parallel composition ($|$) of

[1]Some stochastic process calculi semantics including the original reduction semantics for Stochastic π (but not subsequent versions of the language [291, 305]) remedy this with a special rule for homodimerizations, however, this approach breaks the strict correspondence between channels (or rather, channel ends) and protein domains and does not extend to more general interactions.

two bound subunits bound together by a shared location ℓ. In its bound state, $\mathrm{REPD}(\ell)$, a Rep molecule may either become unbound from the dimer by communicating on joinRep, or bind again, this time to an operator of the λ-switch[2]. The interactions capacities of these agent's reaction sites follow the affinity network,

$$\mathcal{A}_{\mathrm{REP}} = \left\{ \begin{array}{ll} \text{joinRep} \parallel \text{joinRep} & @\ 2 \times \texttt{KaRepDimer}, \\ \text{unjoinRep} \mid \text{unjoinRep} & @\ \texttt{KdRepDimer}, \\ \text{degradeRep} & @\ \texttt{KdRep} \end{array} \right\}$$

Here the pattern joinRep ∥ joinRep specifies that the joinRep is compatible with itself, resulting in a dimerization reaction between two separate molecules containing a joinRep site, whilst the pattern unjoinRep|unjoinRep results in an undimerization reaction involving a single molecule with two parallel components, each having a unjoinRep site. Importantly, since our model faithfully captures the fact that the dimerization reaction is between multiple instances of a single site on a single species, our semantics halves the resultant rate in accordance with the law of mass action, and accordingly our rate parameter is twice that specified in the Stochastic π model.

We can also view the affinity network graphically as the hypergraph in Fig. 4.4, by considering each cluster of sites as a node, and patterns as hyperedges (this directly generalises the affinity networks of Continuous π, which are labelled graphs).

The above model demonstrates how the combination of affinity patterns and our communication mechanism based on location agreement can capture the symmetrical nature of dimerization. Another advantage of this symmetry is that it allows us to, for example, consider variants of the mechanism such as replacing dimerization with tetramerization (as considered in [80]) simply by replacing the patterns for binding/unbinding with,

$$\text{joinRep} \parallel \text{joinRep} \parallel \text{joinRep} \parallel \text{joinRep} \ @\ \mathrm{MA}_{k_1}$$
$$\text{unjoinRep}|\text{unjoinRep}|\text{unjoinRep}|\text{unjoinRep} \ @\ \mathrm{MA}_{k_{-1}}$$

without the need to modify the definitions of the agents involved in the reactions. Thus, affinity patterns effectively separate how agents respond to communication at given sites, and the patterns of interaction these sites engage in.

We may use the bond-calculus tool to perform stochastic simulation, or extract a system of ODEs describing the dynamics of dimerization. Two ODE simulation results showing how the equilibrium shifts depending on whether we start at low or high Rep

[2]For simplicity, we will not model dynamic binding of Reps to operators explicitly, as we have done for dimerization and as is done in [230].

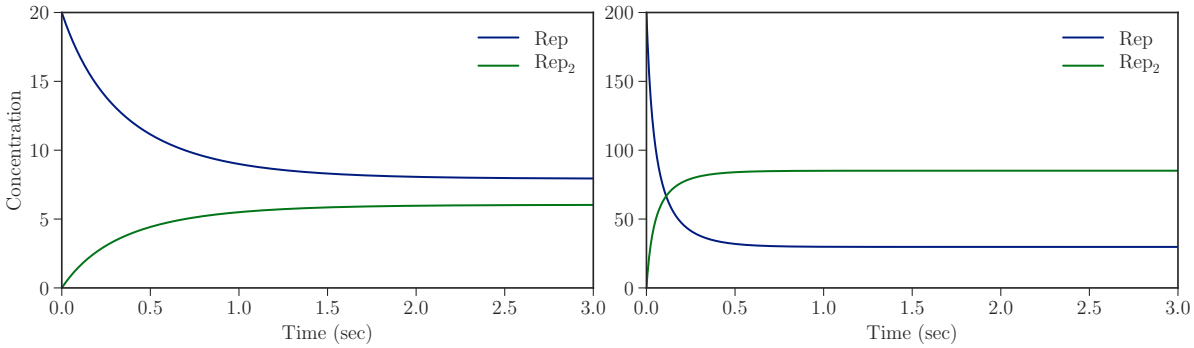


Figure 4.5: Rep, starting from low level (left) and a high level (right); this reproduces the concentration dependent equilibrium of Rep and matches [230, Fig. 25].

concentration are shown in Fig. 4.5; these graphs are consistent with the stochastic simulation results shown in [230, Fig. 25].

The definitions for Cro and its affinity network $\mathcal{A}_{\mathrm{CRO}}$ are nearly identical to those for Rep except the sites are renamed appropriately.

4.1.3 Modelling the Switch: Agents

We now get to the heart of the model, the λ-switch itself. Just as in Fig. 4.2, the λ-switch is described as consisting of the three operators, OR_1, OR_2, OR_3, and the two promoters, P_{RM} and P_{R}; this is captured in our model as a parallel composition of individual agents representing each of these operators and promoters, bound together at a shared location ℓ,

$$\textsc{Switch} \triangleq (\nu\,\ell)(\mathrm{P}_{\mathrm{RM}}(\ell) \mid \mathrm{OR}_3(\ell) \mid \mathrm{OR}_1(\ell) \mid \mathrm{OR}_2(\ell) \mid \mathrm{P}_{\mathrm{R}}(\ell))$$

We must now give definitions for each of these constituent agents. We start with the operators OR_i $(i = 1, 2, 3)$. These will each have three possible states: the unbound state

OR_i, and the bound states $\mathrm{OR}_i\textsc{Rep}$ and $\mathrm{OR}_i\textsc{Cro}$ for Rep and Cro respectively.

$$
\begin{aligned}
\mathrm{OR}_i(\ell) \triangleq\ & \mathrm{bindOR}_i\mathrm{Cro}@\ell(m).\mathrm{OR}_i\textsc{Cro}(m) \\
& + \mathrm{bindOR}_i\mathrm{Rep}@\ell(m).\mathrm{OR}_i\textsc{Rep}(m) \\
& + \mathrm{unboundOR}_i@\ell.\mathrm{OR}_i(\ell) \\
& + \mathrm{noRepOR}_i@\ell.\mathrm{OR}_i(\ell) \\
\mathrm{OR}_i\textsc{Rep}(\ell) \triangleq\ & \mathrm{unbindOR}_i\mathrm{Rep}@\ell(m).(\mathrm{OR}_i(\ell) \mid \textsc{RepD}(m) \mid \textsc{RepD}(m)) \\
& + \mathrm{boundOR}_i@\ell.\mathrm{OR}_i\textsc{Rep}(\ell) \\
& + \mathrm{hasRepOR}_i@\ell.\mathrm{OR}_i\textsc{Rep}(\ell) \\
\mathrm{OR}_i\textsc{Cro}(\ell) \triangleq\ & \mathrm{unbindOR}_i\mathrm{Cro}@\ell(m).(\mathrm{OR}_i(\ell) \mid \textsc{CroD}(m) \mid \textsc{CroD}(m)) \\
& + \mathrm{boundOR}_i@\ell.\mathrm{OR}_i\textsc{Cro}(\ell) \\
& + \mathrm{noRepOR}_i@\ell.\mathrm{OR}_i(\ell)
\end{aligned}
$$

Note that unlike in previous models, we have a single uniform definition for each operator, which makes no mention of the reaction rates, or updates of neighbouring sites; this is because we will describe all of the quantitative features of the operators in the affinity network. We should note that as well as the sites for binding/unbinding, operators indicate their binding status to the other sites using their $\mathrm{boundOR}_i$/$\mathrm{unboundOR}_i$ sites, and, more specifically, whether they are bound to a Rep molecule using their $\mathrm{hasRepOR}_i$/$\mathrm{noRepOR}_i$ sites. Next we describe the agent for the promoter $\mathrm{P_{RM}}$,

$$
\begin{aligned}
\mathrm{P_{RM}}(\ell) \triangleq\ & \mathrm{bindP_{RM}}@\ell.\mathrm{P_{RM}}\textsc{Bound}(\ell) \\
& + \mathrm{unboundP_{RM}}@\ell.\mathrm{P_{RM}}(\ell) \\
\mathrm{P_{RM}}\textsc{Bound}(\ell) \triangleq\ & \mathrm{unbindP_{RM}}@\ell.(\mathrm{P_{RM}}(\ell) \mid \mathrm{RNAP}) \\
& + \mathrm{transcribeRep}@\ell.(\mathrm{MRNA_{cI}} \mid \mathrm{P_{RM}}(\ell) \mid \mathrm{RNAP})
\end{aligned}
$$

The agent for the promoter $\mathrm{P_R}$ is defined nearly identically (with appropriately renamed channels). Finally, the RNAP agent is defined as

$$\mathrm{RNAP} \triangleq \mathrm{bindRNAP}.\mathbf{0}$$

4.1.4 Modelling the Switch: Affinity Network

It now remains to give affinity networks, specifying the dynamics of interactions at the λ-switch. Interactions at the operators are specified in the networks $\mathcal{A}_{\mathrm{OR}_1}$, $\mathcal{A}_{\mathrm{OR}_2}$, and $\mathcal{A}_{\mathrm{OR}_3}$. For example, Rep_2 binding is specified via the patterns in Fig. 4.6, allowing Rep

dimers to bind to the sites at OR_1, OR_2, and OR_3, whilst ensuring mutual exclusion with RNAP at $\mathrm{P_R}/\mathrm{P_{RM}}$ by matching against the $\mathrm{unboundP_R}/\mathrm{unboundP_{RM}}$ sites. The full affinity network for $\mathrm{O\,R}_1$ is,

$$\mathcal{A}_{\mathrm{OR}_1} = \left\{ \begin{array}{ll} \mathrm{bindOR}_1\mathrm{Cro}|\mathrm{unboundP_R} \parallel \mathrm{bindCro}|\mathrm{bindCro} & @\ \texttt{Ka\_protein}, \\ \mathrm{bindOR}_1\mathrm{Rep}|\mathrm{unboundP_R} \parallel \mathrm{bindRep}|\mathrm{bindRep} & @\ \texttt{Ka\_protein}, \\ \mathrm{unbindOR}_1\mathrm{Rep} & @\ \texttt{Kd\_OR1\_rep}, \\ \mathrm{unbindOR}_1\mathrm{Cro} & @\ \texttt{Kd\_OR1\_cro} \end{array} \right\}$$

Here, in addition to the patterns for Rep/Cro binding, we have unary patterns $\mathrm{bindOR}_1\mathrm{Rep}$ @ `Kd_OR1_rep` and $\mathrm{bindOR}_1\mathrm{Cro}$ @ $\mathrm{MA}_{\texttt{Kd\_OR1\_cro}}$ which specify the rates of unbinding for Rep/Cro dimers.

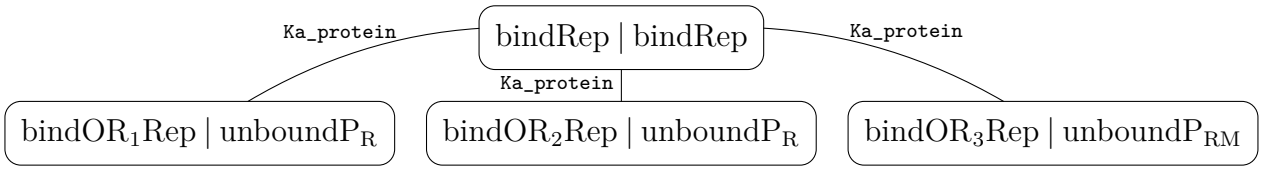


Figure 4.6: Affinity patterns for binding of Rep dimers to the operators.

The full affinity network for OR_2 is,

$$\mathcal{A}_{\mathrm{OR}_2} = \left\{ \begin{array}{ll} \mathrm{bindOR}_2\mathrm{Cro}|\mathrm{unboundP_R} \parallel \mathrm{bindCro}|\mathrm{bindCro} & @\ \texttt{Ka\_protein}, \\ \mathrm{bindOR}_2\mathrm{Rep}|\mathrm{unboundP_R} \parallel \mathrm{bindRep}|\mathrm{bindRep} & @\ \texttt{Ka\_protein}, \\ \mathrm{unbindOR}_2\mathrm{Rep} & @\ \texttt{Kd\_OR2\_rep}, \\ \mathrm{unbindOR}_2\mathrm{Rep}|\mathrm{noRepOR}_1 & @\ \texttt{Kd\_OR2\_boost}, \\ \mathrm{unbindOR}_2\mathrm{Cro} & @\ \texttt{Kd\_OR2\_cro} \end{array} \right\}$$

Here the pattern $\mathrm{unbindOR}_2\mathrm{Rep}|\mathrm{noRepOR}_1$ @ `Kd_OR1_boost` captures cooperativity at OR_2 by decreasing the dissociation rate by `Kd_OR1_boost` whenever Rep is not bound at OR_1. The affinity network $\mathcal{A}_{\mathrm{OR}_3}$ is analogous to $\mathcal{A}_{\mathrm{OR}_1}$ except that binding is mutually exclusive with $\mathrm{P_{RM}}$ rather than $\mathrm{P_R}$ (Fig. 4.6) so we will not list it in full.

Finally, interactions at the promoters are specified in the two networks $\mathcal{A}_{\mathrm{P_R}}$ and $\mathcal{A}_{\mathrm{P_{RM}}}$. The key elements of these are the patterns for RNAP binding shown in Fig. 4.7, which specify that RNAP may bind to $\mathrm{P_{RM}}$ whenever both OR_1 and OR_2 are unbound, and to

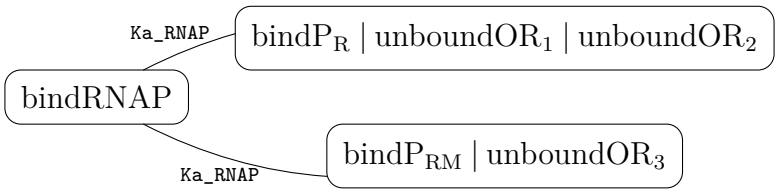


Figure 4.7: Affinity patterns for RNAP binding.

P_R whenever both OR_3 is unbound. The full network for P_R is defined as

$$\mathcal{A}_{P_R} = \left\{ \begin{array}{ll} \text{bindP}_R|\text{unboundOR}_1|\text{unboundOR}_2 \parallel \text{bindRNAP} & @\ \texttt{Ka\_RNAP}, \\ \text{unbindP}_R & @\ \texttt{Kd\_PR\_RNAP}, \\ \text{transcribeCro} & @\ \texttt{Kf\_pr} \end{array} \right\}$$

which has additional patterns for the unbinding of RNAP and for the transcription of Cro, whilst the full network for P_{RM} is defined as,

$$\mathcal{A}_{P_{RM}} = \left\{ \begin{array}{ll} \text{bindP}_{RM}|\text{unboundOR}_3 \parallel \text{bindRNAP} & @\ \texttt{Ka\_RNAP}, \\ \text{unbindP}_{RM} & @\ \texttt{Kd\_PRM\_RNAP}, \\ \text{transcribeRep} & @\ \texttt{Kf\_prm}, \\ \text{transcribeRep}|\text{hasRepOR}_2 & @\ \texttt{Kf\_prm\_boost} \end{array} \right\}$$

In the network $\mathcal{A}_{P_{RM}}$, the extra pattern transcribeRep|hasRepOR$_2$ @ `Kf_prm_boost` captures cooperative modification by increasing the Rep transcription rate by `Kf_prm_boost` whenever Rep is bound at OR_2.

4.1.5 Overall Model

We can put together the components of our model by defining the mixture

$$\Pi_0 \triangleq 1 \cdot \textsc{Switch} \parallel [\textsc{Cro}] \cdot \textsc{Cro} \parallel [\textsc{Rep}] \cdot \textsc{Rep} \parallel [\mathrm{RNAP}] \cdot \mathrm{RNAP}$$

and defining the overall affinity network containing the patterns for each component,

$$\mathcal{A} \triangleq \mathcal{A}_{\textsc{Rep}} \cup \mathcal{A}_{\textsc{Cro}} \cup \mathcal{A}_{OR_1} \cup \mathcal{A}_{OR_2} \cup \mathcal{A}_{OR_3} \cup \mathcal{A}_{P_R} \cup \mathcal{A}_{P_{RM}}.$$

We can then use this bond-calculus description to generate more conventional mathematical models for simulation and analysis. Our tool translates this particular model into a chemical reaction network with 47 species and 181 reaction rules: of the 47 species, 40 correspond to the different possible binding states of the right operator of the λ-switch,

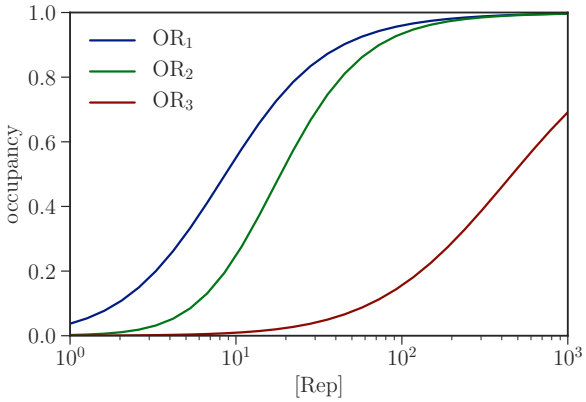


Figure 4.8: The mean occupancy of each operator over 5000 seconds, given initial Rep concentration [Rep], and no RNAP or Cro. Based on numerical simulation of bond-calculus ODEs: this matches [230, Fig. 28], which was generated using stochastic simulation.

and these match the states enumerated combinatorially in [331, Table 2]. The tool also generates a system of coupled ODEs associated to the model. These differential equations should be viewed with caution since the low copy numbers of molecules involved in gene regulation (especially the switch itself) mean that stochastic or hybrid stochastic models [67] might give a more faithful view of the dynamics. We also note that for more practical analysis of the system these ODEs could be reduced via appropriate equilibrium assumptions as in [35, 331], or by network level modelling using general kinetic laws as considered in the next section. Nevertheless, these ODEs give a useful indication of the mean behaviour of the switch under similar assumptions to existing thermodynamic models [2, 325]. In Fig. 4.8 we see the binding curves for repressors at each operator of the switch, computed by simulation of the ODEs at different levels of Rep concentration; this matches [230, Fig. 28], which records the mean behaviour of the original Stochastic π model.

4.2 Network-Level Modelling

In Section 4.1 we saw how gene regulation can be modelled at a molecular level, by modelling all possible binding states of the regulatory region. This level of detail is not, however, necessary to give a useful model of the regulatory interactions in a network. In this section we will give a general purpose, high level model of gene transcription and translation in the bond-calculus, and show how affinity patterns and general kinetic laws may be used to capture the patterns of regulation which interconnect them. We will then

look at the specific example of the Plant Circadian Clock [135].

4.2.1 Modelling the Central Dogma

To begin our modelling of gene regulation we need to define species capturing the agents involved in the production of a generic protein which we will denote X. To this end we will define three species: GENE_X which denotes the gene encoding X, mRNA_X which denotes the RNA form of X, and PROTEIN_X which denotes the protein form of X. We define these species as follows,

$$\begin{aligned}\text{GENE}_X &\triangleq c_X.(\text{GENE}_X \mid \text{mRNA}_X)\\ \text{mRNA}_X &\triangleq dM_X.\mathbf{0} + t_X.(\text{mRNA}_X \mid \text{PROTEIN}_X)\\ \text{PROTEIN}_X &\triangleq d_X.\mathbf{0} + i_X.\text{PROTEIN}_X.\end{aligned}$$

These species can interact at a number of sites according to the *central dogma of molecular biology*: interaction at site c_X causes GENE_X to be *transcribed* into its RNA form mRNA_X, whilst interaction at site t_X causes mRNA_X to be *translated* into its protein form PROTEIN_X. We also allow the mRNA_X to decay by interacting at site dM_X and for PROTEIN_X to decay by interacting at site d_X. Finally, PROTEIN_X has an additional site i_X which does not change its state, but will allow it to act as an activator or repressor for other reactions.

4.2.2 Gene Gates via General Kinetics and Affinity Patterns

In order to model the dynamics of concrete gene regulatory networks, we need to supplement the skeletal model of translation and transcription given in the previous section with an affinity network which specifies the dynamics of the network according to general kinetic laws. If we have a detailed knowledge of the mechanism of binding (similar to our model in Section 4.1), it is possible to derive suitable kinetic laws based on the probability of binding at equilibrium [2, 49, 50], however, in practice, since this knowledge is rarely available, a more pragmatic approach is pursued where kinetic laws are used as phenomenological models to fit experimental data. One common such model derives from the Hill equation [195], which approximates the occupancy of an operator O by a protein P as

$$f_{[P],K,n} = \frac{[P]^n}{K^n + [P]^n} = \frac{\left(\frac{[P]}{K}\right)^n}{1 + \left(\frac{[P]}{K}\right)^n}$$

where $K \in \mathbb{R}_{\geq 0}$ is the protein concentration producing 50% operator occupancy, and the Hill coefficient $n \in \mathbb{R}$ which measures the degree of cooperativity. Whilst the Hill equation was originally derived as an equilibrium model for cooperative binding at n binding sites [195], in practice the Hill coefficient n is used to capture many different types of positive cooperativity and rarely corresponds precisely to the number of binding sites [357].

We may then define the general kinetic laws,

$$\begin{aligned}\mathrm{Hill}^{+}_{v,K,n}([G],[P]) &\triangleq v f_{[P],K,n}\,[G]\\ \mathrm{Hill}^{-}_{v,K,n}([G],[P]) &\triangleq v(1-f_{[P],K,n})\,[G]\end{aligned}$$

which give the rate of transcription of a gene G under positive or negative regulation by protein P respectively.

For example, this allows us to model a simple gene regulatory network where transcription of B is activated by A and A is repressed by B (Fig. 4.9) using the affinity patterns,

$$\begin{aligned}c_A \parallel i_B \;\; @\; \mathrm{Hill}^{+}_{k_1,K_1,n_1},\\ c_B \parallel i_A \;\; @\; \mathrm{Hill}^{-}_{k_2,K_2,n_2},\end{aligned}$$

plus extra patterns to account for degradation and translation, and the mixture defined by,

$$\Pi \triangleq [A]\,\textsc{Gene}_A \parallel [B]\,\textsc{Gene}_B.$$

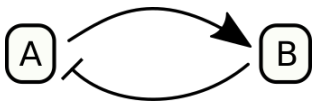


Figure 4.9: A simple gene regulatory network, exhibiting positive and negative regulation.

For the case of a general gene gate with n activators $A_1,\ldots,A_n$ and m inhibitors $I_1,\ldots,I_m$, we can use generalised Hill-type laws[3] such as [135],

$$\mathrm{Hill}_{\boldsymbol{v};\boldsymbol{l};\boldsymbol{k};\boldsymbol{s};\boldsymbol{t}}([G];[A_1],\ldots,[A_m];[I_1],\ldots,[I_n]) \triangleq \frac{v_0+\frac{v_1\left(\frac{[A_1]}{l_1}\right)^{s_1}+\ldots+v_m\left(\frac{[A_m]}{l_m}\right)^{s_m}}{1+\left(\frac{[A_1]}{l_1}\right)^{s_1}+\ldots+\left(\frac{[A_m]}{l_m}\right)^{s_m}}}{1+\left(\frac{[I_1]}{k_1}\right)^{t_n}+\ldots+\left(\frac{[I_n]}{k_n}\right)^{t_n}}\,[G]$$

Here we have a vector of parameters $\boldsymbol{v}=(v_0,v_1,\ldots,v_m)$ for the activation velocities, $\boldsymbol{l}=(l_1,\ldots,l_m)$ and $\boldsymbol{k}=(k_1,\ldots,k_n)$ for the activation/repression fractional occupancies, and $\boldsymbol{s}=(s_1,\ldots,s_m)$ and $\boldsymbol{t}=(t_1,\ldots,t_n)$ for the Hill coefficients for activators/inhibitors. For the remainder of this paper we will consider only Hill coefficients $s_1=\ldots=s_m=t_1=\ldots=t_n=2$, and write simply, $\mathrm{Hill}_{\boldsymbol{v};\boldsymbol{l};\boldsymbol{k}}$ for the corresponding kinetic laws.

[3]Here we have used the same form of Hill-type function as [135], however, there are many variants in use in the literature e.g. [49, 202, 301, 343].

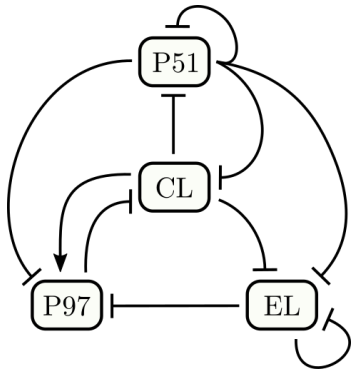


Figure 4.10: Compact gene regulation network for the plant circadian clock of [135, Fig. 1]: each gene represents two genes of the underlying network.

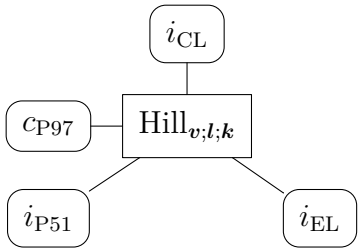


Figure 4.11: Affinity pattern for transcription regulation at P97 where $\boldsymbol{v} = (v_{2A}, v_{2B})$, $\boldsymbol{l} = (K_3)$, and $\boldsymbol{k} = (K_4, K_5)$.

4.2.3 Example: Compact Plant Circadian Clock

In this section we will look at the plant circadian clock, a complex genetic circuit consisting of at least a dozen genes sustaining regular oscillations through a number of interconnected positive and negative transcriptional feedback loops [273]. The circadian clock of model organism *Arabidopsis thaliana* in particular has been the subject of a series of increasingly sophisticated models over the last decade [164, 246, 301, 302, 327, 362]. Bio-PEPA has been used to analyse the stochastic properties of plant circadian clock [185] starting from Pokhilko et al's continuous model [302], along with several other clocks [5, 6, 21, 125], whilst the Kai-based (non-transcriptional) circadian clock [21, 234] is a standard case study for Continuous π. Here we will take as our basis the compact model of [135, 136] which expresses qualitative behaviour of the clock using only four species (Fig. 4.10), and for simplicity we restrict our model to constant daylight conditions (the original model includes separate rates for light/dark conditions [136] which can be modelled either as discrete events [122, 173] or smooth transition functions [185, 320]).

We start our model with a mixture covering all of the genes in the network, constructed by instantiating the species defined in Section 4.2.1.

$$\Pi_0 \triangleq 1 \cdot \textsc{Gene}_{\mathrm{CL}} \parallel 1 \cdot \textsc{Gene}_{\mathrm{P97}} \parallel 1 \cdot \textsc{Gene}_{\mathrm{P51}} \parallel 1 \cdot \textsc{Gene}_{\mathrm{EL}}$$

These species only represent the various agents implementing the mechanism of gene regulation. The main regulatory interactions between the various genes are specified

separately via the affinity network,

$$\begin{aligned}\mathcal{A}_{\mathrm{REG}} = \Big\{\, & c_{\mathrm{CL}} \parallel i_{\mathrm{P97}} \parallel i_{\mathrm{P51}} && @\ \mathrm{Hill}_{(v_1);;(K_1,K_2)},\\ & c_{\mathrm{P97}} \parallel i_{\mathrm{CL}} \parallel i_{\mathrm{P51}} \parallel i_{\mathrm{EL}} && @\ \mathrm{Hill}_{(v_{2A},v_{2B});(K_3);(K_4,K_5)},\\ & c_{\mathrm{P51}} \parallel i_{\mathrm{CL}} \parallel i_{\mathrm{P51}} && @\ \mathrm{Hill}_{(v_3);();(K_6,K_7)},\\ & c_{\mathrm{EL}} \parallel i_{\mathrm{CL}} \parallel i_{\mathrm{P51}} \parallel i_{\mathrm{EL}} && @\ \mathrm{Hill}_{(v_4);();(K_8,K_9,K_{10})} \,\Big\}\end{aligned}$$

These patterns assign sites as the arguments of the kinetic law in a positional manner so, for example, the affinity pattern $c_{\mathrm{CL}} \parallel i_{\mathrm{P97}} \parallel i_{\mathrm{P51}}$ @ $\mathrm{Hill}_{(v_1);();(K_1,K_2)}$ indicates that the transcription of the CL mRNA is inhibited by the P97 and P51 proteins whilst the affinity pattern $c_{\mathrm{P97}} \parallel i_{\mathrm{CL}} \parallel i_{\mathrm{P51}} \parallel i_{\mathrm{EL}}$ @ $\mathrm{Hill}_{(v_{2A},v_{2B});(K_3);(K_4,K_5)}$ (displayed graphically in Fig. 4.11) indicates that the transcription of the P97 mRNA is activated by the CL protein and is inhibited by the P51 and EL proteins (as in Fig. 4.10). Additionally the degradation and translation rates are captured in the network,

$$\begin{aligned}\mathcal{A}_{\mathrm{REST}} = \Big\{\, & dM_{\mathrm{CL}} \ @\ \mathrm{MA}_{k_{1L}}, && d_{\mathrm{CL}} \ @\ \mathrm{MA}_{d_1}, && t_{\mathrm{CL}} \ @\ \mathrm{MA}_{p_{1T}},\\ & dM_{\mathrm{P97}} @\ \mathrm{MA}_{k_2}, && d_{\mathrm{P97}} @\ \mathrm{MA}_{d_{2L}}, && t_{\mathrm{P97}} @\ \mathrm{MA}_{p_2},\\ & dM_{\mathrm{P51}} @\ \mathrm{MA}_{k_3}, && d_{\mathrm{P51}} @\ \mathrm{MA}_{d_{3L}}, && t_{\mathrm{P51}} @\ \mathrm{MA}_{p_3},\\ & dM_{\mathrm{EL}} \ @\ \mathrm{MA}_{k_4}, && d_{\mathrm{EL}} \ @\ \mathrm{MA}_{d_{4L}}, && t_{\mathrm{EL}} \ @\ \mathrm{MA}_{p_4} \,\Big\}\end{aligned}$$

This demonstrates the ability of bond-calculus to capture a general model of gene regulation as agents, whilst the affinity network captures the specific interactions in the network with dynamics following a general kinetic law.

Generating ODEs from this bond-calculus model gives a system of differential equations that coincides with the model of [135] under constant daylight conditions; Fig. 4.12 (left) shows the result of numerical simulation. This give a reasonable agreement with a single run of stochastic simulation in Fig. 4.12 (right), also generated from the bond-calculus model.

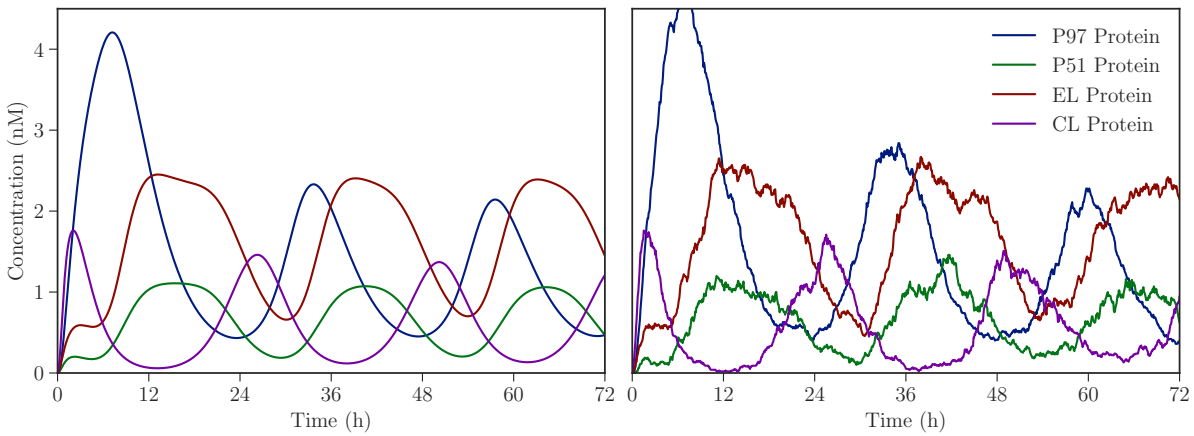


Figure 4.12: Plant circadian clock protein levels, with ODE simulation on the left (with parameter values based on [135]), and stochastic simulation with discrete levels of step size $h = 0.01$ on the right.

Chapter 5

Uncertain Models and Uncertain Contexts

In this chapter we introduce the *Logic for Behaviour in Uncertain Contexts (*$\mathcal{LBUC}$*)* for reasoning about the behaviour of bond-calculus models under uncertain contexts. This builds upon the existing logic $\mathcal{LBC}$ [19], which combined Signal Temporal Logic properties over Continuous π models with a *context operator*, $\mathcal{C} \triangleright \varphi$. $\mathcal{LBUC}$ extends this approach to handling different sources of uncertainty, whether they be the initial conditions and reaction rates of the system (expressed as an uncertain bond-calculus model), the imprecise numerical methods used for simulation, the context process $\mathcal{C}$, or the time point at which the context is introduced.

In Section 5.1 we begin by extending the syntax and semantics of the bond-calculus to encompass interval uncertainty in initial concentrations and reaction rate parameters. In Section 5.2 we introduce the syntax and semantics of $\mathcal{LBUC}$. Finally, Section 5.3 discusses the range of properties which $\mathcal{LBUC}$ allows us to express.

5.1 Uncertain Systems

In this section we will outline how bond-calculus models and their semantics are extended to express models involving uncertain initial conditions and reaction rates. We start by introducing uncertain mixtures, defined as follows,

Definition 5.1.1. An *uncertain mixture* P, Q is defined according to the grammar,

$$P, Q \ ::= \ I \cdot S \ \mid \ P \parallel Q$$

where $I = [a, b] \in \mathbb{IR}$ is a real interval defining the range of possible concentrations for a species S.

We reinterpret the syntactic and semantic rules introduced in Section 3.3 over uncertain mixtures, using interval arithmetic in place of real arithmetic in the definition of structural congruence $\equiv$ over uncertain mixtures. The mixture space $\textsc{Mix}$ is similarly generalized to an interval vector space $\widehat{\textsc{Mix}}$ of uncertain mixtures.

It remains to define uncertain affinity networks. We start by generalising rate laws and rate law families to be interval functions.

Definition 5.1.2. An *interval rate law* consists of an interval function $\widehat{\mathrm{R}} : \mathbb{IR}^n \to \mathbb{IR}$.

Definition 5.1.3. An *interval rate law family* consists of an interval function $\widehat{\mathrm{R}} : \mathbb{IR}^m \to \mathbb{IR}^* \to \mathbb{IR}$.

We can also relate standard rate laws to their interval variants by defining when the latter is an interval extension of the former.

Definition 5.1.4. An interval rate law $\widehat{\mathrm{R}} : \mathbb{IR}^n \to \mathbb{IR}$ is an *interval extension* of a rate law $R : \mathbb{R}^n \to \mathbb{R}$ if $\mathrm{R}(x) \in \widehat{\mathrm{R}}(X)$ for every interval X and real $x \in X$.

Definition 5.1.5. An interval rate law family $\widehat{\mathrm{R}} : \mathbb{IR}^m \to \mathbb{IR}^* \to \mathbb{IR}$ is an *interval extension* of a rate law family $\mathrm{R} : \mathbb{R}^m \to \mathbb{R}^* \to \mathbb{R}$ if $\widehat{\mathrm{R}}(\mathbf{X})$ is an interval extension of $\mathrm{R}(\mathbf{x})$ for every interval vector $\mathbf{X}$ and real vector $\mathbf{x} \in \mathbf{X}$.

We can now apply this to extend affinity networks to interval rate parameters.

Definition 5.1.6. An *uncertain affinity network* has the form

$$\widehat{\mathcal{A}} = \left\{ \boldsymbol{\gamma}^{(1)} \mathbin{@} \widehat{\mathrm{L}_1}, \ldots, \boldsymbol{\gamma}^{(n)} \mathbin{@} \widehat{\mathrm{L}_n} \right\} \in \widehat{\textsc{Aff}} \triangleq \mathcal{P}\Big(\textsc{Ordered-Pattern} \times \left[\mathbb{IR}^*_{\geq 0} \to \mathbb{IR}\right]\Big)$$

where each $\widehat{\mathrm{L}_i} : \mathbb{IR}^{m_i}_{\geq 0} \to \mathbb{IR}$ is an interval rate law.

Many examples of uncertain affinity networks take the form

$$\widehat{\mathcal{A}} = \left\{ \boldsymbol{\gamma}^{(1)} \mathbin{@} \mathrm{L}^{(1)}_{\widehat{\mathbf{k}}_i}, \ldots, \boldsymbol{\gamma}^{(n)} \mathbin{@} \mathrm{L}^{(n)}_{\widehat{\mathbf{k}}_i} \right\}$$

of an affinity network whose rate laws families $\mathrm{L}^{(i)}$ are each applied to a vector $\widehat{\mathbf{k}}_i = \left(K^{(i)}_1, \ldots, K^{(i)}_{m_i}\right)$ of interval uncertain parameters. These are interpreted as uncertain affinity networks in the sense of Definition 5.1.6 by taking an appropriate interval extension $\widehat{\mathrm{L}^{(i)}}$ of each rate law family $\mathrm{L}^{(i)}$.

We can also define when an uncertain affinity network is an interval extension of an affinity network.

Definition 5.1.7. An uncertain affinity network $\widehat{\mathcal{A}} = \left\{ \boldsymbol{\gamma}^{(1)} \mathbin{@} \widehat{\mathrm{L}_1}, \ldots, \boldsymbol{\gamma}^{(n)} \mathbin{@} \widehat{\mathrm{L}_n} \right\}$ is an *interval extension of* the affinity network $\mathcal{A} = \left\{ \boldsymbol{\gamma}^{(1)} \mathbin{@} \mathrm{L}_1, \ldots, \boldsymbol{\gamma}^{(n)} \mathbin{@} \mathrm{L}_n \right\}$ if $\widehat{\mathrm{L}_i}$ is an interval extension of L_i for all i. In this case we write $\mathcal{A} \in \widehat{\mathcal{A}}$.

We are now ready to define uncertain bond-calculus models.

Definition 5.1.8. An *uncertain bond-calculus model* $\left(\widehat{\Pi}_0, \widehat{\mathcal{A}}\right)$ consists of:

- an uncertain mixture $\widehat{\Pi}_0 = I_1 S_1 \parallel \ldots \parallel I_n S_n$;
- an uncertain affinity network $\widehat{\mathcal{A}} = \left\{ \boldsymbol{\gamma}^{(1)} \mathbin{@} \widehat{\mathrm{L}_1}, \ldots, \boldsymbol{\gamma}^{(n)} \mathbin{@} \widehat{\mathrm{L}_n} \right\}$.

Uncertain bond-calculus models then act as an over-approximation of a whole set of bond-calculus models that they soundly enclose.

Definition 5.1.9. A bond-calculus model $\left(\Pi_0, \mathcal{A}\right)$ *is contained in the uncertain bond-calculus model* $\left(\widehat{\Pi}_0, \widehat{\mathcal{A}}\right)$ if $\Pi_0 \in \widehat{\Pi}_0$ and $\widehat{\mathcal{A}}$ is an interval extension of $\mathcal{A}$. In this case we write $\left(\Pi_0, \mathcal{A}\right) \in \left(\widehat{\Pi}_0, \widehat{\mathcal{A}}\right)$.

Over the next few subsections we will discuss how we should interpret interval uncertainties in an uncertain bond-calculus model's initial mixture (Section 5.1.1) and its affinity network (Section 5.1.2), leading up to defining a formal semantics based on trajectories (Section 5.1.3) and an interval extension of the ODE extraction procedure for the bond-calculus (Section 5.1.4).

5.1.1 Uncertain Initial Mixtures

We now consider how we should interpret a bond-calculus model $(\widehat{\Pi}_0, \mathcal{A})$ with an uncertain initial mixture

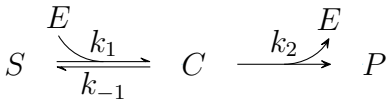
$$\widehat{\Pi}_0 \triangleq I_1 S_1 \parallel \ldots \parallel I_n S_n$$

which, for the moment, we assume is paired with a real affinity network $\mathcal{A}$. In this case the treatment of uncertainty is fairly straightforward since we may assume that each interval I_j corresponds to an unknown constant parameter $c_j \in I_j$. Hence, we may regard the uncertain mixture $\widehat{\Pi}_0$ as representing a parametric set of mixtures

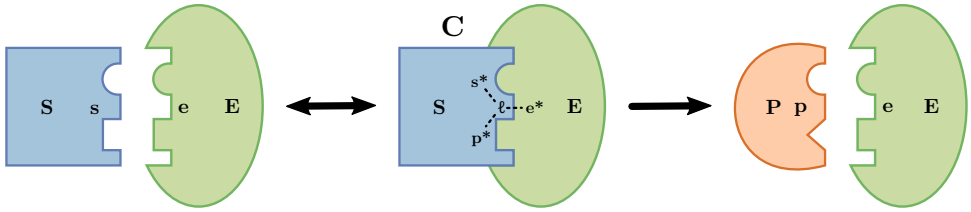
$$\widehat{\Pi}_0 \triangleq \left\{ \Pi_0(c_1, \ldots, c_n) \mid c_1 \in I_1, \ldots, c_n \in I_n \right\}$$

where $\Pi_0(c_1, \ldots, c_n) \triangleq c_1 S_1 \parallel \ldots \parallel c_n S_n$. This means that interval mixtures can be interpreted as a non-deterministic choice of initial mixture.

We now demonstrate the role of initial uncertainty to an example enzymatic reaction system.



(a) Reaction scheme



(b) The sites and locations in the enzyme binding process.

Figure 5.1: Mass Action Enzyme System

Example 5.1.10. Consider the bond-calculus model with species

$$S \triangleq s(\ell).(s^*@\ell.S + p^*@\ell.P) \qquad E \triangleq e(\ell).e^*@\ell.E \qquad P \triangleq p.\mathbf{0}$$

and affinity network

$$\mathcal{A}_{\text{MA}} \triangleq \left\{ s \parallel e \ @\ \text{MA}_{k_1},\ s^*|e^*\ @\ \text{MA}_{k_{-1}},\ p^*|e^*\ @\ \text{MA}_{k_2} \right\}.$$

based on the fixed rate parameters

$$k_1 \triangleq 1.0 \qquad k_{-1} \triangleq 0.1 \qquad k_2 \triangleq 0.5.$$

This models a classic mass action enzyme reaction system which is shown in Fig. 5.1 and consists of a substrate S and an enzyme E which dynamically bond to form a complex $C \triangleq (\nu\ell)(s^*@\ell.S|p^*@\ell.P)$ which degrades to form a product P.

If we start with an initial mixture with fixed real concentrations

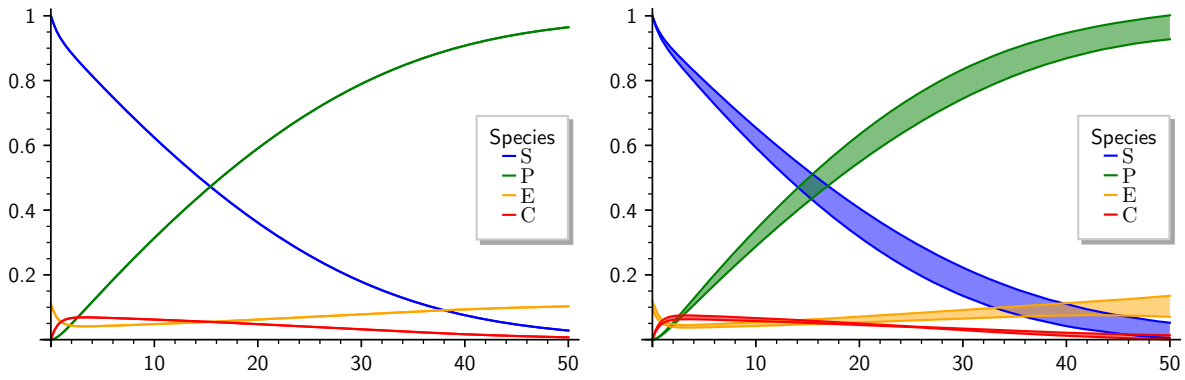
$$\Pi_0 \triangleq 0.1\,E \parallel 1.0\,S$$

the evolution of the system is as shown in Fig. 5.2a.

Suppose we are unsure about the exact initial concentration of enzyme. In this case we can model the system using an uncertain initial mixture,

$$\widehat{\Pi}_0^{\text{UI}} \triangleq [0.10, 0.12]\,E \parallel 1.0\,S$$

Even this small uncertainty in the initial conditions leads to a comparatively large uncertainty in the final state of the system as shown in Fig. 5.2b.



(a) Certain initial conditions.

(b) Uncertain initial conditions.

Figure 5.2: Enzyme simulation results with definite and uncertain initial conditions.

5.1.2 Uncertain Affinity Networks

We will now discuss the interpretation of interval uncertainties in affinity networks, focusing for the moment on the special case of uncertain affinity networks with interval rate parameters.

The interval uncertainties in these affinity networks are a little trickier to handle than those in initial mixtures, since they impact not just the initial state of the system, but its evolution at every subsequent time point. This gives us a choice as to how we interpret this uncertainty over time. The first choice is to treat these intervals as representing *time-invariant uncertainty*. In this case consider an uncertain affinity network of form

$$\widehat{\mathcal{A}} = \left\{ \boldsymbol{\gamma}^{(1)} \mathbin{@} \mathrm{L}^{(1)}_{\widehat{\mathbf{k}}}, \ldots, \boldsymbol{\gamma}^{(n)} \mathbin{@} \mathrm{L}^{(n)}_{\widehat{\mathbf{k}}} \right\}$$

where each rate law is derived from a rate law family $\mathrm{L}^{(i)}$ based on a vector of interval parameters $\widehat{\mathbf{k}} = (K_1, \ldots, K_m)$[1]. This can be interpreted as a parametric affinity network

$$\mathcal{A}(k_1, \ldots, k_n) = \left\{ \boldsymbol{\gamma}^{(1)} \mathbin{@} \mathrm{L}^{(1)}_{(k_1,\ldots,k_m)}, \ldots, \boldsymbol{\gamma}^{(n)} \mathbin{@} \mathrm{L}^{(n)}_{(k_1,\ldots,k_m)} \right\}$$

where each of these intervals parameters K_i corresponds to a fixed rate parameter which takes the same value $k_i \in K_i$ at every point during the evolution of the system. That is, every rate parameter of the system really has a single fixed value k_i, we just don't know its precise value and hence represent our uncertain knowledge of its value by the interval K_i.

Under this assumption, we can actually encode our uncertain affinity network $\mathcal{A}$ as a bond-calculus model $\left(\widehat{\Pi}_{\mathbf{k}}, \mathcal{A}_{\mathbf{k}}\right)$ where all of the uncertainty has been shifted to the initial

[1]For simplicity we assume w.l.o.g. that all rate laws depend on a single shared parameter vector $\widehat{\mathbf{k}}$.

mixture $\widehat{\Pi}_{\mathbf{k}} \triangleq \|_i K_i W_i$ in which the fresh species W_i are defined by $W_i \triangleq w_i \to W_i$, the affinity network $\mathcal{A}_{\mathbf{k}}$ is defined by

$$\mathcal{A}_{\mathbf{k}} \triangleq \left\{ \boldsymbol{\gamma}^{(i)} \parallel w_1 \parallel \ldots \parallel w_m \mathbin{@} \mathrm{L}_i : i \in \{1, \ldots, n\} \right\},$$

and the real rate laws $\mathrm{L}_i : \mathbb{R}^{n+p} \to \mathbb{R}$ are defined by

$$\mathrm{L}_i\Big([w_1], \ldots, [w_m], \left[\gamma_1^i\right], \ldots, \left[\gamma_p^i\right]\Big) \triangleq \Big(\mathrm{L}^{(i)}([w_1], \ldots, [w_m])\Big)\Big(\left[\gamma_1^i\right], \ldots, \left[\gamma_p^i\right]\Big).$$

In general, however, we have no reason to believe that the uncertain parameters in an affinity network have a constant value throughout the evolution of a system. In many situations, uncertain rate parameters in a reaction correspond to the impact of reactants which are not included in our model due either to their complexity or our uncertain knowledge of their exact structure or state. In such cases we must assume that these rates may vary over time since we have no reason to think that the reactants to which they correspond have a constant concentration. This meant that we must treat the interval rate parameters as time-dependant unknown parameters $k_i(t) \in K_i$ in a time-varying parametric affinity network

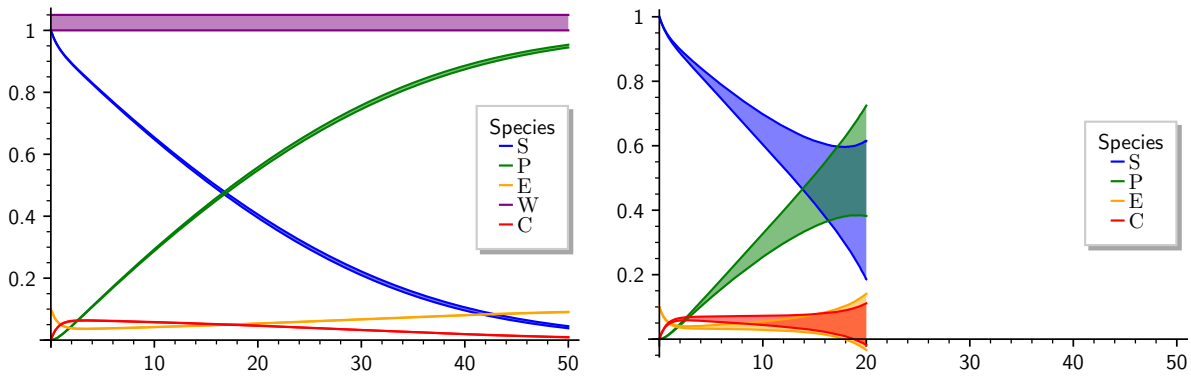
$$\mathcal{A}(t) = \left\{ \boldsymbol{\gamma}^{(i)} \mathbin{@} \mathrm{L}^{(i)}_{\mathbf{k}(t)} \;\middle|\; i \in \{1, \ldots, n\} \right\}$$

where $\mathbf{k}(t) = (k_1(t), \ldots, k_m(t))$. Unlike the time-variant case, the uncertainties in such affinity networks cannot be encoded as uncertain initial conditions. If, however, we apply the bond-calculus semantics to the parametric model $(\Pi_0, \mathcal{A}(t))$, we get an IVP

$$\frac{\mathrm{d}\Pi}{\mathrm{d}t} = f(\Pi, \mathbf{k}(t)) \triangleq \frac{\mathrm{d}(\Pi, \mathcal{A}(t))}{\mathrm{d}t}; \quad \Pi(0) \in \Pi_0 \tag{5.1}$$

with an uncertain parameter vector function $\mathbf{k} : \mathbb{R}_{\geq 0} \to \mathbb{R}^m$ matching the form of Definition 2.4.3. The case of constant $\mathbf{k}(t) \equiv \mathbf{k}_0 \in \mathbb{R}^m$ in fact coincides with our previous discussion of time-invariant uncertain affinity networks. Specific methods for directly analysing this class of IVPs (Definition 2.4.2) were also introduced by Lin and Stadtherr [241]. The time-varying case is in general much harder since $\mathbf{k}(t)$ could potentially make arbitrary discontinuous jumps from each time point to the next, however, if we make the (sometimes reasonable) assumption of continuous $\mathbf{k} \in \mathcal{C}([-\infty, \infty])$, this class of systems also yields a unique solution and is supported in Flow* via an extension of the methods of Lin and Stadtherr to time-independent uncertainty [109].

Example 5.1.11. Suppose that in the enzyme system of Example 5.1.10, in the place of uncertain initial conditions, we have interval uncertain knowledge of the rate parameter $k_1 \in [1.00, 1.05]$.



(a) Time-invariant uncertainty.

(b) Time-varying enzyme flowpipe.

Figure 5.3: Enzyme systems with time-invariant and time-varying uncertainty.

If we want to interpret k_1 as a time-invariant rate parameter, it is possible to transform this into a system with the uncertain initial conditions

$$\widehat{\Pi}_0^{\mathrm{TI}} \triangleq 0.1\, E \parallel 1.0\, S \parallel [1.00, 1.05]\, W$$

with additional species

$$W \triangleq w.W$$

and the affinity network,

$$\mathcal{A}_{\mathrm{MA}} \triangleq \left\{ w \parallel s \parallel e \mathbin{@} \mathrm{MA}_1,\ s^*|e^* \mathbin{@} \mathrm{MA}_{k_{-1}},\ p^*|e^* \mathbin{@} \mathrm{MA}_{k_2},\ p \mathbin{@} \mathrm{MA}_{k_3} \right\}.$$

The potential behaviours of this system are shown in Fig. 5.3a. In this case the uncertainty in the rate parameter leads to a reasonably small uncertainty in the final state of the system.

If instead we assume that the unknown rate parameter k_1 (and hence the affinity network) varies continuously over time and use Flow* to handle it as a time-varying parameter, this leads to much greater uncertainty in the system evolution and for the flowpipe for the system to eventually blow up as shown in Fig. 5.3b.

5.1.3 Dynamics of Uncertain Models

We are now ready to formally define the possible behaviours of an uncertain bond-calculus models based on bond-model trajectories which generalize the discussions in the previous two sections. Bond-model trajectories couple a trajectory in the space of mixtures representing the evolving state of the system with a trajectory in the space of affinity networks representing a time-varying affinity network.

Definition 5.1.12. A function $\Gamma : \mathbb{R}_{\geq 0} \to \textsc{Mix} \times \textsc{Aff}$ is a *bond-model trajectory* iff Γ is the product $\Gamma = \Pi \times \mathcal{A}$ (defined such that $(\Pi \times \mathcal{A})(t) = (\Pi(t), \mathcal{A}(t))$ for all $t \in \mathbb{R}_{\geq 0}$) of some continuously time-varying affinity network $\mathcal{A} : \mathbb{R}_{\geq 0} \to \textsc{Aff}$ and some trajectory $\Pi : \mathbb{R}_{\geq 0} \to \textsc{Mix}$ of the vector field

$$\frac{\mathrm{d}\Pi}{\mathrm{d}t} = \frac{\mathrm{d}(\Pi, \mathcal{A}(t))}{\mathrm{d}t}. \tag{5.2}$$

We then define when a bond-model trajectory is consistent with a given interval bond-model based on the interval uncertainties contained in both its initial mixture and its affinity network.

Definition 5.1.13 (Uncertain Bond-Model Trajectory)**.** Given an uncertain bond-calculus model $\mathcal{M} = \left(\widehat{\Pi}_0, \widehat{\mathcal{A}}\right)$, a bond-model trajectory $\Gamma = \Pi \times \mathcal{A}$ is consistent with $\mathcal{M}$ if $\Pi(0) \in \widehat{\Pi}_0$ and if $\mathcal{A}(t) \in \widehat{\mathcal{A}}$ for all $t \in \mathbb{R}_{\geq 0}$.

We note that this definition is slightly more flexible than the discussion in the previous section since it encompasses arbitrary interval uncertainties in the affinity network, not just those arising from a fixed set of interval rate parameters.

5.1.4 Interval ODE Extraction

Whilst bond-model trajectories define the possible behaviours of an uncertain bond-calculus model, in order to apply analysis tools such as Flow*, we need to represent the system in a closed form as an interval Initial Value Problem.

This is done by extending the semantics of the bond-calculus from Chapter 3 to a (symbolic) interval semantics, with intervals replacing every occurrence of real numbers and interval arithmetic replacing real arithmetic. Applying the ODE extraction procedure will still produce a symbolic initial value problem, however, intervals may now occur as both the initial conditions and in the right hand side of the resulting systems of ODEs. Whilst these interval systems are outside of the range of normal numerical solvers, they match the input format of Flow* [110] which is able to handle uncertain initial conditions, whilst treating interval uncertainties on the right hand side of ODEs as time-varying uncertain inputs.

The following result shows us that a flowpipe for the interval vector field derived from an uncertain bond-calculus model gives an interval extension of the bond-model trajectories consistent with the model.

Proposition 5.1.14. *Consider an uncertain bond-calculus model* $\left(\widehat{\Pi}_0, \widehat{\mathcal{A}}\right)$. *Given any flowpipe* Π_f *for the symbolic Interval Initial Value Problem*

$$\frac{\mathrm{d}\widetilde{\Pi}}{\mathrm{d}t} = \frac{\mathrm{d}\left(\widetilde{\Pi}, \widehat{A}\right)}{\mathrm{d}t}; \quad \widetilde{\Pi}_0 = \widehat{\Pi}_0 \tag{5.3}$$

and any bond-model trajectory $\Pi \times \mathcal{A}$ *consistent with* $\left(\widehat{\Pi}, \widehat{\mathcal{A}}\right)$, *we have that* Π_f *is an interval extension for* Π.

Proof. (Sketch) We first observe that

$$\frac{\mathrm{d}(\Pi, \mathcal{A}(t))}{\mathrm{d}t} \in \frac{\mathrm{d}\left(\Pi, \widehat{A}\right)}{\mathrm{d}t}$$

for any time point $t \in \mathbb{R}_{\geq 0}$. This holds since, by construction, the interval semantics of bond-calculus models forms an interval enclosure of its real semantics. But then a Flow* flowpipe for the interval system Eq. (5.3) is guaranteed to enclose any trajectory of the vector field Eq. (5.2), giving the result. □

5.2 $\mathcal{LBUC}$ Logic: Syntax and Semantics

We are now ready to introduce the $\mathcal{LBUC}$ logic to express the temporal behaviour of uncertain bond-calculus models in uncertain bond-calculus contexts.

The syntax of $\mathcal{LBUC}$ formulae is defined as follows.

Definition 5.2.1. A $\mathcal{LBUC}$ *formula* is defined according to the following grammar:

$$\begin{aligned}
\varphi, \psi \ &::= \rho && \textit{(atomic propositions)} \\
&\mid \varphi \wedge \psi \ \mid \varphi \vee \psi \ \mid \varphi \Rightarrow \psi \ \mid \neg\varphi && \textit{(logical operators)} \\
&\mid \varphi \, \mathcal{F}_I \, \psi \ \mid \varphi \, \mathcal{G}_I \, \psi \ \mid \varphi \, \mathcal{U}_I \, \psi && \textit{(temporal operators)} \\
&\mid \mathcal{C} \triangleright \varphi && \textit{(context operator)},
\end{aligned}$$

where:

- $I = [a, b]$ is a time interval;
- a context $\mathcal{C} \triangleq (\Pi, \mathcal{A})$ consists of an uncertain bond-calculus model.

As in STL $\mathcal{LBUC}$ formulae include expressions $\rho \triangleq f(\mathbf{x}) > 0$ over system variables as atomic propositions, along with the standard logical operators, and MITL temporal operators. To these we add the context operator, $\mathcal{C} \triangleright \varphi$, which states that the property

φ should hold immediately after the system is placed within the uncertain context $\mathcal{C}$. In the case that $\mathcal{C}$ consists of a fully defined bond-calculus model $\mathcal{C} = C$, this corresponds directly to the context operator of $\mathcal{LBC}$, however, this also includes non-trivial uncertain contexts,

$$\mathcal{C} \triangleq \Big([a_1, b_1]\, X_1 \parallel \ldots \parallel [a_n, b_n]\, X_n,\ \mathcal{A}([c_1, d_1], \ldots, [c_m, d_m]) \Big)$$

which specify an interval box of uncertain initial conditions $x_0^{(i)} \in [a_i, b_i]$ and whose affinity networks $\mathcal{A}$ may include intervals denoting uncertain reaction rates $r_i \in [c_i, d_i]$.

We now define the semantics of $\mathcal{LBUC}$ based on the relations $\Pi \times \mathcal{A} \models_t \varphi$ and $\mathcal{M} \models_t \varphi$ which define when a logical property φ is satisfied by a bond-model trajectory or an overall (uncertain) bond-calculus model $\mathcal{M}$ respectively.

Definition 5.2.2 ($\mathcal{LBUC}$ semantics)**.** The semantics of $\mathcal{LBUC}$ is given by the $\models_t$ relation defined over bond-model trajectories at a given time point t by

$$\begin{array}{lll}
\Pi \times \mathcal{A} \models_t \rho & \text{iff} & \rho(\Pi(t)) > 0 \\
\Pi \times \mathcal{A} \models_t \varphi \wedge \psi & \text{iff} & \Pi \times \mathcal{A} \models_t \varphi \text{ and } \Pi \times \mathcal{A} \models_t \psi \\
\Pi \times \mathcal{A} \models_t \neg\varphi & \text{iff} & \Pi \times \mathcal{A} \not\models_t \varphi \\
\Pi \times \mathcal{A} \models_t \varphi\, \mathcal{U}_J\, \psi & \text{iff} & \begin{cases} \text{for some } t' \in t + J, \\ \Big(\Pi \times \mathcal{A} \models_{t'} \psi \text{ and for all } t'' \in [t, t'], \Pi \times \mathcal{A} \models_{t''} \varphi\Big) \end{cases} \\
\Pi \times \mathcal{A} \models_t \widehat{\mathcal{C}} \triangleright \varphi & \text{iff} & (\Pi \times \mathcal{A})(t) \parallel \widehat{\mathcal{C}} \models_0 \varphi
\end{array}$$

and for an overall uncertain bond-calculus model by,

$$\Big(\widehat{\Pi}, \widehat{\mathcal{A}}\Big) \models_t \varphi \qquad \text{iff} \qquad \Pi \times \mathcal{A} \models_t \varphi \text{ for every bond-model trajectory of } \Big(\widehat{\Pi}, \widehat{\mathcal{A}}\Big).$$

The semantics for the other logical and temporal operators are defined in the standard way based on logical and temporal equivalences.

Remark 5.2.3. We note that in the case of uncertain contexts, we do not have that

$$\Big(\widehat{\Pi}, \widehat{\mathcal{A}}\Big) \models_t \neg\varphi \quad \text{iff} \quad \Big(\widehat{\Pi}, \widehat{\mathcal{A}}\Big) \not\models_t \varphi$$

as was the case for STL or $\mathcal{LBC}$, since we implicitly quantify over uncertain models. Instead, we have

$$\Big(\widehat{\Pi}, \widehat{\mathcal{A}}\Big) \not\models_t \varphi \quad \text{iff} \quad \begin{cases} \text{for some bond model-trajectory } \Pi \times \mathcal{A} \\ \text{consistent with } \Big(\widehat{\Pi}, \widehat{\mathcal{A}}\Big),\ (\Pi, \mathcal{A}) \models_t \neg\varphi. \end{cases}$$

That is, if the property φ is refuted by the solution Π of the system under some possible time-varying affinity network $\mathcal{A}$.

Remark 5.2.4. We can consider interval bond-calculus models as a special case of uncertain contexts since

$$\left(\widehat{\Pi}, \widehat{\mathcal{A}}\right) \models_t \varphi \quad \text{iff} \quad \mathbf{0} \models_0 \left(\widehat{\Pi}, \widehat{\mathcal{A}}\right) \triangleright \mathcal{G}_t \, \varphi \quad \text{iff} \quad \mathbf{0} \models_0 \left(\widehat{\Pi}, \widehat{\mathcal{A}}\right) \triangleright \mathcal{F}_t \, \varphi$$

where we employ temporal operators with the singleton time interval $t = [t, t]$. This means that all logical properties involving uncertain initial conditions can be regarded as contextual properties, although directly handling uncertain initial conditions may still be useful to perform direct simulation and visualization. However, uncertain context operators are strictly more powerful than uncertain initial models since they can introduce new contexts into the system at arbitrary points in its subsequent evolution and, through alternation with logical and temporal operators, make it possible to express *contingent context introductions*.

5.3 Expressing Temporal Behaviour Under Uncertainty

In this section we will consider the expressive power of $\mathcal{LBUC}$, give examples of how it relates to biological modelling, and discuss how it relates to existing modelling formalisms.

The $\mathcal{LBUC}$ logic brings together two elements: a logic for specifying the temporal behaviour of models and a language for defining models with uncertain states. These are brought together via the context operator $\triangleright$, which introduces new uncertainty into the model at a given instant in time.

The temporal component (which corresponds to STL) already suffices to express properties on the ordering of events involving the state of a system such as Example 5.1.10, for example,

$$\mathcal{F}_{[0,5]}([S] > 5 \wedge \mathcal{G}_{[0,3]}([P] < 2))$$

which states that, at some point within the first 5 time units, the substrate species S will have a concentration greater than 5 and the concentration of the product P will be less than 2 for the next 3 time units.

In $\mathcal{LBC}$, this was extended by allowing events — the introduction of a fixed process via a context operator $C \triangleright \varphi$ — to be nested within a temporal formula. Whilst this might be compared to modelling formalisms with *discrete events*, when combined with a temporal logic, this makes it possible to express hypothetical biological experiments on a model, which mix observations of its temporal behaviour over time and probes which perturb the system by introducing new reactants. For example, this permits properties such as

$$\mathcal{F}_{[0,5]}([S] > 5 \wedge 0.5I \triangleright \mathcal{G}_{[0,3]}([P] < 0.1))$$

which states that there is some time point within the first 5 time units, for which S has a concentration greater than 5 at which *if* we introduce a 0.5 concentration of an inhibitor species I, *then* the concentration of P will be less than 0.1 for the next 3 time units.

In $\mathcal{LBUC}$ this is further extended with contexts involving both the mixture and its affinity network and with uncertainty which may be introduced either instantaneously or gradually over time. We will now see how, when embedded within temporal logic properties, this enables the logic to express a diverse range of contingent behaviours, and opens many new possibilities for biological modelling.

5.3.1 Robustness Under Perturbation

Robustness of behaviour under environmental perturbations is considered a key organizing principle of biochemical networks, which frequently operate in noisy environments and need to continue to function under a variety of genetic mutations [222]. $\mathcal{LBC}$ is already well-suited to expressing the robustness of properties to the timing of events (consisting of context introductions) and has been employed to study robustness of post-translational oscillators in [22]. Uncertain contexts translate naturally to a more general class of *contextual robustness* properties; that is, the uncertain contextual property $\widehat{\mathcal{C}} \triangleright \varphi$ states that the property φ holds robustly under the class of contexts in $\widehat{\mathcal{C}}$. In the simplest case, an uncertain context representing the initial state of a system allows us to state that a property of a given biological system holds robustly under reactant concentrations and time-varying rate parameters. More generally, the mixture of uncertain contexts with temporal operators also makes it possible to specify robustness under both the timing and the state of the context by which a system is perturbed; this represents robustness of a system to interactions with an uncertain external environment.

5.3.2 Existence of Coreactants

Whilst the role of the context operator as a universal quantifier suggests that it is mainly directed towards verifying properties hold over a range of uncertain conditions, it can also be used to verify the existence of the conditions required for a desired property to hold. To this end we may define an *existential context operator* by

$$\widehat{\mathcal{C}} \triangleright_{\exists} \varphi \equiv \neg\big(\widehat{\mathcal{C}} \triangleright \neg\varphi\big).$$

This asserts that there exists some context consistent with $\widehat{\mathcal{C}}$ which can make the system satisfy φ. Such properties are very interesting from a biochemical perspective as the search

for coreactants or coreactant concentrations which move a system into a given behaviour regime is central to the development of pharmaceutical interventions.

5.3.3 Experimental Protocols

The mixture of contextual and temporal operators offers a way to model experimental protocols in which one modifies a system at given points in time and observes the response of the system to this stimulus. We may, for example, model simple protocols such as,

$$\psi \triangleq \mathcal{G}_{[t_1-\delta,t_1+\delta]}(\Phi_1 \rhd \mathcal{G}_{[t_2-\delta,t_2+\delta]}(\Phi_2 \rhd \ldots \rhd \mathcal{G}_{[t_n-\delta,t_n+\delta]}(\Phi_n \rhd \mathcal{F}_{[0,T]}\,\varphi))$$

which states that after modifying the system by introducing the context of reactants Φ_1 after t_1 time units, Φ_2 after another t_2 time units, etc, then subsequently conditions described by the property φ will occur within at most T time units. Furthermore, this property requires that the result holds robustly under variations in the timing of reactant introductions (up to δ time units) and under any uncertainty in the initial conditions or in any of the reactant contexts Φ_i.

These experiments are somewhat similar to the use of discrete events in frameworks such as discrete event simulation and hybrid automata. However, $\mathcal{LBUC}$ allows us to directly specify experimental protocols and expected sequences of observations without entangling them with the model of the intrinsic behaviour of the system. Additionally, $\mathcal{LBUC}$ makes it possible to specify more complex protocols in which how we probe the system is dependent upon its previous behaviour, whilst also handling uncertainty as to the timing of events and the state of the system.

5.3.4 Differential Species Introduction

As well as directly introducing a species into a system at an instant in time, in $\mathcal{LBUC}$ it is also possible to define a *differential context operator* $\Pi \rhd_\partial \varphi$ which gradually introduces each species S_i in the process $\Pi \triangleq \|_{j=1}^{n} r_i S_i$ into the system at a (potentially uncertain) rate r_i. This may be defined as a derived modality based on the context operator

$$\Pi \rhd_\partial \varphi \quad \equiv \quad \mathcal{C}_\Pi \rhd \varphi$$

where the context $\mathcal{C}_\Pi$ is defined by

$$\mathcal{C}_\Pi \triangleq \Big\|_{j=1}^{n} (1 \cdot G_S, \mathcal{A}_{S,r})$$

given the species $G_S \triangleq g_S \rightarrow (G_S|S)$ and affinity networks $\mathcal{A}_{S,r} \triangleq \{\ g_S \mathbin{@} r\ \}$.

This makes it possible to model experimental protocols in which a new species is introduced not instantaneously but gradually at a given rate. This also corresponds to the central role of compartments (such as cell membranes) in biological systems, which limit the rate of flow of reactants in between well-mixed reaction environments.

The use of intervals of uncertainty in these introduction rates r_i plays different roles in behavioural specification and modelling. When we are interested in probing the behaviour of a model in an experimental protocol, this allows us to verify that the property we are interested in is robust to the exact rate of introduction of a new reactant. When instead we are interested in modelling the influx of reactants from the wider environment, this allows us to account for our uncertain knowledge of the rate of flow into the system. Similarly to normal context operators, the uncertain rates in a differential context may be interpreted as either time-invariant or time-varying. These respectively cover the case in which reactants flow into the system at a fixed but unknown rate (due to, say, imprecision in measurements) and the case that reactants are introduced at a time-varying rate (depending on, say, the state of the membrane, or fluctuations external to the system).

5.3.5 Reconfigurable Networks

Whilst most reaction based modelling frameworks for biological systems assume that species react according to a static set of reaction rules, bond-calculus contexts offer the possibility of modelling the introduction of new reaction rules over time. This can be modelled as an affinity network context $\mathcal{A} \triangleright \varphi$ which introduces an affinity network $\mathcal{A}$ which is instantaneously imposed upon the system.

This corresponds to *reconfigurable biochemical networks* in which certain events change the network structure over time. Reconfigurable systems are also of interest in synthetic biology, as they can model multi-functional biochemical circuits which can switch between different roles based on chemical inputs [183] or other input types such as light [176]. We are also able to encode the removal of reaction rules, by adding reaction rules with negative rates (although not when these rates are uncertain).

Chapter 6

Verified monitoring with Flow*

Our focus thus far has been on modelling languages for biological systems (such as the bond-calculus) and specification logics describing their behaviour under uncertainty (such as STL and $\mathcal{LBUC}$). In order to practically apply such specification languages to analysing biological systems, we require automated methods to verify whether a model conforms to a given specification. In particular, these verification methods must be able to handle uncertainties in the system arising from uncertain initial conditions and parameters as well as environmental uncertainties introduced through uncertain contexts. Temporal logic verification for continuous systems has frequently been achieved via lightweight formal methods such as the Maler and Nickovic's signal monitoring algorithm for STL [253] and Banks and Stark's signal-based monitoring algorithm for $\mathcal{LBC}$ [20, 23], which use numerical traces from a model or a running system to compute Boolean signals for propositions in a bottom up manner.

In this chapter we propose to combine the signal-based monitoring approach with Flow* verified integration to acheive a sound model checking procedure for the full range of STL and $\mathcal{LBUC}$ properties given a wide range of different uncertainties. To achieve soundness of monitoring, we make use of three-valued signals to record our uncertainty in the Boolean truth values of propositions over time. Similarly to the role of intervals in numerical computations, these utilize three-valued logic as a way of quantifying our monitoring results over each of the uncertainties in a given model as well as any additional uncertainties introduced throughout the verification procedure. We introduce an interval-based representation for three-valued signals and see how three-valued signals may be computed compositionally for each STL and $\mathcal{LBUC}$ operator, extending Maler and Nickovic's Boolean signal monitoring algorithm. We also introduce novel symbolic techniques for monitoring atomic propostions over Flow* flowpipes, allowing us to more precisely

monitor complex atomic propositions, whilst we introduce an adaptive monitoring approach to improve the efficiency of monitoring by reserving these symbolic techniques to the most difficult segments of the time domain. We will see that these methods are able to efficiently verify interesting properties of biological systems modelled in the bond-calculus whilst our adaptive monitoring algorithm acheives an effective tradeoff between monitoring precision and efficiency.

As well as allowing us to handle uncertainties, the use of exact formal methods also gives us a way to reason about the long-term behaviour of a system in a manner not possible with methods based on finite numerical traces. To this end, we introduce a method for verifying unbounded temporal operators, which employs invariants of a system's dynamics to extend three-valued signals over unbounded time domains. This offers a practical way to extend our verified monitoring method to tackle many interesting classes of unbounded time properties.

In Section 6.1.1 we introduce three-valued signals and specify rules to combine these to derive signals for complex propositions. We then develop an efficient algorithm for monitoring signals of atomic propositions based over Flow* flowpipes in Sections 6.1.2 and 6.1.3. In Section 6.3 we show that in some cases this may be extended to verification of unbounded temporal operators through the use of invariant sets. Then in Section 6.4 we examine the performance of our monitoring algorithm when applied to a 9-dimensional genetic oscillator whilst in Section 6.5 we demonstrate the application of our verification techniques to a model of Predator Prey Role-Reversal in a marine ecosystem.

Some of the results from this chapter have been published as part of the paper [360].

6.1 Three-Valued Monitoring over Flow* flowpipes

In this section we present an effective verified monitoring algorithm for Signal Temporal Logic by adaptively applying three-valued signal monitoring to Flow* flowpipes.

6.1.1 Three-Valued Signals

Our verified monitoring algorithm is based on *three-valued signals*:

Definition 6.1.1. A *three-valued signal* is a function $s : \mathbb{R}_{\geq 0} \to \mathbb{T}$.

These extend the Boolean signals $s : \mathbb{R}_{\geq 0} \to \mathbb{B}$ used in numerical STL monitoring algorithms to track the validity of the answer at each time point $t \in [0, \infty)$, to use the

third logical value *Unknown* (**U**) of Three-Valued Logic, to indicate when we can neither verify nor refute the proposition. A three-valued signal is consistent with the (Boolean) truth values of a proposition φ if it soundly under-approximates the time-regions on which φ is true (**T**) and false (**F**):

Definition 6.1.2. Given a proposition φ and a three-valued signal s, we say s *is a signal for* φ (over the trajectories of a given system) if at every time t,

$$\begin{aligned} s(t) = \mathbf{T} \quad &\Longrightarrow \quad \mathbf{x} \models_t \varphi \quad \text{for every trajectory } \mathbf{x} \\ s(t) = \mathbf{F} \quad &\Longrightarrow \quad \mathbf{x} \models_t \neg\varphi \quad \text{for every trajectory } \mathbf{x} \,. \end{aligned}$$

This definition allows a single proposition φ to be approximated by many different signals to differing degrees of precision. Indeed, the signal which is unknown everywhere is a valid (but uninformative) signal for every proposition.

For concrete computation we work with those three-valued signals s that can be represented by a finite set of nonempty disjoint intervals $I_j = [l_{I_j}, u_{I_j}]$ and Boolean values $s_j \in \mathbb{B}$, with j ranging over some index set Γ. These values determine signal s on those intervals, with **U** assumed elsewhere, and we write $s = (I_j, s_j)_j$. For convenience we also admit some improper representations including empty intervals, values $s_j = \mathbf{U}$, and intervals with overlapping endpoints (but consistent logical values); these may all be rewritten as proper representations.

Given propositions φ and ψ with $s = (I_j, s_j)_j$ a signal for φ and $w = (I_j, w_j)_j$ a signal for ψ, we have the following constructions:

$\neg\varphi$ has a signal given by $\neg s = (I_j, \neg s_j)_j$

$\varphi \wedge \psi$ has a signal given by $s \wedge w = (I_j, s_j \wedge w_j)_j$ where we may assume, without loss of generality, that s and w are represented using a common set of intervals I_j; this is always possible by taking a common refinement of the representations of s and w respectively.

$\mathcal{F}_{[a,b]}\,\varphi$ has a signal given by $\mathcal{F}_{[a,b]}\, s = (K_j \cap [0,\infty), s_j)_j$ where

$$K_j = \begin{cases} I_j - [a,b] & \text{if } s_j = \mathbf{T} \\ I_j \dot{-} [a,b] & \text{if } s_j = \mathbf{F} \end{cases}$$

so that the true intervals I are shifted back and outwards by interval subtraction in the same way as in the Boolean interval representation of STL [253], whilst the false intervals I are shifted back and inwards by inner subtraction.

Whilst the pointwise semantics of the until operator could be extended directly to three-valued logic, it is somewhat trickier to define a closed representation, as required for interval-based verified monitoring. In the case of Boolean signals, the until signal $s \,\mathcal{U}_J\, w$ is usually computed by subdividing s into *disjoint unitary signals* s_j (that is, signals which are indicator functions $s_j(t) = \chi_{I_j}(t) = (\mathbf{T}$ if $t \in I_j$ otherwise $\mathbf{F})$ of pairwise disjoint intervals) [253]. For three-valued signals we will follow a similar approach, however we need an appropriate three-valued generalisation of unitary signals. To this end we define a *connected signal*.

Definition 6.1.3. We say a three-valued signal s is *connected* if for every interval $[a, b]$ we have that,

$$(s(a) \wedge s(b)) \leq s(t) \qquad \text{for all } t \in [a, b].$$

under the ordering $\mathbf{F} \lesssim \mathbf{U} \lesssim \mathbf{T}$.

This means both the regions of the signal which are definitely true ($\mathbf{T}$) or possibly true ($\mathbf{T}$ or $\mathbf{F}$) are both connected sets. We can also see that a connected signal can be represented as a three-valued indicator signal.

Proposition 6.1.4. *A three-valued signal s is connected iff there exist intervals $J \subseteq I$ such that s is equal to the* three-valued indicator signal*,*

$$\chi_{J,I}(t) \triangleq \begin{cases} \mathbf{T} & \text{if } t \in J \\ \mathbf{U} & \text{if } t \in I \setminus J \\ \mathbf{F} & \text{if } t \notin I \end{cases}$$

Proof. First we note that an indicator signal $\chi_{J,I}$ is clearly connected. Conversely, given a connected signal s, take

$$J = \left[\; l_J \triangleq \inf_{s(t)=\mathbf{T}} t, \qquad u_J \triangleq \sup_{s(t)=\mathbf{T}} t \; \right] \quad \text{and}$$
$$I = \left[\; l_I \triangleq \inf_{s(t)\in\{\mathbf{T},\mathbf{U}\}} t, \quad u_I \triangleq \sup_{s(t)\in\{\mathbf{T},\mathbf{U}\}} t \; \right] \supseteq J$$

(for simplicity we assume that the properties defining these intervals hold at their endpoints; the other cases result in open/half-open intervals and are analogous). Then we see that $s = \chi_{J,I}$: for any $t \in J$, $\mathbf{T} \geq s(t) \geq \min(s(l_J), s(u_J)) = \mathbf{T}$ by the connectedness of s, whilst for any $t \notin I$, supposing $s(t) \neq \mathbf{F}$ and w.l.o.g. that $t > u_J$, since s is connected, $s([u_J, t]) = \{\mathbf{T}, \mathbf{U}\}$ contrary to the maximality of u_J, and finally, for any $t \in I \setminus J$, we

first see by connectedness of s that $s(t) \in \{\mathbf{T}, \mathbf{U}\}$, and then if we suppose $s(t) = \mathbf{T}$ we get a contradiction, similarly to the previous case, leading us to conclude that $s(t) = \mathbf{U}$. □

We note that it is straightforward to compute a signal for $\varphi \,\mathcal{U}_K\, \psi$ on connected signals.

Proposition 6.1.5. *If s and w are signals for φ and ψ respectively, and s is connected then*

$$s \,\mathcal{U}_K\, w \triangleq s \wedge \mathcal{F}_K(s \wedge w)$$

is a signal for $\varphi \,\mathcal{U}_K\, \psi$.

Proof. Suppose $(s \,\mathcal{U}_K\, w)(t) = \mathbf{T}$. Then $s(t) = \mathbf{T}$ and for some $t' \in t + K$, $s(t') = \mathbf{T}$ and $w(t') = \mathbf{T}$. But then since s is connected, $s(t') = \mathbf{T}$ for all $t'' \in [t, t']$, showing that $\mathbf{x} \models_t \varphi \,\mathcal{U}_K\, \psi$.

Suppose $(s \,\mathcal{U}_K\, w)(t) = \mathbf{F}$. Then either $s(t) = \mathbf{F}$ in which case $\mathbf{x} \models_t \neg(\varphi \,\mathcal{U}_K\, \psi)$, or for all $t' \in t + K$, $s(t) = \mathbf{F}$ or $w(t) = \mathbf{F}$, in which case again $\mathbf{x} \not\models_t \varphi \,\mathcal{U}_K\, \psi$. □

We next decompose a three-valued signal into connected signals.

Proposition 6.1.6. *Given a three-valued signal s and disjoint intervals I_j such that $s^{-1}(\{\mathbf{T}, \mathbf{U}\}) = \biguplus_j I_j$, we have a decomposition $s = \bigvee_j \bigvee_k s_{j,k}$ of s into the connected components:*

- $s_{j,0} = \chi_{\varnothing, I_j}$ *whenever* $I_j \cap s^{-1}(\{\mathbf{T}\}) = \varnothing$;
- $s_{j,k} = \chi_{J_{j,k}, I_j}$ *given intervals* $J_{j,k}$ *such that* $I_j \cap s^{-1}(\{\mathbf{T}\}) = \biguplus_k J_{j,k}$.

Proof. Consider any time point t.

If $s(t) = \mathbf{F}$ then we see that $t \notin s^{-1}(\{\mathbf{T}, \mathbf{F}\})$ and hence $\bigvee_j \bigvee_k s_{j,k}(t) = \mathbf{F} = s(t)$.

If $s(t) = \mathbf{T}$, then we see that $t \in J_{j,k} \subseteq I_j$ for some j, k. But then

$$\mathbf{T} \geq \bigvee_l \bigvee_m s_{l,m}(t) \geq s_{j,k}(t) = \mathbf{T},$$

showing that $\bigvee_j \bigvee_k s_{j,k}(t) = \mathbf{T} = s(t)$.

If $s(t) = \mathbf{U}$ then we have $t \in I_j$ for some unique j and so

$$\bigvee_l \bigvee_m s_{l,m}(t) = \bigvee_m s_{j,m}(t).$$

But then we note $s_{j,0}(t) = \chi_{\varnothing, I_j}(t) = \mathbf{U}$ and for $m > 0$, $s_{j,m}(t) = \chi_{J_{j,k}, I_j}(t) = \mathbf{U}$, showing that the RHS is $\mathbf{U}$ also. □

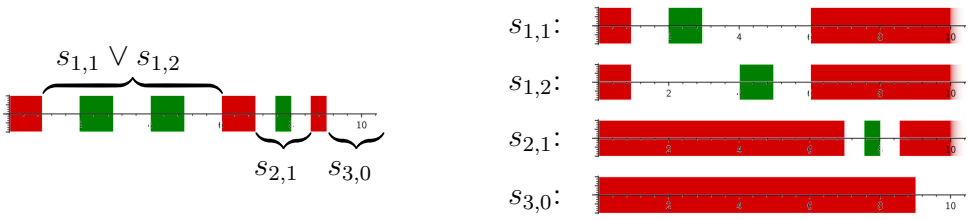


Figure 6.1: The decomposition of s into components $s \equiv (s_{1,1} \vee s_{1,2}) \vee s_{2,1} \vee s_{3,0}$.

Example 6.1.7. Given the three-valued signal

$$s = (([0,1], \mathbf{F}), ([2,3], \mathbf{T}), ([4,5], \mathbf{T}), ([6,7], \mathbf{F}), ([7.5,8], \mathbf{T}), ([8.5,9], \mathbf{F}))$$

we have the decomposition (Fig. 6.1)

$$s = (s_{1,1} \vee s_{1,2}) \vee s_{2,1} \vee s_{3,0} = (\chi_{[2,3],\,(1,6)} \vee \chi_{[4,5],\,(1,6)}) \vee \chi_{[7.5,8],\,(7,8.5)} \vee \chi_{\varnothing,\,(9,\infty)}.$$

We now use this decomposition to construct a signal for $\varphi \,\mathcal{U}_K\, \psi$:

Proposition 6.1.8. *If φ has a three-valued signal $s = \bigvee_j \bigvee_k s_{j,k}$ with connected components $s_{j,k}$ and ψ has a signal w, then $\varphi \,\mathcal{U}_K\, \psi$ has a signal given by,*

$$s \,\mathcal{U}_K\, w \triangleq \bigvee_j \bigvee_k s_{j,k} \wedge \mathcal{F}_K(s_{j,k} \wedge w)\,. \tag{6.1}$$

Proof. If $(s \,\mathcal{U}_K\, w)(t) = \mathbf{T}$ then for some j, k,

$$s_{j,k} \wedge \mathcal{F}_K(s_{j,k} \wedge w)\,(t) = \mathbf{T}.$$

Hence $s_{j,k}(t) = \mathbf{T}$ and for some $t' \in t + K$, $s_{j,k}(t') = \mathbf{T}$ and $w(t) = \mathbf{T}$. However, since $s_{j,k}$ is connected, we conclude $s_{j,k}(t'') = \mathbf{T}$ for all $t'' \in [t, t']$ and hence $\mathbf{x} \models_t \varphi$ for all $t'' \in [t, t']$, showing $\mathbf{x} \models_t \varphi \,\mathcal{U}_K\, \psi$.

Suppose that $\mathbf{x} \not\models_t \varphi \,\mathcal{U}_K\, \psi$. Then $s(t) \neq \mathbf{F}$ and for some $t' \in t + K$, $w(t) \neq \mathbf{F}$ and for all $t'' \in [t, t']$, $s(t) \neq \mathbf{F}$ and $w(t) \neq \mathbf{F}$. Then since $s(t'') \neq F$ on the connected region $[t, t']$, there must hence be some l with $[t, t'] \subseteq I_m$. But then, (picking some m given l), $s_{lm} \wedge \mathcal{F}_K(s_{lm} \wedge s_\psi)\,(t) \neq \mathbf{F}$, and so

$$(s \,\mathcal{U}_K\, w)(t) = \bigvee_j \bigvee_k s_{j,k} \wedge \mathcal{F}_K(s_{j,k} \wedge w)\,(t) \neq \mathbf{F}$$

completing the proof by contrapositive. □

6.1.2 Signals for Atomic Propositions

We now turn our attention to generating a signal for an atomic proposition $\rho \triangleq p > 0$ with defining polynomial p based on a Flow* flowpipe.

We do this in a single pass algorithm which iterates over each time segment of the flowpipe and encloses all roots of p for the current time-step. For each time-step $[t_k, t_{k+1}]$, Flow* provides two Taylor models, $\left(\mathbf{q}^{(k)}_{\text{post}}, \mathbf{I}^{(k)}_{\text{post}}\right)$ and $\left(\mathbf{q}^{(k)}_{\text{pre}}, \mathbf{I}^{(k)}_{\text{pre}}\right)$, whose composition encloses all trajectories of the system over the time step. The value of p over system trajectories is hence enclosed by the Taylor model $G^{(k)}_p(\mathbf{s}, t)$ defined by the composition

$$G^{(k)}_p \triangleq p \,\square\, \left(\mathbf{q}^{(k)}_{\text{post}}, \mathbf{I}^{(k)}_{\text{post}}\right) \square \left(\mathbf{q}^{(k)}_{\text{pre}}, \mathbf{I}^{(k)}_{\text{pre}}\right) \tag{6.2}$$

where $t \in [t_k, t_{k+1}]$ and $\mathbf{s}$ ranges over the space domain $\mathbf{S}$ of the flowpipe (in the case of preconditioned Taylor models, $\mathbf{S}$ is the n-dimensional box $[-1, 1]^n$ [109, 251]). Therefore, we have an interval extension of p over the time interval $[t_k, t_{k+1}]$ given by the interval function $H^{(k)}_p(t) \triangleq G^{(k)}_p(S, t)$ which may be evaluated using interval arithmetic.

We may then determine a signal for ρ.

Proposition 6.1.9. *Given an atomic proposition $\rho = p(\mathbf{x}) > 0$, the three-valued signal $s \triangleq (I_j, s_j)_j$ is a signal for ρ where I_j are the interval components of*

$$[0, T] \setminus \bigcup \left\{ x_0 \;\middle|\; x_0 \in \text{roots}\left(H^{(k)}_p, \frac{\mathrm{d}}{\mathrm{d}t} H^{(k)}_p, [t_k, t_{k+1}], \overline{\mathbb{R}}, \tau\right) \text{ for some } k \right\},$$

s_j is $\mathbf{T}$ iff $H^{(k)}_p(t') > 0$ for some k and $t' \in I_j \cap [t_k, t_{k+1}]$, and $\tau \in \mathbb{R}_{\geq 0}$ is our desired monitoring tolerance parameter.

The unknown regions are given by amalgamating the roots of $H^{(k)}_p$ over each time step. These roots are soundly enclosed by applying the extended interval Newton-Raphson

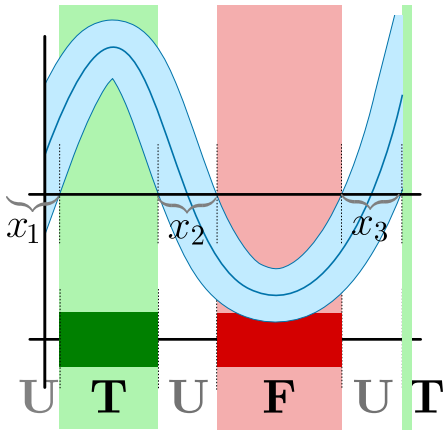


Figure 6.2: Transition from root finding to three-valued signals.

method (Algorithm 1) to $H_p^{(k)}$ (using its derivative $\frac{\mathrm{d}}{\mathrm{d}t}H_p^{(k)}$ which may be derived by Taylor model differentiation [48]), and are guaranteed to enclose the roots of p. Then ρ must have a consistent Boolean value in between these roots, which we may sample by performing interval evaluation of $H_p^{(k)}$ (see Fig. 6.2).

6.1.3 Efficient Monitoring of Composed Taylor Models

The method described in Section 6.1.2 relies on being able to efficiently compute the interval function $H_p^{(k)}$ defined as a symbolic composition of Taylor models (Eq. (6.2)). This is potentially very expensive since the composition involves symbolic operations on high-order polynomials and a flowpipe may consist of thousands of time steps, each requiring a separate composition.

However, since we only need to deduce the signal for the atomic proposition, rather than the exact function value at each point, it will often be sufficient to inexpensively over-approximate the range of Eq. (6.2) over the current time step via interval arithmetic, which we do by replacing some of the Taylor model compositions (denoted $\square$) with functional compositions (denoted $\circ$). Hence, we use the following adaptive algorithm:

- Perform the interval evaluation stepwise using interval arithmetic to check if

$$0 \in \operatorname{range}\left[p \circ \left(\mathbf{q}_{\text{post}}^{(k)}, \mathbf{I}_{\text{post}}^{(k)}\right) \circ \left(\mathbf{q}_{\text{pre}}^{(k)}, \mathbf{I}_{\text{pre}}^{(k)}\right)\right]$$

- If so, perform one stage of symbolic composition and check if

$$0 \in \operatorname{range}\left[p \circ \left(\mathbf{q}_{\text{post}}^{(k)}, \mathbf{I}_{\text{post}}^{(k)}\right) \square \left(\mathbf{q}_{\text{pre}}^{(k)}, \mathbf{I}_{\text{pre}}^{(k)}\right)\right]$$

- If the result is still ambiguous, perform full symbolic composition of $G_p^{(k)}$ for the current time step and apply root finding.

Hence, we are able to generate a precise signal for an atomic proposition over the whole time domain, whilst only performing symbolic Taylor model composition and root finding on demand where necessary to disambiguate the result of the signal (i.e. near the roots of the atomic proposition). We may additionally skip the composition of the preconditioned Taylor model on dimensions which do not correspond to any variable of p.

This method may, however, still spend effort trying to determine the truth value of the signal of an atomic proposition in regions of time which are not crucial to the truth of the overall signal; this issue is addressed in Chapter 7 with the introduction of *masks*.

6.2 Monitoring $\mathcal{LBUC}$

We will now apply the monitoring techniques discussed in the previous section, to specify a complete verified monitoring procedure for $\mathcal{LBUC}$ over bond-calculus models. We define the monitoring procedure for $\mathcal{LBUC}$ defining two mutually recursive functions, $\texttt{signal}(\varphi, (\Pi_0, \mathcal{A}), d)$ and $\texttt{signalF}(\varphi, \Pi_f, \mathcal{A}, \mathbf{S}, T)$, the first of which defines a signal for a property φ in a bond-calculus model and the latter of which defines a signal for φ given a Flow* flowpipe Π_f over space domain $\mathbf{S}$ and time domain $T \subseteq \mathbb{R}_{\geq 0}$ given an affinity network $\mathcal{A}$.

We first define the time duration of a flowpipe required for monitoring a given property.

Definition 6.2.1. The *duration* of a given STL formula is defined by

$$\begin{aligned}
\texttt{duration}(\rho) &\triangleq 0 \\
\texttt{duration}(\neg\varphi) &\triangleq \texttt{duration}(\varphi) \\
\texttt{duration}(\varphi \wedge \psi) &\triangleq \max(\texttt{duration}(\varphi), \texttt{duration}(\psi)) \\
\texttt{duration}(\varphi \vee \psi) &\triangleq \max(\texttt{duration}(\varphi), \texttt{duration}(\psi)) \\
\texttt{duration}(\mathcal{F}_{[a,b]}\, \varphi) &\triangleq b + \texttt{duration}(\varphi) \\
\texttt{duration}(\mathcal{G}_{[a,b]}\, \varphi) &\triangleq b + \texttt{duration}(\varphi) \\
\texttt{duration}(\varphi\, \mathcal{U}_{[a,b]}\, \psi) &\triangleq b + \max(\texttt{duration}(\varphi), \texttt{duration}(\psi)) \\
\texttt{duration}(\mathcal{C} \triangleright \varphi) &\triangleq 0
\end{aligned}$$

This then allows us to define the first of our monitoring functions as follows.

Definition 6.2.2. We may compute a duration d signal $\texttt{signal}(\varphi, (\Pi_0, \mathcal{A}), d)$ for a $\mathcal{LBUC}$ property φ given uncertain bond-calculus model $\left(\widehat{\Pi}_0, \widehat{\mathcal{A}}\right)$ by

$$\texttt{signal}(\varphi, (\Pi_0, \mathcal{A}), d) \triangleq \texttt{signalF}(\varphi, \Pi_f, \mathcal{A}, \mathbf{S}, [0, D])$$

where $D = d + \texttt{duration}(\varphi)$ and Π_f is a duration D flowpipe for the interval IVP

$$\frac{\mathrm{d}\Pi}{\mathrm{d}t} \triangleq \frac{\mathrm{d}(\Pi_0, \mathcal{A})}{\mathrm{d}t}; \quad \Pi(0) \in \Pi_0$$

with space domain $\mathbf{S}$.

This relies on the signal $\texttt{signalF}(\varphi, \Pi_f, \mathcal{A}, \mathbf{S}, T)$ for a property φ over flowpipe Π_f given affinity network $\mathcal{A}$, flowpipe space domain $\mathbf{S}$, and interval time domain T, which is defined as follows.

Atomic Propositions For an atomic proposition φ, the signal $\texttt{signalF}(\varphi, \Pi_f, \mathcal{A}, \mathbf{S}, T)$ is computed by applying the procedure set out in Sections 6.1.2 and 6.1.3.

Logical and Temporal Operators We build up signals for complex properties according to the operations in Section 6.1.1 so that we have,

$$\begin{aligned}
\texttt{signalF}(\neg\varphi, \Pi_f, \mathcal{A}, \mathbf{S}, T) &\triangleq \neg\texttt{signalF}(\varphi, \Pi_f, \mathcal{A}, \mathbf{S}, T) \\
\texttt{signalF}(\varphi \wedge \psi, \Pi_f, \mathcal{A}, \mathbf{S}, T) &\triangleq \texttt{signalF}(\varphi, \Pi_f, \mathcal{A}, \mathbf{S}, T) \wedge \texttt{signalF}(\psi, \Pi_f, \mathcal{A}, \mathbf{S}, T) \\
\texttt{signalF}(\varphi \vee \psi, \Pi_f, \mathcal{A}, \mathbf{S}, T) &\triangleq \texttt{signalF}(\varphi, \Pi_f, \mathcal{A}, \mathbf{S}, T) \vee \texttt{signalF}(\psi, \Pi_f, \mathcal{A}, \mathbf{S}, T) \\
\texttt{signalF}(\mathcal{F}_K\,\varphi, \Pi_f, \mathcal{A}, \mathbf{S}, T) &\triangleq \mathcal{F}_K(\texttt{signalF}(\varphi, \Pi_f, \mathcal{A}, \mathbf{S}, T)) \\
\texttt{signalF}(\mathcal{G}_K\,\varphi, \Pi_f, \mathcal{A}, \mathbf{S}, T) &\triangleq \mathcal{G}_K(\texttt{signalF}(\varphi, \Pi_f, \mathcal{A}, \mathbf{S}, T)) \\
\texttt{signalF}(\varphi\,\mathcal{U}_K\,\psi, \Pi_f, \mathcal{A}, \mathbf{S}, T) &\triangleq \texttt{signalF}(\varphi, \Pi_f, \mathcal{A}, \mathbf{S}, T)\,\mathcal{U}_K\,\texttt{signalF}(\psi, \Pi_f, \mathcal{A}, \mathbf{S}, T).
\end{aligned}$$

Context Operators To compute $\texttt{signalF}\,(\mathcal{C} \triangleright \varphi, \Pi_f, \mathcal{A}, \mathbf{S}, T)$, we first compute an interval uncertain model encompassing the range of possible states over the time domain T. This can be derived as the bond-calculus model

$$\Phi \triangleq (\Pi_f(\mathbf{S}, T), \mathcal{A})$$

which is derived by first computing the range of the flowpipe over the space and time domain and then combining this with the (interval) affinity network $\mathcal{A}$. We next compute the signal

$$s \triangleq \texttt{signal}\,(\varphi, \Phi \parallel \mathcal{C}, 0) \tag{6.3}$$

for φ in the composed system $\Phi \parallel \mathcal{C}$, which places the current state of the system in the context $\mathcal{C}$. If either $s(0) \in \mathbb{B}$ or $\text{width}(T) < \varepsilon$ (where the parameter ε is our desired precision of monitoring) then we are done and can return the signal $(T, s(0))$. If not, we bisect the interval into two intervals T_1, T_2 and return the union of the signals $\texttt{signalF}(\mathcal{C} \triangleright \varphi, \Pi_f, \mathcal{A}, \mathbf{S}, T_1)$ and $\texttt{signalF}(\mathcal{C} \triangleright \varphi, \Pi_f, \mathcal{A}, \mathbf{S}, T_2)$.

6.3 Unbounded Temporal Operators

Our method so far is restricted to verifying properties featuring temporal operators with bounded time intervals since Flow* can only construct flowpipes covering a finite time horizon. However, in general three-valued signals are defined over the unbounded time domain $[0, \infty)$ (since they may be unknown over the unbounded time regions on which

they have not been computed), which offers a way to verify unbounded time properties given appropriate signals. Whilst the general problem of unbounded-time verification for continuous system is known to be extremely challenging, it is sometimes possible to combine flowpipes with deductive reasoning in order to verify unbounded time properties and determine the long term behaviour of a system [333]. In this section we will extend the approach of Sogokon, Jackson, and Johnson [333] and show how invariants of continuous systems can often be combined with our verified monitoring results in order to derive unbounded-time signals for atomic propositions and hence to verify unbounded temporal operators.

We first note that our interval representation of signals extends directly to signals of the form

$$s = (([a_1, b_1], s_1), \ldots, ([a_{n-1}, b_{n-1}], s_{n-1}), ([a_n, \infty), s_n))$$

where the final interval extends to infinity, and we may also allow such unbounded intervals to appear in temporal operators. All of the normal logical operators may be applied to unbounded signals in much the same manner as before, whilst for the temporal operators we may used interval arithmetic extended with unbounded open endpoints[1] so that, in particular,

$$\begin{aligned}
[a,b] - [c,\infty) &= (-\infty, b-c], & [a,b] \dot{-} [c,\infty) &= \varnothing, \\
[c,\infty) - [a,b] &= [c-b,\infty), & [c,\infty) \dot{-} [a,b] &= [c-a,\infty), \\
[a,\infty) - [c,\infty) &= (-\infty,\infty), & [a,\infty) \dot{-} [c,\infty) &= [a-c,\infty).
\end{aligned}$$

which results in the following computational rules for the eventually signal,

$$\begin{aligned}
\mathcal{F}_{[a,b]}(([c,\infty), \mathbf{T})) &= (([\min(0, c-b),\infty), \mathbf{T})) \\
\mathcal{F}_{[a,b]}(([c,\infty), \mathbf{F})) &= (([\min(0, c-a),\infty), \mathbf{F})) \\
\mathcal{F}_{[c,\infty)}(([a,b], \mathbf{T})) &= (([0, b-c], \mathbf{T})) \\
\mathcal{F}_{[c,\infty)}(([a,b], \mathbf{F})) &= (([0, a-c], \mathbf{F})) \\
\mathcal{F}_{[c,\infty)}(([a,\infty), \mathbf{T})) &= (([0,\infty), \mathbf{T})) \\
\mathcal{F}_{[c,\infty)}(([a,\infty), \mathbf{F})) &= (([\min(0, a-c),\infty), \mathbf{F}))
\end{aligned}$$

Example 6.3.1. Given a signal

$$s = (([2,3], \mathbf{T}), ([4,5], \mathbf{F}))$$

[1] This corresponds to the standard behaviour of many interval arithmetic libraries including that of Sagemath, and can be formally understood in extended interval arithmetics such as [256].

we can compute the unbounded eventually signal as

$$\mathcal{F}_{[1,\infty)}\, s = (([0,2],\mathbf{T}),(\varnothing,\mathbf{F})) = (([0,2],\mathbf{T}))$$

or the unbounded globally signal as

$$\mathcal{G}_{[1,\infty)}\, s = \neg\, \mathcal{F}_{[1,\infty)}\, \neg s = \neg((\varnothing,\mathbf{F}),([0,4],\mathbf{T})) = (([0,4],\mathbf{F})).$$

This example shows that even bounded signals may be sufficient to verify unbounded eventually operators or to refute unbounded globally operators.

Example 6.3.2. Given an unbounded signal

$$s = (([2,3],\mathbf{T}),([4,5],\mathbf{F}),([6,\infty),\mathbf{T}))$$

we can compute the unbounded eventually signal as

$$\mathcal{F}_{[1,\infty)}\, s = (([0,2],\mathbf{T}),(\varnothing,\mathbf{F}),([0,\infty),\mathbf{T})) = (([0,\infty),\mathbf{T}))$$

or the unbounded globally signal as

$$\mathcal{G}_{[1,\infty)}\, s = \neg\, \mathcal{F}_{[1,\infty)}\, \neg s = \neg((\varnothing,\mathbf{F}),([5,\infty),\mathbf{T}),(\varnothing,\mathbf{F})) = (([5,\infty),\mathbf{F})).$$

As seen in the above examples, whilst bounded signals may sometimes be sufficient to prove unbounded eventually properties, if we wish to verify unbounded safety properties (that is, to verify $\mathcal{G}_{[0,\infty)}\,\varphi$ or refute $\mathcal{F}_{[0,\infty)}\,\varphi$) we need more information than that which is contained in the bounded signals we may derive from the finite duration flowpipes which Flow* is able to generate for atomic propositions. We will now, however, see that the method of safety verification based on invariants can often allow us to extend a bounded signal to an unbounded one.

To this end we first need the concept of a continuous invariant set, which is a generalization of the notion of *positively invariant sets* [332] from dynamical systems theory which coincides with the requirements for verifying unbounded safety properties[2].

Definition 6.3.3. A set $I \subseteq \mathbb{R}^n$ is a *positive invariant set* (or, simply, *invariant*) for a system if every system trajectory $\mathbf{x}$ satisfies

$$\forall t \in [0,\infty)\Big(\mathbf{x}(t) \in I \implies \Big(\forall t' \in [t,\infty), \mathbf{x}(t') \in I\Big)\Big)$$

that is, if the system satisfies the STL formula

$$\mathbf{x} \in I \implies \mathcal{G}_{[0,\infty)}(\mathbf{x} \in I).$$

[2]The generalized notion of continuous invariants [297] is also frequently used when applying invariant-based reasoning to hybrid systems verification.

We next see that a suitable invariant allows us to extend a bounded signal into an unbounded one, generalizing [332, Proposition 22.].

Proposition 6.3.4. *Given a proposition φ and a signal*

$$s = (([a_1, b_1], s_1), \ldots, ([a_n, b_n], s_n))) \tag{6.4}$$

for φ, if there exists an invariant I such that φ has truth value s_n on I (that is, for every trajectory $\mathbf{x}$ with $\mathbf{x}(0) \in I$, $\mathbf{x} \models_0 (\varphi \Leftrightarrow s_n)))$ and the system satisfies

$$\mathcal{F}_{[a_n,b_n]}(\mathbf{x} \in I)$$

then the unbounded extension of *s,*

$$s^\infty = (([a_1, b_1], s_1), \ldots, ([a_n, \infty), s_n)))$$

is also a signal for φ.

Proof. Take any system trajectory $\mathbf{x}$ and any time point $t \in [0, \infty)$ such that $s^\infty(t) = \mathbf{T}$. If $t \leq b_n$ then $s(t) = s^\infty(t) = \mathbf{T}$ and so $\mathbf{x} \models_t \varphi$. If, however, $t > b_n$, then since $\mathcal{F}_{[a_n,b_n]}(\mathbf{x} \in I)$ holds, we must have some $t_0 \in [a_n, b_n]$ such that $\mathbf{x}(t_0) \in I$ and hence $\mathbf{x}(t) \in I$ since I is a invariant and $t > t_0$. Therefore $\mathbf{x} \models_t \varphi$ since this is equivalent to $\mathbf{x}_{t_0} \models_{t-t_0} \varphi$ where $\mathbf{x}_{t_0}(t) \triangleq \mathbf{x}(t + t_0)$ is a trajectory of the system with initial condition in I. Similarly, if $s^\infty = \mathbf{F}$ then $\mathbf{x} \models_t \neg\varphi$, proving s^∞ is a signal for ρ. □

This means that if we have used the Flow* flowpipe F to generate the signal s (Eq. (6.4)) for atomic proposition $\rho \triangleq p(\mathbf{x}) > 0$, we can extended it into an unbounded signal s^∞ as long as we are able to find an invariant I for the system satisfying

$$F(t_0) \subseteq I \subseteq \left\{\mathbf{x} \;\middle|\; (p(\mathbf{x}) > 0) \Leftrightarrow s_n\right\}$$

for some $t_0 \in [a_n, b_n]$. Thus suitable invariants allow us to derive unbounded signals for atomic propositions from flowpipes. This then makes it possible to verify complex propositions involving unbounded temporal operators by combining the unbounded signals from atomic propositions in the normal way, applying extended interval arithmetic where necessary.

This has a useful special case when an atomic proposition (or its negation) defines an invariant of the system.

Corollary 6.3.5. *Given an atomic proposition* $\rho \triangleq p(\mathbf{x}) > 0$ *if*

$$s = (([a_1, b_1], s_1), \ldots, ([a_n, b_n], s_n)))$$

is a signal for ρ *and the set*

$$I \triangleq \begin{cases} \{\mathbf{x} \mid p(\mathbf{x}) > 0\} & \text{if } s_n = \mathbf{T} \\ \{\mathbf{x} \mid p(\mathbf{x}) < 0\} & \text{if } s_n = \mathbf{F} \end{cases}$$

is an invariant, then

$$s^{\infty} = (([a_1, b_1], s_1), \ldots, ([a_n, \infty), s_n)))$$

is also a signal for ρ*.*

Our ability to verify unbounded properties via this approach depends crucially on our ability to synthesise appropriate invariants. Invariant synthesis in an ongoing challenging problem in continuous systems verification [332] and currently often requires specific insight about the system at hand, in contrast to the fully automated methods which are our main focus. Whilst we will demonstrate invariant-based verification for specific systems, the general problem of invariant synthesis required to automate this method is outside of our scope. It is, however, possible to verify that a given semi-algebraic set is a invariant of a polynomial system in an automated manner based on the decision procedure of Liu, Zhan, and Zhao [242]; other methods of invariant checking are surveyed in [332, Chapter 4]. This method also only handles properties for which the truth-value of atomic propositions eventually stabilises. Thus, our method is unable to verify properties such as $\mathcal{G}_{[0,\infty)}(\mathcal{F}_{[0,\infty)}\,\rho \wedge \mathcal{F}_{[0,\infty)}(\neg\rho))$, which requires that the truth value of ρ oscillates arbitrarily many times. Therefore, additional verification methods will be required for a more general treatment of unbounded time properties.

6.4 Demonstration: 9-Dimensional Genetic Oscillator

We will now examine the performance of our core STL monitoring algorithm by considering the genetic oscillator model [354]. This consists of a gene regulatory network involving genes A and B which bind to each other and whose translation is governed by the binding of A to their respective promoters. This model is a standard benchmark of Flow* verified integration performance [110] and has also been considered in [74] to compare Stochastic π and PEPA modelling styles. In our performance analysis, we will assess the core design

choices of our monitoring algorithms by benchmarking our algorithm against a "closed-box" monitoring approach, which treats the entire flowpipe as an interval function, and compare the functional composition and symbolic composition approaches (as introduced in Section 2.4.4) for evaluating atomic propositions over the flowpipe.

6.4.1 Bond-Calculus Model

We first model the genetic oscillator of [354] in the bond-calculus based on the modelling styles presented in Chapter 4. To this end we first explicitly model promoters to which proteins may bind and unbind:

$$\begin{aligned}
\textsc{Promoter}_X &\triangleq \mathrm{bindP}_X(\ell).\textsc{PromoterBound}_{X,\ell} \\
&\quad + \mathrm{transcribe}_X.\,(\textsc{Promoter}_X \mid \textsc{mRNA}_X) \\
\textsc{PromoterBound}_{X,\ell} &\triangleq \mathrm{unbindP}_X@\ell.\textsc{Promoter} \\
&\quad + \mathrm{transcribeB}_X@\ell.\,(\textsc{PromoterBound}_{X,\ell} \mid \textsc{mRNA}_X)
\end{aligned}$$

and the protein $\textsc{Protein}_X$ which can bind, unbind, or decay,

$$\begin{aligned}
\textsc{Protein}_X &\triangleq \mathrm{bind}_X(\ell).\textsc{ProteinBound}_{X,\ell} + \mathrm{degrade}_X.\mathbf{0} \\
\textsc{ProteinBound}_{X,\ell} &\triangleq \mathrm{unbind}_X@\ell.\textsc{Protein}_{X,\ell} + \mathrm{degradeB}_X@\ell.\mathbf{0}
\end{aligned}$$

we then model the mRNA which control the transcription of each protein

$$\textsc{mRNA}_X \triangleq \mathrm{degradeM}_X.\mathbf{0} + \mathrm{translate}_X.\,(\textsc{Protein}_X \mid \textsc{mRNA}_X)$$

These definitions specialise to the network of [354] when instantiated for proteins $X = A$ and $X = R$ with the rates of each interaction defined via the affinity network,

$$\mathcal{A} \triangleq \left\{ \begin{array}{llll}
\mathrm{bind}_A \parallel \mathrm{bindP}_A & @\ 0.1, & \mathrm{unbind}_A \parallel \mathrm{unbindP}_A & @\ 50, \\
\mathrm{bind}_A \parallel \mathrm{bindP}_R & @\ 1, & \mathrm{unbind}_A \parallel \mathrm{unbindP}_R & @\ 100, \\
\mathrm{bind}_A \parallel \mathrm{bind}_R & @\ 2, & \mathrm{degradeB}_A \mid \mathrm{unbind}_R & @\ 1, \\
\mathrm{transcribe}_A & @\ 0.5, & \mathrm{transcribeB}_A & @\ 5, \\
\mathrm{transcribe}_R & @\ 0.01, & \mathrm{transcribeB}_R & @\ 50, \\
\mathrm{translate}_A & @\ 50, & \mathrm{translate}_R & @\ 0.5, \\
\mathrm{degrade}_A & @\ 1, & \mathrm{degradeM}_A & @\ 10, \\
\mathrm{degrade}_R & @\ 0.2, & \mathrm{degradeM}_R & @\ 0.5
\end{array} \right\}$$

using the same rate parameters as [111]. The molecular interactions corresponding to these affinity patterns are shown in [354, Figure 1].

When the bond-calculus is applied to this system starting from at least the two promoters, we can see that it dynamically generates a system involving 9 species, which we abbreviate as follows (in accordance with [111]),

$$
\begin{aligned}
X_1 &\triangleq \textsc{Promoter}_A \\
X_2 &\triangleq \textsc{Promoter}_R \\
X_3 &\triangleq \nu\, \ell\, (\textsc{ProteinBound}_{A,\ell} \mid \textsc{PromoterBound}_{A,\ell}) \\
X_4 &\triangleq \nu\, \ell\, (\textsc{ProteinBound}_{A,\ell} \mid \textsc{PromoterBound}_{R,\ell}) \\
X_5 &\triangleq \textsc{mRNA}_A \\
X_6 &\triangleq \textsc{Protein}_A \\
X_7 &\triangleq \textsc{mRNA}_R \\
X_8 &\triangleq \textsc{Protein}_R \\
X_9 &\triangleq \nu\, \ell\, (\textsc{ProteinBound}_{A,\ell} \mid \textsc{ProteinBound}_{R,\ell})
\end{aligned}
$$

Following [111], we are interested in the initial mixture

$$
\begin{aligned}
\Pi_0 \triangleq\ & \left[0.98, 1.02\right] X_1 \parallel \left[1.28, 1.32\right] X_2 \parallel \left[0.08, 0.12\right] X_3 \parallel \left[0.08, 0.12\right] X_4 \parallel \left[0.08, 0.12\right] X_5 \\
\parallel\ & \left[1.28, 1.32\right] X_6 \parallel \left[2.48, 2.52\right] X_7 \parallel \left[0.58, 0.62\right] X_8 \parallel \left[1.28, 1.32\right] X_9
\end{aligned}
$$

The overall bond-calculus model is thus defined by $\mathcal{M} \triangleq (\Pi_0, \mathcal{A})$.

Finally, we see that applying the bond-calculus' continuous ODE semantics to the model $(\Pi_0, \mathcal{A})$ results in the following system of 9 coupled ODEs,

$$
\begin{aligned}
\mathrm{d}\,[X_1]/\mathrm{d}t &\triangleq 50\,[X_3] - 0.1\,[X_1]\,[X_6] \\
\mathrm{d}\,[X_2]/\mathrm{d}t &\triangleq 100\,[X_4] - [X_2]\,[X_6] \\
\mathrm{d}\,[X_3]/\mathrm{d}t &\triangleq 0.1\,[X_1]\,[X_6] - 50\,[X_3] \\
\mathrm{d}\,[X_4]/\mathrm{d}t &\triangleq\ [X_2]\,[X_6] - 100\,[X_4] \\
\mathrm{d}\,[X_5]/\mathrm{d}t &\triangleq 5\,[X_3] + 0.5\,[X_1] - 10\,[X_5] \\
\mathrm{d}\,[X_6]/\mathrm{d}t &\triangleq 50\,[X_5] + 50\,[X_3] + 100\,[X_4] - [X_6]\,(0.1\,[X_1] + [X_2] + 2\,[X_8] + 1) \\
\mathrm{d}\,[X_7]/\mathrm{d}t &\triangleq 50\,[X_4] + 0.01\,[X_2] - 0.5\,[X_7] \\
\mathrm{d}\,[X_8]/\mathrm{d}t &\triangleq 0.5\,[X_7] - 2\,[X_6]\,[X_8] + [X_9] - 0.2\,[X_8] \\
\mathrm{d}\,[X_9]/\mathrm{d}t &\triangleq 2\,[X_6]\,[X_8] - [X_9]
\end{aligned}
$$

matching the model of [354].

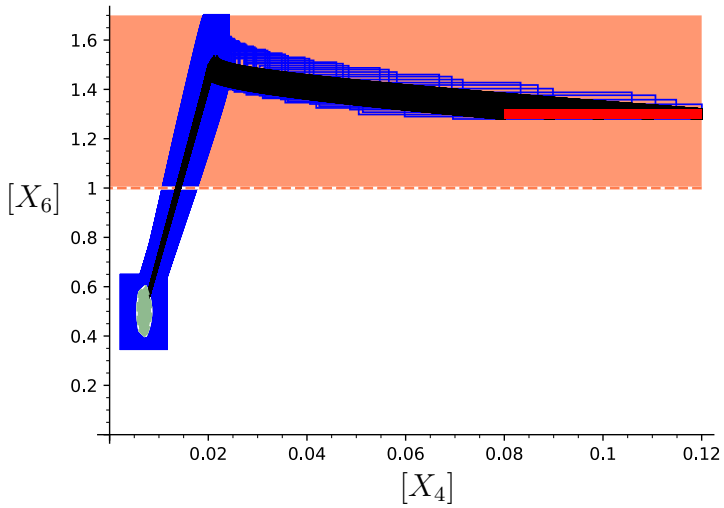


Figure 6.3: An interval over-approximation of the Flow* flowpipe at each time step is illustrated in blue, numerical trajectories for different initial conditions in black, initial conditions in red, and the regions involved in properties P and Q are in orange and green respectively.

6.4.2 Model Analysis and Monitoring Performance

The evolution of the variables $[X_4]$ and $[X_6]$ over 5 hours (of simulated model time) is shown in Fig. 6.3 which includes numerical traces from a sample of many different fixed initial conditions alongside a coarse interval over-approximation of a Flow* flowpipe covering the whole box of uncertain initial conditions. This shows the system moving from the red box of initial conditions through the orange region before the converging to the green region. We can describe the temporal behaviour of this system much more precisely with STL properties such as

$$\varphi \triangleq \mathcal{G}_{[0,1.5]}\Big(P \vee \mathcal{G}_{[3,3.5]}(Q)\Big)$$

in which we have polynomial atomic propositions

$$P \triangleq [X_6] - 1 > 0$$

and

$$Q \triangleq 0.032 - 125^2([X_4] - 0.007)^2 - 3([X_6] - 0.5)^2 > 0.$$

The property φ states that at any point within the first hour, the system will either remain within the half-space P or, at any point between 3 and 3.5 hours in the future will be within the elliptical region Q.

In Fig. 6.4 we break down the time taken to monitor φ for 0.5 hours of simulated time using a number of variants of our monitoring algorithm in order to evaluate the impact

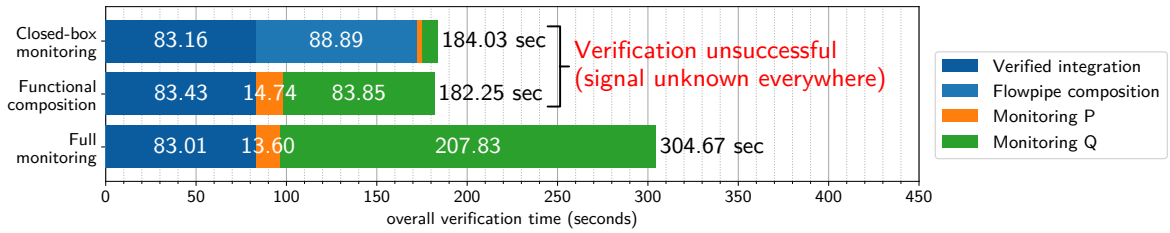


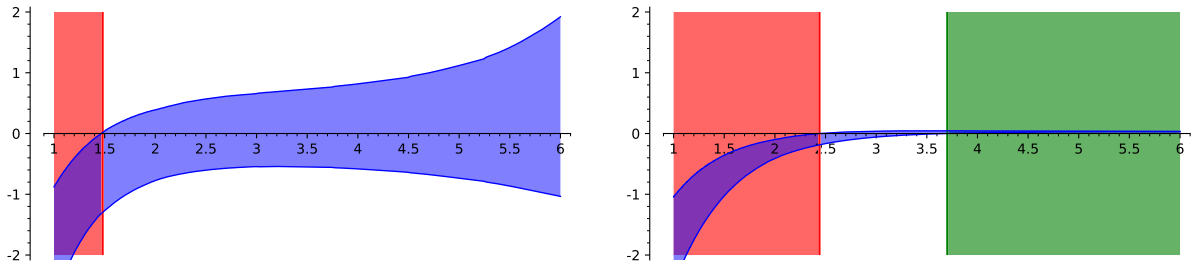
Figure 6.4: Combined Flow* verified integration and STL monitoring times in seconds, showing the cost of each stage for a number of variants of our monitoring algorithm.

of each of its elements on monitoring cost and precision. First we consider the closed box monitoring approach where we first run Flow* to perform verified integration and flowpipe composition, before using interval analysis and functional composition to monitor φ over the entire flowpipe. Whilst the monitoring cost for the propositions P and Q is very small in comparison to the time it took Flow* to perform verified integration, the flowpipe composition stage is more expensive and takes almost as long as the verified integration itself. Next we monitor φ in the same way, but perform the flowpipe composition on demand as described in Section 6.1.3. We see that if we just monitor the simple atomic proposition P we save most of the cost of flowpipe composition, although once we also monitor Q we need to pay the full cost. These two methods also do not yield sufficient precision to verify φ, both producing a useless signal which is unknown everywhere. This imprecision can be seen in Fig. 6.5a which shows the result of monitoring the complex polynomial atomic proposition Q over the flowpipe using functional composition and the corresponding signal.

In order to produce a useful signal for φ we need to run our full monitoring algorithm, permitting symbolic composition at each stage. Whilst the monitoring cost for the simple proposition P is similar to before, the cost for the complex proposition Q is significantly higher. This, however, now gives a much more precise signal for Q as shown in Fig. 6.5b. This means we now get the overall signal $s = (([0], \mathbf{T}))$ for φ, allowing us to verify that φ is true at time 0 and that $P \vee \mathcal{G}_{[3,3.5]}(Q)$ holds for at least the first 1.5 hours.

6.5 Demonstration: Predatory-Prey Role-Reversal Model

We firstly investigate the application of our monitoring algorithm to an ecological model of predator-prey interaction with concentration dependent species roles. Lotka-Volterra type models are classically used to model interactions between a prey species and a predatory



(a) Function Composition

(b) Symbolic Composition

Figure 6.5: Monitoring Q.

which grows by consuming their prey and have been applied to modelling a wide variety of ecosystems, but typically assume that the role of each species remained fixed. However, ecosystems have been observed in which, under certain circumstances, the predatory can undergo a role-reversal and become the prey. For example, a classic study [28] of populations of rock lobsters and whelks on islands of the west coast of South Africa observed that, in the particular case of Marcus Island, the usually dominant lobsters are overwhelmed and consumed by the whelks due to their numerical superiority. This suggested a multi-stable ecosystem with stable states in which each of the species is dominant and that the predator-prey relationship within this ecosystem is population dependant. Role-reversal is also thought to play an important role in other aquatic ecosystems where many species of fish are able to prey upon juvenile forms of their usual predators [156].

We will look at a modified Lotka-Volterra model proposed by [321] as a theoretical model of role-reversal. This involves concentrations of two species, which, based on the prototypical scenario of [28], we may call $W = [\text{WHELK}]$ and $L = [\text{LOBSTER}]$. The species may be modelled by bond-calculus processes,

$$
\begin{aligned}
\text{WHELK} \triangleq\ & \text{dieWhelk}.\mathbf{0} \\
& + \text{growWhelk}.(\text{WHELK}|\text{WHELK}) \\
& + \text{beWhelk}.\text{WHELK} \\
\text{LOBSTER} \triangleq\ & \text{dieLobster}.\mathbf{0} \\
& + \text{beLobster}.\text{LOBSTER} \\
& + \text{growLobster}.(\text{LOBSTER}|\text{LOBSTER})
\end{aligned}
$$

the affinity network,

$$\mathcal{A}_{k,b,c,e,f} \triangleq \left\{ \begin{array}{c} \text{growWhelk @ NLGrowth}_b \\ \text{dieWhelk} \parallel \text{beLobster @ RRPredation}_{c,k} \\ \text{dieLobster @ NLDecay}_e \\ \text{beWhelk} \parallel \text{growLobster @ RRPredation}_{f,k} \end{array} \right\}$$

which uses the following nonlinear rate laws,

$$\begin{aligned} \text{NLGrowth}_g([X]) &\triangleq g\,[X]\,(1-[X]) \\ \text{NLDecay}_d([X]) &\triangleq d\,[X]\,(1+[X]) \\ \text{RRPredation}_{g,h}([X],[Y]) &\triangleq g\,[X]\,(h-[X])\,[Y] \end{aligned}$$

This network models predation via a pair of patterns: dieWhelk $\parallel$ beLobster @ $\text{RRPredation}_{c,k}$ which causes units c of whelk biomass consumption, and beWhelk $\parallel$ growLobster @ $\text{RRPredation}_{f,k}$ which causes f units of lobster biomass production, based on the overall net predation rate $W(k-W)L$. The threshold parameter k determines the population of whelks at which they become overall predators. Once the whelk population exceeds the threshold k the biomass consumption and biomass production rates both change sign, causing a role-reversal to occur between these consumption and production reactions. The intrinsic growth of whelks and death of lobsters are modelled by nonlinear growth laws governed by rate parameters b and e respectively. We note that concentration of whelks and lobsters are not in physical units since the variables of the system have been rescaled to minimize the number of parameters as described in [321].

Applying the ODE extraction algorithm we get the set of two nonlinear ODEs,

$$\begin{aligned} \frac{\mathrm{d}W}{\mathrm{d}t} &\triangleq bW(1-W) - cW(k-W)L \\ \frac{\mathrm{d}L}{\mathrm{d}t} &\triangleq -eL(1+L) + fW(k-W)L \end{aligned}$$

matching the model of [321].

The phase portrait of the system is shown in Fig. 6.6. We consider the eventual evolution of the system from two different sets of uncertain initial conditions:

$$\begin{aligned} \Pi_1 &\triangleq [0.2, 0.4]\ \text{Whelk} \parallel [7, 8]\ \text{Lobster} \\ \Pi_2 &\triangleq [1.0, 1.2]\ \text{Whelk} \parallel [4, 6]\ \text{Lobster}. \end{aligned}$$

We can see that the trajectories of the system tend towards one of two stable fixed points, one involving the extinction of the lobsters, and the other involving an equilibrium of

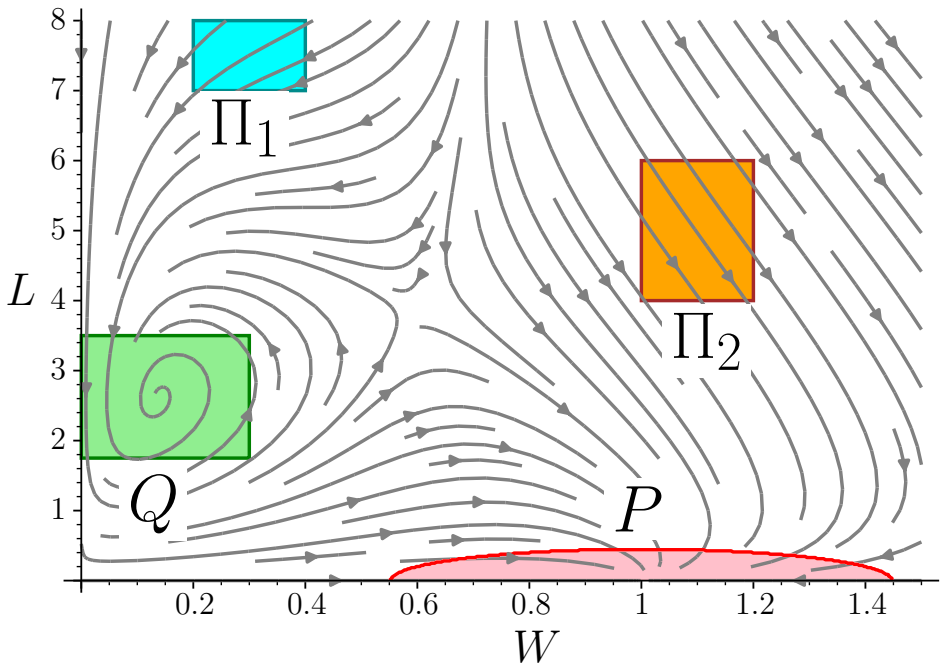


Figure 6.6: Role Reversal Phase Portrait

whelks and lobsters. These may be associated with the following two $\mathcal{LBUC}$ properties,

$$P \triangleq (W-1)^2 + L^2 < 0.2$$
$$Q \triangleq 0 < W \wedge W < 0.3 \wedge 1.75 < L \wedge L < 3.5$$

which correspond respectively to an ellipse and a square region surrounding each fixed point.

To verify the evolution of the system, we consider the property

$$\varphi_1 \triangleq \mathcal{F}_{[0,10]}\, Q$$

which states that at some point within the first 10 time units the systems trajectories will definitely have entered the region Q[3], and the property,

$$\varphi_2 \triangleq \mathcal{F}_{[0,5]}\, \mathcal{G}_{[0,5]}\, P$$

which states that the we will definitely enter P within 5 time units, and will remain inside for at least 5 time units.

Firstly we note that with the initial system $\mathcal{M}_1 \triangleq (\Pi_1, \mathcal{A}_{0.8,0.6,0.3,0.05,2})$, monitoring property φ_1 for 5 time units giving signal $(([0,5], \mathbf{T}))$, whilst monitoring φ_2 for 5 time units gives signal $(([0,5], \mathbf{F}))$. Conversely, with initial system $\mathcal{M}_2 \triangleq (\Pi_2, \mathcal{A}_{0.8,0.6,0.3,0.05,2})$,

[3]The trajectories may, however, subsequently leave Q.

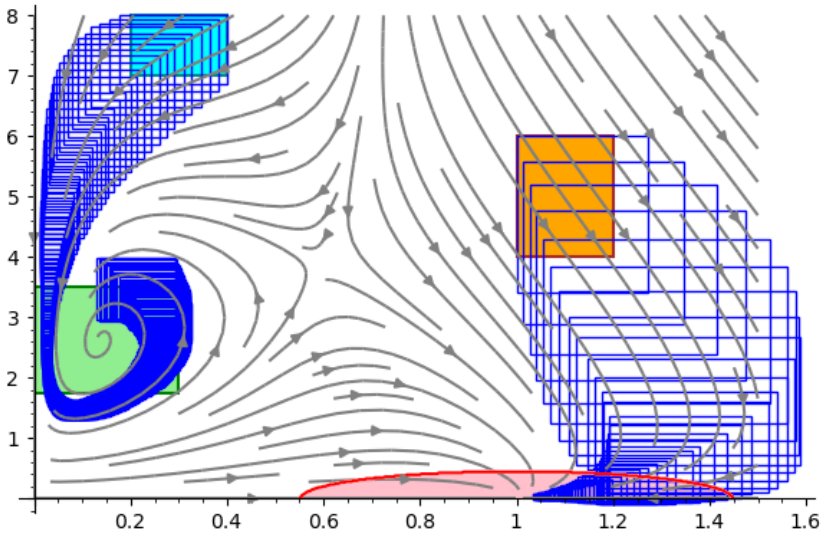


Figure 6.7: Flowpipes from initial conditions Π_1 and Π_2.

monitoring φ_1 for 5 time units gives signal $(([0,5],\mathbf{F}))$ and monitoring φ_2 for 5 time units gives signal $(([0,5],\mathbf{T}))$. This corresponds to the fact that Π_1 is inside the basin of attraction of the first fixed point, whilst Π_2 is inside the basin of attraction of the latter.

We can also use context operators to explore the impact of the introduction of a new population of whelk or lobsters into an existing ecosystem. To this end we consider an initial system $\Pi_3 \triangleq [0.9, 1.1]\ \textsc{Whelk}$ consisting entirely of a population of between 0.9 and 1.1 concentration of whelks, and the property $\varphi_3 \triangleq [0,2]\ \textsc{Lobster} \triangleright \varphi_2$ which states that the introduction of between 0 and 2 concentration of lobsters will lead to property φ_2 being satisfied — that is, the introduced lobster population will be wiped out. Applying our monitoring algorithm for a duration of 5 time units we get the signal $(([0,5],\mathbf{T}))$ for φ_2, confirming that this property does indeed hold. This corresponds to the experimental observations of Barkai and McQuaid [28] who found that when introducing small numbers of lobsters to an island dominated by whelks the whelks overwhelmed and consumed the lobsters returning the ecosystem to its original state.

However, it may be possible that introducing a sufficient population of a species is sufficient to push the ecosystem from one steady state to another, resulting in a shift of the overall ecosystem dynamics. For example, we consider the property

$$\varphi_4 \triangleq \varphi_1\ \mathcal{U}_{[2,4]}\ ([0.9, 1.1]\ \textsc{Whelk} \triangleright \varphi_2)$$

which states that there is a time point between times 2 and 4 at which after introducing 0.9 and 1.1 concentration of whelks, the evolution of the system leads to P (φ_2 will be true), whilst at any point beforehand the evolution of the system lead to Q (φ_1 was true).

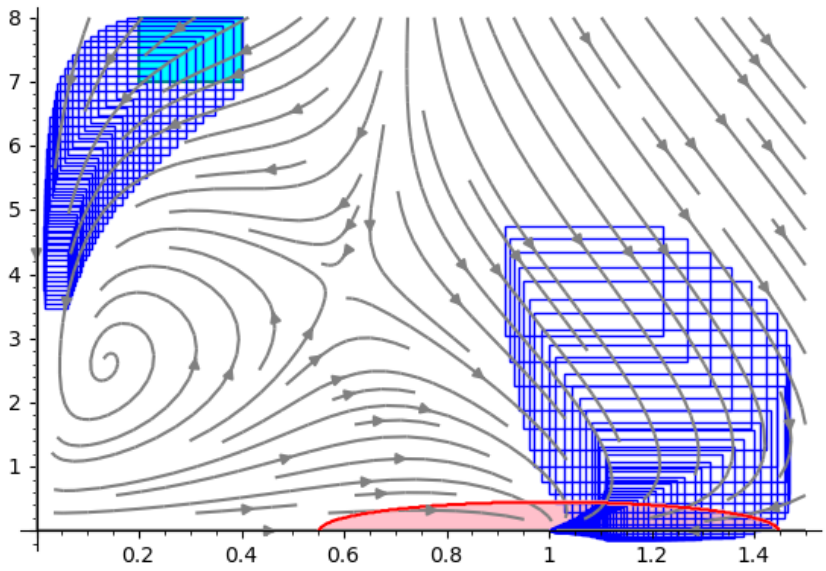


Figure 6.8: Applying jump to flowpipe.

A flowpipe showing the jump in the dynamics of the ecosystem corresponding to one particular time of context introduction is shown in Section 6.5. Applying our monitoring algorithm for a duration of 1 time units gives a signal $(([0,1],\mathbf{T}))$ verifying that φ_4 holds for at least 1 time unit. The ability to explore such hypothetical properties illustrates the advantages of performing computational experiments in $\mathcal{LBUC}$ since it would be difficult to test this property experimentally without significantly impacting the ecosystem under observation.

Whilst these examples all feature bounded time properties, their unbounded time counterparts are also interesting to determine the long term behaviour of the system. If we monitor the property P for 5 seconds starting from $\mathcal{M}_2$ we obtain the signal,

$$s_P = (([0, 0.802], \mathbf{F}), ([2.695, 5], \mathbf{T}))$$

and we are able to verify that P is an invariant of the system by applying the Liu, Zhan, and Zhao [242] verification procedure which reduced the invariance checking for P to a quantifier elimination problem in 2 variables which QEPCAD [78] was able to verify in 374 ms. Thus by applying Corollary 6.3.5 we are able to derive an unbounded signal for P,

$$s_P^\infty = (([0, 0.802], \mathbf{F}), ([2.695, \infty), \mathbf{T}))$$

from which we may derive the signals $\mathcal{G}_{[0,\infty)}\, s_P^\infty = s_P^\infty$ and

$$\mathcal{F}_{[0,5]}(\mathcal{G}_{[0,\infty)}\, s_P^\infty) = (([0, \infty), \mathbf{T})) = \mathcal{F}_{[0,\infty)}(\mathcal{G}_{[0,\infty)}\, s_P^\infty)$$

which suffices to verify the mixed bounded/unbounded-time property $\mathcal{F}_{[0,5]}\,\mathcal{G}_{[0,\infty)}\,P$ or the weaker doubly unbounded-time property $\mathcal{F}_{[0,\infty)}\,\mathcal{G}_{[0,\infty)}\,P$. Conversely, if we consider Q and this same initial condition, monitoring Q for 5 time units gives signal $(([0,5],\mathbf{F}))$ and as we know that $\mathcal{F}_{[0,5]}\,P$ is true, P is an invariant, and P and Q are mutually exclusive[4], applying Proposition 6.3.4 tells us that $(([0,\infty),\mathbf{F}))$ is also a signal for Q, and hence also for $\mathcal{F}_{[0,\infty)}\,Q$, allowing us to refute the unbounded time property.

We are also able to verify the unbounded time contextual property $\mathcal{G}_{[0,\infty)}\,\varphi_3 \equiv \mathcal{G}_{[0,\infty)}([0,2]\,\textsc{Lobster} \triangleright \varphi_2)$ since we note that the initial set $\Phi \triangleq [0.9,1.1]\,\textsc{Whelk}$ is an invariant of the system[5] and we know $(([0,5],\mathbf{T}))$ is a signal for φ_3, so by Proposition 6.3.4, $(([0,\infty),\mathbf{T}))$ is a signal for φ_3, and hence $(([0,\infty),\mathbf{T}))$ is a signal for $\mathcal{G}_{[0,\infty)}\,\varphi_3$.

[4]If need be this can also be verified in an automated manner by applying quantifier elimination to show $\exists W, \exists L(P \wedge Q)$ is false.

[5]This can easily be seen since $[\textsc{Lobsters}] = 0$ is is a nullcline and $\mathrm{d}[\textsc{Whelk}]/\mathrm{d}t > 0$ at $[\textsc{Whelk}] = 0.9$ and $\mathrm{d}[\textsc{Whelk}]/\mathrm{d}t < 0$ at $[\textsc{Whelk}] = 1.1$.

Chapter 7

Masks

A limitation of conventional signal monitoring algorithms is that their bottom-up nature — first monitoring signals for atomic propositions which are propagated upwards to derive signals for complex propositions — does not take into account the overall verification goal or our existing progress towards it during the monitoring of each atomic proposition. This can lead to excess work monitoring regions of time which give us no additional information about the property at hand. This limitation is especially important in the case of verified monitoring over Flow* flowpipes, given the expensive symbolic operations which are necessary to extract precise interval bounds on a region of the flowpipe, and in the case of context operators, which require repeated Flow* verified integration runs to monitor a signal over a given region. In this chapter we resolve these issues by introducing *masked monitoring*, which supplements the normal bottom-up monitoring process with the top-down computation of *masks*, a special type of signal which indicates the regions of interest for a given proposition. These masks direct the monitoring of atomic propositions to just the part of the time-domain relevant to the overall verification problem. Overall, this enables a more *property-directed* verification process, where the monitoring process for atomic propositions is driven by the knowledge required to move towards our overall verification goals. As we will see, this can substantially reduce the monitoring cost for complex logical properties and can reduce the monitoring costs for some nested contextual properties by an order of magnitude.

In Section 7.1 we introduce the basic definitions for masks and give some examples. In Section 7.2 we define the context in which an atomic proposition may occur within the overall STL monitoring process. In Section 7.3 we show how masks can be computed to check different types of complex logical propositions. Finally, in Section 7.6 we present some benchmarks demonstrating the effect of masks on signal monitoring performance.

Some of the results from this chapter have been published as part of the paper [360].

7.1 Basic Notions

Firstly we introduce *masks* as follows:

Definition 7.1.1. A *mask* is a finite sequence $m = (I_j)_j$ of disjoint intervals I_j. We refer to these intervals as the *regions of interest under the mask* m.

We can interpret a mask m as a Boolean signal $m : \mathbb{R}_{\geq 0} \to \mathbb{B}$ such that $m(x) = \mathbf{T}$ iff $x \in \bigcup_j I_j$. Such a mask represents the region of time for which we wish to monitor a given proposition. Since for monitoring soundness we only need to over-approximate these regions of interest, in a practical implementation we may restrict ourselves to masks whose components are all closed intervals $I_j \triangleq [a_j, b_j] \in \mathbb{IR}$ (using e.g. floating point endpoints) and consistently round outwards. We will however sometimes use other types of interval endpoints in what follows in order to state crisp results.

We can apply a mask to an existing signal, erasing any truth values that lie outside the mask.

Definition 7.1.2. Given a signal s and a mask m, the *masked signal of* s *by* m is the signal $s|_m$ defined by

$$s|_m(t) \triangleq \begin{cases} s(t) & \text{if } m(t) = \mathbf{T} \\ \mathbf{U} & \text{otherwise.} \end{cases}$$

7.1.1 Examples of Masking

Before laying out rules for using and computing masks, we will illustrate their use in two different examples, demonstrating the importance of the temporal and logical context of a proposition within a STL formula.

Example 7.1.3. Suppose we want to monitor the property $\varphi \triangleq \mathcal{F}_{[5,6]}\, \psi$ for 2 seconds (that is, over the time-domain $I \triangleq [0, 2]$). This would naively require computing a signal for ψ over 8 seconds, despite the fact that φ only looks at the behaviour of ψ between 5 and 6 seconds in the future — that is, within the absolute time-domain

$$I + [5, 6] = [0, 2] + [5, 6] = [5, 8].$$

This means that in checking φ it is sufficient to compute a signal for ψ under the mask $m \triangleq ([5, 8])$, allowing us to ignore more than half of the time-domain.

Example 7.1.4. Suppose we want to monitor the property $\varphi \triangleq \psi \wedge \sigma$ for 5 seconds (that is, over the time-domain $I \triangleq [0,5]$). This would normally require computing signals for ψ and σ over the whole time domain I. However, if we have already computed a signal for ψ such as

$$s_\psi \triangleq (([0,1], \mathbf{T}), ([2,4], \mathbf{F}))$$

then it is evident that computing a signal for φ only depends on the truth value of σ on the intervals $[0,2)$ and $(4,5]$. It thus suffices to compute a signal for σ under the mask $m \triangleq ([0,2), (4,5])$. This demonstrates how masks can enable a form of *temporal short-circuiting*.

Whilst in both of the above examples the required masks are quite simple, in general they quickly become much more complex depending on what signals we have already computed for other parts of the property (as in Example 7.1.4) and the position of the current proposition in a larger property. Later in this section we will see how the reasoning in these two examples can be generalised and combined to build up masks for arbitrary properties in a compositional manner.

7.1.2 Operations on Masks

We next need to define the operations with which masks can be build. Firstly, masks inherit all of the normal logical operations on Boolean signals [253]. In particular, given masks $m_I = (I_k)_k$ and $m_J = (J_l)_l$ we have that $m_I \wedge m_J = (I_k \cap J_l)_{l,k}$, and we write the negation of a mask m as $\neg m$.

We will also need the temporal operators $\mathcal{P}_J$ *(past)* and $\mathcal{H}_J$ *(historically)* defined on masks by,

Definition 7.1.5. Given a mask $m_I = (I_k)_k$ and an interval $J = [a,b]$, the *past mask* is defined by,

$$\mathcal{P}_J\, m_I \triangleq (I_k + J)_k$$

whilst the *historically mask* is defined by,

$$\mathcal{H}_J\, m_I \triangleq \neg\, \mathcal{P}_J\, (\neg m) = ((I_k + a) \cap (I_k + b))_k\,.$$

The operator $\mathcal{P}_J$ is the time-reversed dual of the signal operator $\mathcal{F}_J$ and shifts a mask forward and outwards in time by the interval J, whilst the operator $\mathcal{H}_J$ is the time-reversed dual of $\mathcal{G}_J$ and shifts a mask forward and inwards.

7.2 Monitoring Contexts

Before we specify how the masks for the monitoring algorithm should be computed, we must first formalise what is required of a mask for it to be used at a given stage of the monitoring algorithm. This motivates us to define *monitoring contexts* which capture our existing knowledge at each recursive step of the monitoring algorithm, by recording the position of an argument ψ within the STL operator currently being monitored and the context given by the signals s we have already computed for any other arguments.

Definition 7.2.1. A *monitoring context* is defined similarly to a STL formula except with subformulae replaced by concrete signals s and, in exactly one place, a *hole* $[\cdot]$. That is, a monitoring context is defined according to the grammar

$$\begin{aligned}\mathcal{S}([\cdot]) \;::=\; [\cdot] \;\mid\; s \vee \mathcal{S}([\cdot]) \;\mid\; s \wedge \mathcal{S}([\cdot]) \;\mid\; \neg\, \mathcal{S}([\cdot]) \\ \mid\; \mathcal{F}_I(\mathcal{S}([\cdot])) \;\mid\; \mathcal{G}_I(\mathcal{S}([\cdot])) \;\mid\; s\, \mathcal{U}_I\, \mathcal{S}([\cdot])\,.\end{aligned}$$

A monitoring context $\mathcal{S}([\cdot])$ is a *monitoring context of the subformula ψ of a STL formula φ*, if $\mathcal{S}([\cdot])$ has the same structure as φ except that, in the place of each atomic proposition ρ of φ, $\mathcal{S}([\cdot])$ has a signal s_ρ which is a signal for ρ, and the hole $[\cdot]$ in place of ψ.

Given a signal s, we can evaluate a monitoring context $\mathcal{S}([\cdot])$ to give a signal $\mathcal{S}(s)$ by substituting s in the place of the hole $[\cdot]$ and following the usual rules for combining signals. This means that a monitoring context captures how the signal for the overall formula depends on the signal for the proposition which remains to be monitored.

Example 7.2.2. In applying a bottom-up STL monitoring algorithm to obtain a signal s_φ for the property $\varphi \triangleq \mathcal{F}_{[5,6]}\, \psi$ (Example 7.1.3), we must first monitor a signal s_ψ for ψ. The monitoring context $\mathcal{S}([\cdot]) \triangleq \mathcal{F}_{[5,6]}([\cdot])$ for the subformula ψ in φ captures the role of the signal s_ψ in monitoring φ given that s_ψ contributes to the overall signal via the signal computation

$$s_\varphi = \mathcal{S}(s_\psi) = \mathcal{F}_{[5,6]}\, \psi.$$

Example 7.2.3. In order to monitor a signal s_φ for the property $\varphi \triangleq \psi \wedge \sigma$ (Example 7.1.4), we first monitor a signal s_ψ for ψ before monitoring a signal s_σ for σ. Once we have obtained a signal s_ψ for ψ, the monitoring context $\mathcal{S}([\cdot]) = s_\psi \wedge [\cdot]$ for the subformula σ in φ captures the role of σ in the remaining task of monitoring φ given that s_σ contributes to the overall signal via the signal computation

$$s_\varphi = \mathcal{S}(s_\sigma) = s_\varphi \wedge s_\sigma.$$

Remark 7.2.4 (Context Operators v.s. Monitoring Contexts). There is an unfortunate overloading of terminology between context operators in $\mathcal{LBUC}$ and the monitoring contexts we introduce in this section. However, we can see that both are special cases of the concept of contexts in programming languages and concurrency theory since context operators substitute a system into a larger bond-calculus model (that is, into a bond-model expression of form $(\Pi, \mathcal{A}) \,\|[\cdot])$, whilst monitoring contexts substitute a signal into an execution frame of our compositional STL monitoring algorithm.

The concept of a monitoring context will be key to defining and establishing the correctness of our masked monitoring algorithm, since the context of a subformula within the overall formula is exactly what determines the region of the time domain (that is, the mask) over which the subformula should be monitored. To this end, we first define when a mask is sufficient for monitoring in a monitoring context.

Definition 7.2.5. A mask m is *sufficient for a monitoring context* $\mathcal{S}([\cdot])$ *under mask* n, if for any signal s we have that

$$\mathcal{S}(s)|_n = \mathcal{S}(s|_m)|_n.$$

That is, a mask is sufficient if signals masked by it are *just as good as* unmasked signals for monitoring the overall formula.

Example 7.2.6. Building on Examples 7.1.3 and 7.2.2 we see that to monitor the property $\varphi \triangleq \mathcal{F}_{[5,6]}\,\psi$ for 2 seconds (that is, over the overall mask $n = ([0,2])$), we need to monitor ψ in the context $\mathcal{S}([\cdot]) \triangleq \mathcal{F}_{[5,6]}([\cdot])$. But then we note the mask $m = ([5,8])$ is sufficient under context $\mathcal{S}([\cdot])$ given the outer mask n since, for any signal s,

$$\begin{aligned}
\Big(\mathcal{S}\Big(s|_m\Big)\Big)|_n &= \Big(\mathcal{F}_{[5,6]}\Big(s|_{([5,8])}\Big)\Big)|_{([0,2])} \\
&= \Big(\Big(\mathcal{F}_{[5,6]}\; s\Big)|_{([5,8]\dot{-}[5,6])}\Big)|_{([0,2])} \\
&= \Big(\Big(\mathcal{F}_{[5,6]}\; s\Big)|_{([0,2])}\Big)|_{([0,2])} \\
&= \Big(\mathcal{F}_{[5,6]}\; s\Big)|_{([0,2])} \\
&= \Big(\mathcal{S}\Big(s\Big)\Big)|_n.
\end{aligned}$$

Example 7.2.7. Building on Examples 7.1.4 and 7.2.3 we see that, having already monitored the signal $s_\psi \triangleq (([0,1], \mathbf{T}), ([2,4], \mathbf{F}))$ for ψ, in order to monitor the property $\varphi \triangleq \psi \wedge \sigma$ for 5 seconds (that is, over the overall mask $n = ([0,5])$), we need to monitor ψ in the context $\mathcal{S}_\sigma([\cdot]) = s_\psi \wedge [\cdot]$. Then we note that the mask $m \triangleq ([0,2),(4,5])$. is

sufficient under context $\mathcal{S}([\cdot])$ given the outer mask n since, for any signal s,

$$\begin{aligned}
\Big(\mathcal{S}\Big(s|_m\Big)\Big)|_n &= \Big(s_\psi \wedge s|_{([0,2),(4,5])}\Big)\Big)|_{([0,5])} \\
&= \Big(\Big(s_\psi \wedge s|_{([2,4])}\Big) \wedge s|_{([0,2),(4,5])}\Big)\Big)|_{([0,5])} \\
&= \Big(s_\psi \wedge \Big(s|_{([2,4])} \wedge s|_{([0,2),(4,5])}\Big)\Big)\Big)|_{([0,5])} \\
&= \Big(s_\psi \wedge s|_{([0,5])}\Big)\Big)|_{([0,5])} \\
&= \Big(s_\psi \wedge s\Big)\Big)|_{([0,5])} \\
&= \Big(\mathcal{S}(s)\Big)|_n.
\end{aligned}$$

In order to establish that we are not monitoring a given subformula on irrelevant regions of the time domain, we also define when a mask is as small as possible for monitoring in a given context.

Definition 7.2.8. A mask m is the *optimal mask in monitoring context* $\mathcal{S}([\cdot])$ *under mask* n if it is the smallest sufficient mask in context $\mathcal{S}([\cdot])$ under mask n with respect to the pointwise truth ordering $\leq$.

It follows directly that the mask defined above is unique (for a given monitoring context and overall mask), allowing us to talk about *the mask* for a given monitoring context.

7.3 Monitoring Under a Mask: Complex Propositions

We are now ready to detail how our masked monitoring algorithm deals with complex propositions, by introducing suitable masks for each temporal and logical context. In each case we prove that these masks are sufficient (and optimal) for the relevant context, which collectively shows the correctness of masked monitoring.

7.3.1 Negation

Suppose we want to monitor a negation $\neg\varphi$ under mask m, then this is equivalent to monitoring φ under mask m and then negating the resulting signal.

Proposition 7.3.1. *The mask m is itself sufficient and optimal for monitoring under m in the monitoring context*

$$\mathcal{S}([\cdot]) = \neg[\cdot].$$

Proof.

Sufficiency Take any signal s and any time $t \in \mathbb{R}_{\geq 0}$. Then if $m(t)$,

$$(\neg(s|_m))|_m(t) = (\neg(s|_m))(t) = \neg(s|_m)(t) = \neg s(t) = (\neg s)|_m(t)$$

and if $\neg m(t)$,

$$(\neg(s|_m))|_m(t) = \mathbf{U} = (\neg s)|_m(t)$$

Optimality Let m' be any sufficient signal. Take the signal $s(t) \equiv \mathbf{F}$ and choose any time $t \in \mathbb{R}_{\geq 0}$ such that $m(t) = \mathbf{T}$. Then, by the sufficiency of m' we must have

$$\mathbf{T} = \neg s(t) = (\neg(s|_{m'}))(t) = \begin{cases} \mathbf{T} & \text{if } m'(t) = \mathbf{F} \\ \mathbf{U} & \text{otherwise} \end{cases}$$

and so we must have $m'(t) = \mathbf{T}$ showing $m \Rightarrow m'$. □

7.3.2 Eventually ($\mathcal{F}_{[a,b]}\,\varphi$) and Globally ($\mathcal{G}_{[a,b]}\,\varphi$)

Suppose we want to monitor the property $\mathcal{F}_{[a,b]}\,\varphi$ or $\mathcal{G}_{[a,b]}\,\varphi$, under mask $m = (I_j)_j$. In this case we should monitor φ under the past mask

$$\mathcal{P}_{[a,b]}\,m = (I_j + [a,b])_j.$$

because the truth of φ at time t could determine the truth of either $\mathcal{F}_{[a,b]}\,\varphi$ or $\mathcal{G}_{[a,b]}\,\varphi$ at any point between a and b seconds ago (in the former case by witnessing its truth, and in the latter case, by witnessing its falsehood) — this generalises the reasoning given in Example 7.1.3.

Proposition 7.3.2. *Given a context*

$$\mathcal{S}([\cdot]) = \mathcal{F}_{[a,b]}[\cdot] \qquad \text{or} \qquad \mathcal{S}([\cdot]) = \mathcal{G}_{[a,b]}[\cdot]$$

under the overall mask m, in each case the mask $\mathcal{P}_{[a,b]}\,m$ is sufficient and optimal for $\mathcal{S}([\cdot])$.

Proof. Here we just prove sufficiency and optimality for $\mathcal{S}([\cdot]) = \mathcal{F}_{[a,b]}[\cdot]$ the results for $\mathcal{S}([\cdot]) = \mathcal{G}_{[a,b]}[\cdot]$ follow since $\mathcal{G}_{[a,b]}\,\varphi \equiv \neg\,\mathcal{F}_{[a,b]}\,\neg\varphi$.

Sufficiency For sufficiency we need to show that

$$\mathcal{S}\Big(s|_{\mathcal{P}_{[a,b]}\, m}\Big)(t) = \mathcal{F}_{[a,b]}\Big(s|_{\mathcal{P}_{[a,b]}\, m}\Big)(t) = \mathcal{F}_{[a,b]}(s)(t) = \mathcal{S}(s)(t)$$

for any three-valued signal s and time point t such that $m(t) = \mathbf{T}$. We do this by showing that $\mathcal{P}_{[a,b]}\, m(t') = \mathbf{T}$ and hence $s|_{\mathcal{P}_{[a,b]}\, m}(t') = s(t')$ at each of the future time points $t' \in t + [a, b]$ to which both of the above $\mathcal{F}_{[a,b]}$ operators refer. This holds by contrapositive since if we had some $t' \in t + [a, b]$ for which $\mathcal{P}_{[a,b]} = \mathbf{F}$, then we would have $m(t'') = \mathbf{F}$ for all $t'' \in t' - [a, b]$ and, in particular, $m(t) = \mathbf{F}$.

Optimality Suppose m' is any mask such that $m' < \mathcal{P}_{[a,b]}\, m$. Then we have some t_0 for which $m'(t_0) = \mathbf{F}$ whilst $\mathcal{P}_{[a,b]}\, m(t_0) = \mathbf{T}$ and hence we must have some $t_0' \in t_0 - [a, b]$ such that $m(t_0') = \mathbf{T}$. But then if we take the signal

$$s(t) = \begin{cases} \mathbf{T} & \text{if } t = t_0 \\ \mathbf{U} & \text{otherwise} \end{cases}$$

we see that $\mathcal{F}_{[a,b]}\, s(t_0') = \mathbf{T}$ whilst $\mathcal{F}_{[a,b]}\, s|_{m'}(t_0') = \mathbf{U}$ and hence m' is not sufficient, proving the optimality of $\mathcal{P}_{[a,b]}\, m$. □

7.3.3 Conjunctions and Disjunctions

Suppose we want to monitor a conjunction $\varphi \wedge \psi$ under mask m. We should first monitor φ under the mask m to give the signal s. Then, generalising Example 7.1.4, we can use the signal s to generate a mask $m_s^\wedge$, the *and-mask of* s.

Definition 7.3.3. Given a three-valued signal $s = (I_j, s_j)_j$, the *and-mask of* s is the mask $m_s^\wedge$ defined by $m_s^\wedge(t) = \mathbf{T}$ iff $s(t) \in \{\mathbf{T}, \mathbf{U}\}$ so

$$m_s^\wedge = \bigwedge_{s_j = \mathbf{F}} m_j$$

where $m_j \triangleq \left(C_j^{(\ell)}, C_j^{(u)}\right)$ is the mask consisting of the two interval complements $C_j^{(\ell)}, C_j^{(u)}$ of I_j in $\mathbb{R}_{\geq 0}$.

If this mask turns out to be empty (i.e. if $\neg s(t) = \mathbf{T} = m_s^\wedge(t)$ for all $x \in m$), then we can stop and conclude s is a signal for $\varphi \wedge \psi$ under m. Otherwise, we monitor ψ under the mask $m_s^\wedge$ giving a signal w, and hence the signal $s \wedge w$ for $\varphi \wedge \psi$ under m.

We see that the and-mask $m_s^\wedge$ is optimal and sufficient for the context $\mathcal{S}([\cdot]) = s \wedge [\cdot]$.

Proposition 7.3.4. *Given a monitoring context, $\mathcal{S}([\cdot]) = s \wedge [\cdot]$ the and-signal $m_s^\wedge$ is sufficient and optimal for this context under the mask m.*

Proof.

Sufficiency Take any signal w and any time $t \in \mathbb{R}_{\geq 0}$ such that $m(t) = \mathbf{T}$. Then if $s(t) = \mathbf{F}$,

$$\mathcal{S}(w|_{m_s^\wedge})(t) = \mathbf{F} \wedge w|_{m_s^\wedge}(t) = \mathbf{F} = \mathbf{F} \wedge w(t) = \mathcal{S}(w)(t)$$

whilst if $s(t) \in \{\mathbf{T}, \mathbf{U}\}$, then $m_s^\wedge(t) = \mathbf{T}$ and hence

$$\mathcal{S}\Big(w|_{m_s^\wedge}\Big)(t) = s(t) \wedge w|_{m_s^\wedge}(t) = s(t) \wedge w(t) = \mathcal{S}(w)(t)$$

showing $m_s^\wedge$ is sufficient.

Optimality Let m' be any mask such that $m' < m_s^\wedge$. Then there must be some time t_0 such that $m'(t_0) = \mathbf{F}$ and $m_s^\wedge(t_0) = \mathbf{T}$ and hence $s(t_0) \in \{\mathbf{T}, \mathbf{U}\}$. But then if we pick the signal $w(t) \equiv \mathbf{F}$, we see that

$$\mathcal{S}(w|_{m'}) = s(t_0) \wedge (s|_{m'})(t_0) = s(t_0) \wedge \mathbf{U} = \mathbf{U}$$

whilst

$$\mathcal{S}\Big(w|_{m_s^\wedge}\Big)(t) = s(t_0) \wedge w|_{m_s^\wedge}(t_0) = s(t_0) \wedge \mathbf{F} = \mathbf{F}$$

and hence m' is not sufficient, proving that $m_s^\wedge$ must be optimal. □

We treat disjunctions similarly and can see that the *or-mask* $m_s^\vee$ defined by $m_s^\vee(t) = \bigwedge_{s_j = \mathbf{T}} m_j$ is an optimal and sufficient mask for the monitoring context $\mathcal{S}([\cdot]) = s \vee [\cdot]$.

7.3.4 Until ($\varphi \, \mathcal{U}_{[a,b]} \, \psi$)

Finally, suppose we wish to monitor the signal for the property $\varphi \, \mathcal{U}_{[a,b]} \, \psi$ under the mask m. As in Section 6.1, we will compute the signal for $\varphi \, \mathcal{U}_{[a,b]} \, \psi$ based on signals for φ and ψ using Eq. (6.1), however we now need to monitor φ and ψ under appropriate masks. We start by monitoring φ under the mask $m \vee \mathcal{P}_{[a,b]} \, m$ (taking into account the two places in which it appears in Eq. (6.1)). Then we could find a suitable mask for ψ by applying the above rules for $\vee$, $\wedge$, and $\mathcal{F}_{[a,b]}$ to Eq. (6.1). However, it turns out that this mask may be computed directly using the historically operator.

Proposition 7.3.5. *Given an unitary mask m we have that*

$$\mathcal{H}_{[0,a]} \, m = m \wedge \mathcal{P}_{[a,b]} \, m.$$

Proof. Suppose $\mathbf{T} = m \wedge \mathcal{P}_{[a,b]}(m)(t)$. Then $\mathbf{T} = m(t)$ and for some $t' \in t + [a,b]$, $m(t') = \mathbf{T}$. But then for any $t'' \in t + [0,a]$, we must have $m(t'') = \mathbf{T}$ since m is unitary, showing $\mathcal{H}_{[0,a]}\, m(t) = \mathbf{T}$.

Conversely, suppose $\mathcal{H}_{[0,a]}\, m(t) = \mathbf{T}$. The for all $t' \in t + [0,a]$ we have that $m(t) = \mathbf{T}$. In particular $m(t) = \mathbf{T}$ and $m(a+t) = \mathbf{T}$ showing that $m \wedge \mathcal{P}_{[a,b]}\, m = \mathbf{T}$. □

And we can see historically distributes over disjoint unitary masks.

Proposition 7.3.6. *Given disjoint, closed unitary masks m, n and an interval $J = [a,b]$, we have that*

$$\mathcal{H}_J(m \vee n) = \mathcal{H}_J\, m \vee \mathcal{H}_J\, n.$$

Proof. Take any time t. If $\mathcal{H}_J\, m(t)$ or $\mathcal{H}_J\, n(t)$ then in either case we clearly have that $\mathcal{H}_J(m \vee n)(t)$.

Conversely, if $\mathcal{H}_J(m \vee n)(t)$ then for any $t' \in J$, $m(t') = \mathbf{T}$ or $n(t') = \mathbf{T}$. If neither of these signals is true both endpoints of $t + J$ (in which case we would have $\mathcal{H}_J\, m \vee \mathcal{H}_J\, n(t') = \mathbf{T}$ by unitarity) we can suppose w.l.o.g. that $m(t+a) = \mathbf{T}$ and $n(t+b) = \mathbf{T}$. Then let

$$l = \sup\Big\{\, t' \in t + J : m(t') = \mathbf{T} \,\Big\} \quad \text{and} \quad u = \inf\Big\{\, t' \in t + J : n(t') = \mathbf{T} \,\Big\}.$$

Suppose $l \leq u$. Then if we let $c = \frac{l+u}{2}$, we must have either $m(c) = \mathbf{T}$ or $n(c) = \mathbf{T}$. W.l.o.g. assuming the former, we must have $l = c = \frac{l+u}{2} \leq u$ and hence $l = u$, a contradiction. Therefore, we must have $l > u$. But then we have $m = n$ by unitarity and disjointness, and hence $\mathcal{H}_J(m \vee n)(t) = \mathcal{H}_J\, m(t) = \mathcal{H}_J\, m \vee \mathcal{H}_J\, n(t) = \mathbf{T}$. □

Proposition 7.3.7. *If m is any mask such that $m = \bigvee_j m_j$ (m_j disjoint closed unitary signals) then*

$$\mathcal{H}_{[0,a]}\, m = \bigvee_j m_j \wedge \mathcal{P}_{[a,b]}\, m_j.$$

Proof. Follows by Proposition 7.3.5 and Proposition 7.3.6. □

Together these give us the mask for the until operator.

Proposition 7.3.8. *The mask*

$$m_s^{\mathcal{U}_a} \triangleq \mathcal{H}_{[0,a]}\, (m_s^{\wedge})$$

is optimal and sufficient for monitoring context $\mathcal{C}([\cdot]) = s\, \mathcal{U}_{[a,b]}\, [\cdot]$.

Proof. For both proofs we will use the decomposition of w into disjoint closed components

$$w = \bigvee_{j,k} w_{j,k}.$$

Sufficiency For any signal s we have

$$w\,\mathcal{U}_{[a,b]}\ s|_m = \bigvee_{j,k} w_{j,k} \wedge \mathcal{F}_{[a,b]}\left(w_{j,k} \wedge s|_m\right).$$

Take any j, k and $t \in \mathbb{R}_{\geq 0}$. If $w_{j,k}(t) = \mathbf{F}$ then

$$w_{j,k} \wedge \mathcal{F}_{[a,b]}\left(w_{j,k} \wedge s|_m\right)(t) = \mathbf{F} = w_{j,k} \wedge \mathcal{F}_{[a,b]}\left(w_{j,k} \wedge s\right)(t)$$

and we are done. If, on the other hand, $w_{j,k}(t) \in \{\mathbf{T}, \mathbf{U}\}$ and we take any $t' \in t + [a, b]$, such that

$$m(t') = \mathcal{H}_{[0,a]}\, m^{\wedge}_w(t') = \mathbf{F},$$

then we have some $t'' \in t' - [0, a]$ such that $m^{\wedge}_w(t'') = \mathbf{F}$ and hence $w(t'') = \mathbf{F}$. But then if we had $w_{j,k}(t') \in \{\mathbf{T}, \mathbf{U}\}$, since we know $w_{j,k}(t) \in \{\mathbf{T}, \mathbf{U}\}$ and $t'' \in [t, t']$, by unitarity we would have $w_{j,k}(t'') \in \{\mathbf{T}, \mathbf{U}\}$, a contradiction. Then, instead we must have $w_{j,k}(t') = \mathbf{F}$, and hence

$$w_{j,k} \wedge s|_m(t') = \mathbf{F} = w_{j,k} \wedge s(t').$$

Therefore, $w_{j,k} \wedge s|_m(t') = w_{j,k} \wedge s(t')$ for any $t' \in t + [a, b]$ and hence

$$w\,\mathcal{U}_{[a,b]}\ s|_m = w\,\mathcal{U}_{[a,b]}\ s$$

giving sufficiency.

Optimality Consider any mask m' such that $m' < m$. Then there must be some $t_0 \geq 0$ such that $m(t_0) = \mathbf{T}$ whilst $m'(t_0) = \mathbf{F}$. Since

$$m(t_0) = \mathcal{H}_{[0,a]}\left(m^{\wedge}_w\right)(t_0),$$

we must have that $t_0 \geq a$ and for all $t' \in t_0$ we must have $m^{\wedge}_w(t') = \mathbf{T}$ and hence $w(t') \in \{\mathbf{T}, \mathbf{U}\}$. Then, by the nature of the decomposition of w, for any j, k, we must have either

$$w_{j,k}(t') \in \{\mathbf{T}, \mathbf{U}\} \quad \text{for all } t' \in t_0 - [0, a] \tag{7.1}$$

or

$$w_{j,k}(t') = \mathbf{F} \qquad \text{for all } t' \in t_0 - [0, a] \tag{7.2}$$

Now, since we know $t_0 \geq a$, we may take $t \triangleq t_0 - a$ and define the signal s as

$$s(t') \equiv \mathbf{F}.$$

Then, in the case we have Eq. (7.1), we see

$$w_{j,k} \wedge \mathcal{F}_{[a,b]} \left(w_{j,k} \wedge s|_m \right)(t) = \mathbf{F}$$

since $s|_m(t_0) = \mathbf{F}$ and for any $t' \in t + [a, b]$ for which $s_m(t') \neq \mathbf{F}$ we must have some $t'' \in [t' - a, t'] \subseteq [t, t']$ such that $w(t'') = w_{j,k}(t'') = \mathbf{F}$. In this case we have that,

$$w_{j,k} \wedge \mathcal{F}_{[a,b]} \left(w_{j,k} \wedge s|_{m'} \right)(t) = \mathbf{U}$$

since $s|_{m'}(t_0) = \mathbf{U}$ and $w_{j,k}(t') \in \{\mathbf{T}, \mathbf{U}\}$ for all $t' \in t_0 - [0, a] = [t, t_0]$.

On the other hand, in the case we have Eq. (7.2), we have

$$w_{j,k} \wedge \mathcal{F}_{[a,b]} \left(w_{j,k} \wedge s|_m \right)(t) = \mathbf{F} = w_{j,k} \wedge \mathcal{F}_{[a,b]} \left(w_{j,k} \wedge s|_{m'} \right)(t),$$

whence we conclude $w \; \mathcal{U}_{[a,b]} \; s|_m(t) = \mathbf{F}$ whilst $w \; \mathcal{U}_{[a,b]} \; s|_{m'}(t) = \mathbf{U}$, showing m' is not sufficient, and thus completing the proof. □

7.4 Monitoring Under a Mask: Atomic Propositions

Once we have determined a mask $m = (I_j)_j$ for a given atomic proposition ρ given its context in the monitoring process, we then aim to directly monitor a signal s for φ *under the mask* m. This means that we only care about the value of $s(t)$ at time points t for which $m(t)$ is true, and so can increase the efficiency of monitoring by avoiding work associated with time points outside of the mask. Whilst there is no way to save Flow* from having to generate the flowpipes for these time points (since they may be required for determining the future evolution of the system), we can avoid the effort associated with every subsequent step of the monitoring process.

We do this by modifying how we carry out monitoring of ρ (via $H_p^{(k)}$) on each flowpipe segment (Section 6.1.3) over its associated time domain $T_k = [t_k, t_{k+1}]$ as follows:

- if $m \wedge (T_k) = \varnothing$ then we set $s(t) = \mathbf{U}$ for all $t \in T_k$ and avoid monitoring over this segment;
- otherwise, we restrict the time domain to the interval $T_k' = \bigcup_j T_k \cap I_j$ and apply the normal monitoring process.

This immediately saves us from performing root finding on regions outside of the mask. Additionally, since we have already seen how the symbolic composition of the two halves of the flowpipe and between the flowpipe and the atomic propositions may be performed

on demand, these expensive operations may also be avoided outside of the mask. Thus masks allow us to direct the monitoring of each atomic proposition based on its context within a wider STL formula.

7.5 Masked Monitoring for $\mathcal{LBUC}$

We now apply the techniques in this section to a full masked monitoring algorithm for $\mathcal{LBUC}$ properties. We do this by defining functions $\texttt{signalM}(\varphi, (\Pi_0, \mathcal{A}), m)$ and $\texttt{signalMF}(\varphi, \Pi_f, \mathcal{A}, \mathbf{S}, m)$ which generalize the definition of `signal` and `signalF` to compute a signal of φ under a mask m.

Firstly, we define `signalM` in terms of `signalMF` as follows.

Definition 7.5.1. We may compute a $\texttt{signalM}(\varphi, (\Pi_0, \mathcal{A}), m)$ for a $\mathcal{LBUC}$ property φ given uncertain bond-calculus model $(\widehat{\Pi}_0, \widehat{\mathcal{A}})$ under mask $m = (T_j)_j$ by

$$\texttt{signalM}(\varphi, (\Pi_0, \mathcal{A}), m) \triangleq \texttt{signalMF}(\varphi, \Pi_f, \mathcal{A}, \mathbf{S}, m)$$

where Π_f is a flowpipe for the interval initial value problem

$$\frac{\mathrm{d}\widetilde{\Pi}}{\mathrm{d}t} \triangleq \frac{\mathrm{d}(\widetilde{\Pi}, \widehat{\mathcal{A}})}{\mathrm{d}t}; \quad \widetilde{\Pi}_0 = \Pi_0$$

with space domain $\mathbf{S}$ and time duration $\max\{\sup T_j\}_j + \texttt{duration}(\varphi)$.

Next, the definition of `signalMF` is given via the following recursive monitoring procedure based on the masks specified in Sections 7.3 and 7.4.

Atomic Propositions For an atomic proposition $\rho = p(\mathbf{x}) > 0$, the signal $\texttt{signalMF}(\rho, \Pi_f, \mathcal{A}, \mathbf{S}, m)$ is defined via the masked flowpipe monitoring algorithm in Section 7.4.

Logical and Temporal Operators We may handle the logical and temporal operators by applying the signal operators defined in Section 6.1.1 to signals computed under suitable

masks as given in Section 7.3. That is,

$$\begin{aligned}
\texttt{signalMF}(\neg\varphi, \Pi_f, \mathcal{A}, \mathbf{S}, m) &\triangleq \neg\texttt{signalMF}(\varphi, \Pi_f, \mathcal{A}, \mathbf{S}, m) \\
\texttt{signalMF}(\varphi \wedge \psi, \Pi_f, \mathcal{A}, \mathbf{S}, m) &\triangleq s_\varphi \wedge s_\psi \\
&\qquad \text{where} \begin{cases} s_\varphi \triangleq \texttt{signalMF}\big(\varphi, \Pi_f, \mathcal{A}, \mathbf{S}, m\big) \\ s_\psi \triangleq \texttt{signalMF}\big(\psi, \Pi_f, \mathcal{A}, \mathbf{S}, m^{\wedge}_{s_\varphi}\big) \end{cases} \\
\texttt{signalMF}(\varphi \vee \psi, \Pi_f, \mathcal{A}, \mathbf{S}, m) &\triangleq s_\varphi \vee s_\psi \\
&\qquad \text{where} \begin{cases} s_\varphi \triangleq \texttt{signalMF}\big(\varphi, \Pi_f, \mathcal{A}, \mathbf{S}, m\big) \\ s_\psi \triangleq \texttt{signalMF}\big(\psi, \Pi_f, \mathcal{A}, \mathbf{S}, m^{\vee}_{s_\varphi}\big) \end{cases} \\
\texttt{signalMF}(\mathcal{F}_K\, \varphi, \Pi_f, \mathcal{A}, \mathbf{S}, m) &\triangleq \mathcal{F}_K\Big(\texttt{signalMF}\big(\varphi, \Pi_f, \mathcal{A}, \mathbf{S}, \mathcal{P}_K\, m\big)\Big) \\
\texttt{signalMF}(\mathcal{G}_K\, \varphi, \Pi_f, \mathcal{A}, \mathbf{S}, m) &\triangleq \mathcal{G}_K(\texttt{signalMF}(\varphi, \Pi_f, \mathcal{A}, \mathbf{S}, \mathcal{P}_K\, m)) \\
\texttt{signalMF}(\varphi\, \mathcal{U}_{[a,b]}\, \psi, \Pi_f, \mathcal{A}, \mathbf{S}, m) &\triangleq s_\varphi\, \mathcal{U}_{[a,b]}\, s_\psi \\
&\qquad \text{where} \begin{cases} s_\varphi \triangleq \texttt{signalMF}\big(\varphi, \Pi_f, \mathcal{A}, \mathbf{S}, m\big) \\ s_\psi \triangleq \texttt{signalMF}\Big(\psi, \Pi_f, \mathcal{A}, \mathbf{S}, \mathcal{H}_{[0,a]}\big(m^{\wedge}_{s_\varphi}\big)\Big) \end{cases}
\end{aligned}$$

Context Operator Finally, we define the masked signal for the context operator $\mathcal{C} \triangleright \varphi$. Firstly, we note that for a unitary mask $m = (T)$, the signal $\texttt{signalMF}(\mathcal{C} \triangleright \varphi, \Pi_f, \mathcal{A}, (T))$ can be computed in the same manner set out in Section 6.2, except that we must take $s \triangleq \texttt{signalM}\,(\varphi, \Phi \parallel \mathcal{C}, (0))$ in the recursive step (Eq. (6.3)). For a general mask $m = (T_j)_j$, the signal can be computed by concatenating the signal for $\mathcal{C} \triangleright \varphi$ over each of the intervals T_j included in m. That is,

$$\texttt{signalMF}(\mathcal{C} \triangleright \varphi, \Pi_f, \mathcal{A}, \mathbf{S}, m) \triangleq \bigcup_j \texttt{signalMF}(\mathcal{C} \triangleright \varphi, \Pi_f, \mathcal{A}, \mathbf{S}, (T_j)).$$

7.6 Performance Evaluation

We will now investigate the impact of masks on the performance of our monitoring algorithm, by first looking at the overall impact of masks on core monitoring performance in the 9-dimensional genetic model from Section 6.4, and then look at how masked monitoring performance varies for different types of contextual properties in the Predator-Prey Role Reversal model from Section 6.5.

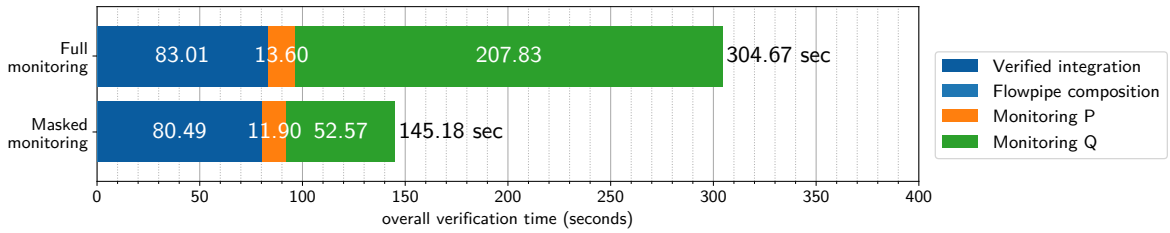


Figure 7.1: Combined Flow* verified integration and STL monitoring times in seconds, comparing the masked and unmasked variants of the monitoring algorithm.

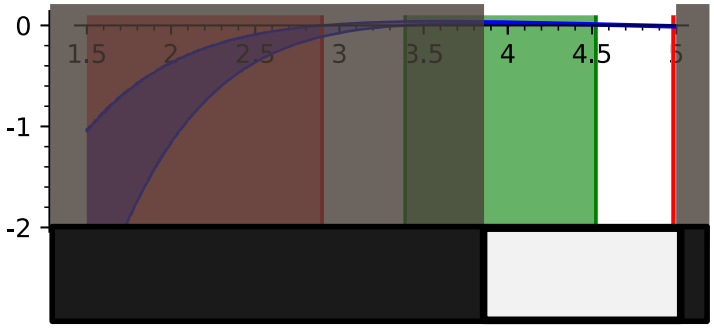


Figure 7.2: Q symbolic masked.

7.6.1 Core Monitoring Performance

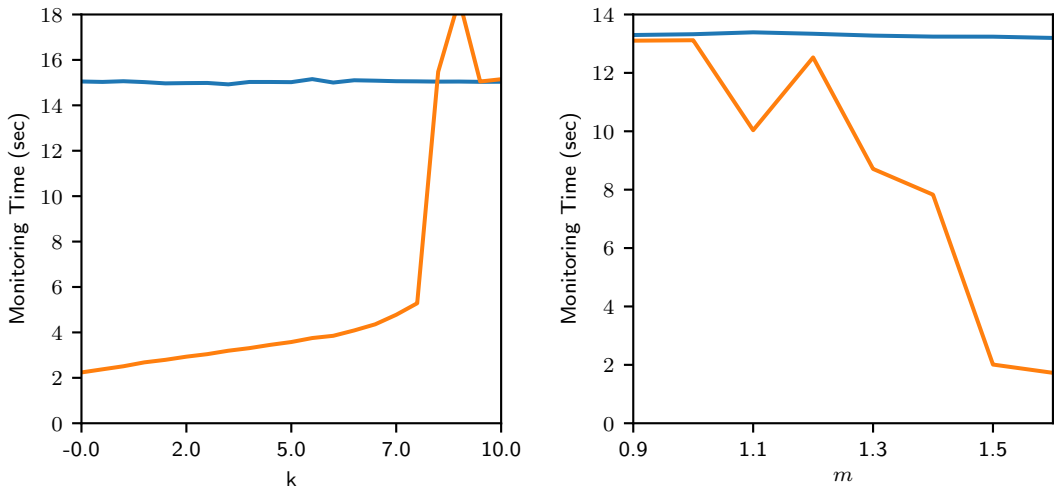
We can evaluate the performance impact of masks on the core monitoring algorithm in a challenging continuous system, by returning to the 9-dimensional genetic oscillator considered in Section 6.4, and the property

$$\varphi \triangleq \mathcal{G}_{[0,1.5]}\Big(P \vee \mathcal{G}_{[3,3.5]}(Q)\Big).$$

We should expect that the masked variant of our monitoring algorithm will reduce the total monitoring cost since both the time window of the inner globally operator and the information provided by monitoring P will restrict the mask under which we need to know Q. Indeed, the masked monitoring algorithm computes the signal for Q under the mask shown in Fig. 7.2. Thus we are able to produce the same signal for φ as before but reduce the monitoring time for Q by 65% as shown in Fig. 7.1.

7.6.2 Contextual Properties

Masking can have an even greater impact on monitoring cost for contextual properties, since in this case it is able to avoid not only monitoring costs for atomic propositions but the much more expensive cost of verified integration, which is required to determine the



(a) Monitoring time for ψ_k as k increases. (b) Monitoring time for σ_m as m increases.

Figure 7.3: The impact of formula structure on unmasked and masked monitoring times.

signal for the context operator, on regions outside of the mask. We can explore this cost by returning to the model of role-reversal in predator prey interactions from Section 6.5 and looking at how the monitoring cost varies for different forms of proposition requiring a contextual subproposition on a varying proportion of the time domain.

Firstly we consider the parametrized property

$$\psi_k \triangleq \mathcal{F}_{[10-k,10]}([0.05, 0.1]\,\mathrm{W}\,\textsc{helk} \triangleright \mathcal{G}_{[0,1]}(\neg P))$$

which states that introducing the context $[0.05, 0.1]\,\mathrm{W}\,\textsc{helk}$ at some point between $10-k$ and 10 time units in the future will cause the system to satisfy $\neg P$ for at least 1 time unit, and look at the time taken to apply the masked and unmasked monitoring algorithm for 1 time unit. Fig. 7.3a shows that whilst for small k the masked monitoring time is substantially reduced compared to the unmasked time since the eventually mask will only include a small region of the time domain, this difference disappears as k approaches 10 and the mask expands to include the whole time domain. We can also see that the rate of increase in masked monitoring time is not uniform; this is because, due to the adaptive nature of the monitoring algorithm, the cost associated with monitoring different regions of the time domain varies substantially depending on how much work is required to determine the Boolean value of the signal.

We also consider the property,

$$\sigma_m \triangleq \mathcal{G}_1(S_m) \wedge \mathcal{F}_{[0,1]}([0.05, 0.1]\,\mathrm{W}\,\textsc{helk} \triangleright \mathcal{G}_{[0,10]}(\neg P))$$

whose atomic proposition S_m is defined as the half-space

$$S_m \triangleq [\text{WHELK}] > \frac{1}{8}[\text{LOBSTER}] + m.$$

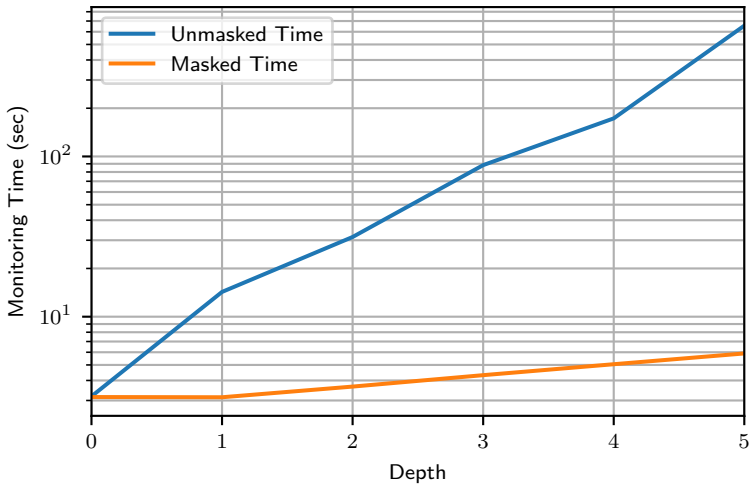
The properties $\mathcal{G}_1(S_m)$ mask a greater proportion of the time domain as m increases, and so, as we see in Fig. 7.3b, the monitoring times for σ_m are substantially reduced from the unmasked times.

We will now look at the cost of alternating temporal and contextual operators. As discussed in section Section 5.3.3, these make it possible to model experimental protocols in which events introduce new agents into a system at given (potentially uncertain) time points, and observe the results. For example, we consider the sequence of nested properties,

$$\begin{aligned}
\zeta_0 &\triangleq \mathcal{F}_{[0,5]}\, P \\
\zeta_1 &\triangleq \mathcal{G}_{[5,5.1]}([1, 1.5] \quad \text{LOBSTER} \triangleright \zeta_0) \\
\zeta_2 &\triangleq \mathcal{G}_{[5,5.1]}([0.5, 0.6]\ \text{WHELK} \quad \triangleright \zeta_1) \\
\zeta_3 &\triangleq \mathcal{G}_{[5,5.1]}([1, 1.5] \quad \text{LOBSTER} \triangleright \zeta_2) \\
\zeta_4 &\triangleq \mathcal{G}_{[5,5.1]}([0.5, 0.6]\ \text{WHELK} \quad \triangleright \zeta_3) \\
\zeta_5 &\triangleq \mathcal{G}_{[5,5.1]}([1, 1.5] \quad \text{LOBSTER} \triangleright \zeta_4)
\end{aligned}$$

which state that the P should be satisfied within 5 time units after we alternate between introducing more whelks or more lobsters after every 5 time units (allowing for an up to 0.1 time unit delay in introduction). We note that each of the formulae ζ_n features n alternations between temporal and contextual operators, and thus, has a *sandwich alternation depth* of n. These deeply nested sequences of interventions are representative of real experiments in the chemical or biological setting which can require initiating a sequence of different reactions over a number of days, however, such properties are challenging for our unmasked monitoring algorithm for $\mathcal{LBUC}$ and for existing monitoring algorithms for $\mathcal{LBC}$, since the number of simulations of the system these algorithms perform is exponential in the sandwich alternation depth of a formula.

The monitoring times for applying the masked and unmasked variants of our monitoring algorithm for 1 time unit from uncertain initial set Π_2 are shown in Fig. 7.4. This shows that whilst the unmasked monitoring time increases exponentially with sandwich alternation depth, the masked monitoring algorithm substantially reduces the coefficient of this exponential growth, reducing the monitoring time for the depth 5 formula from 655 seconds to 6 seconds. This demonstrates how monitoring many multi-stage experimental protocols is made feasible via masks. As we have seen, however, these efficiency gains



(a) Log plot of unmasked and masked times.

Depth	0	1	2	3	4	5
Unmasked Time (sec)	3.18	14.28	31.43	88.29	172.83	654.95
Masked Time (sec)	3.15	3.14	3.66	4.32	5.05	5.9

(b) Table of monitoring times.

Figure 7.4: Monitoring time for formulae ζ_n of increasing alternation depth n.

from masks depend upon the intervals of uncertainty in the time windows used by these operators being relatively small; for wider windows of uncertainty the masked monitoring times can be expected to grow closer to the unmasked monitoring times and so these properties remain difficult to monitor.

Chapter 8

Contextual Signals

In this chapter we refine our view of a system's behaviour over an uncertain context by introducing a new type of *contextual signal* which captures the truth of a proposition over the space of possible system contexts. Whilst our previous monitoring procedure already allows us to generate a single signal φ which quantifies over the uncertainty induced by an interval initial context $\mathcal{C}$ or by the initial conditions and parameters of a system, we are limited by Flow*'s ability to simulate the behaviour of a system given large initial sets or uncertain parameter ranges; once these uncertainties grow too large, validated integration will give unsatisfactorily coarse results or fail completely. Moreover, even when Flow* can handle a given initial set and produce signals for each atomic proposition of a complex formula φ, these may be insufficient to conclude the overall truth of the formula given the variety of behaviour the system exhibits for different regions exhibits over the context. For example, given a property $\zeta = \mathcal{C} \triangleright (\varphi \vee \psi)$, we are only able to conclude ζ is true if either of φ and ψ hold over the whole of $\mathcal{C}$ and not, say, when φ holds over half of $\mathcal{C}$ and ψ over the other half. That is, our existing signal monitoring algorithm for contextual properties is inherently limited in precision since it implicitly distributes the quantification over contexts from the top level of the formula to its atomic propositions, in a manner analogous to the *dependency problem* of interval arithmetic. Contextual signals address this problem by recording the behaviour of propositions over each individual point in the context in a compositional manner, allowing us to minimize the uncertainty at each point and defer the quantification over the context as long as possible to avoid the accumulation of uncertainties. In order to concretely compute contextual signals we introduce signal trees, which consist of infinite trees of signals for successive subdivisions of the system context, allowing us to derive contextual signals of arbitrary precision.

In Section 8.1 we introduce our contextual signals and our concrete signal tree monitor-

ing algorithm. In Section 8.3 we combine this with masks to implement masked contextual monitoring and down-tree masking. Then in Section 8.3 we present a number of demonstrations and benchmarks of our contextual monitoring procedure on our predator-prey role-reversal model.

8.1 Contextual Signals

In this section we will first introduce the abstract definition of contextual signals, which track the truth value of a proposition over an uncertain context, before introducing the more concrete methods of signal trees and flowpipe trees to monitor and combine contextual signals in a computable manner.

8.1.1 Abstract Contextual Signals and Refinement

In this chapter we will restrict our attention to uncertain contexts of form,

$$\mathcal{C} \equiv [a_1, b_1]\, S_1 \parallel \ldots \parallel [a_n, b_n]\, S_n.$$

We may interpret such contexts as sets

$$\mathcal{C} = \left\{ s_1\, S_1 \parallel \ldots \parallel s_n\, S_n \;\middle|\; \forall j,\ s_j \in [a_j, b_j] \right\}$$

containing each concrete context process consistent with the uncertain context, and thus as an n-dimensional *context space* into which a system may be placed, parametrized by the uncertain parameters s_j. By Remark 5.2.4 this form of context also encompasses uncertain initial conditions as a special case, and based on Section 5.1.2 we can also decompose affinity networks with *time-invariant* uncertain parameters so that their uncertain part is a context of this form. However, affinity networks including *time-varying* uncertainty are out of the scope of the methods in this chapter.

We may then introduce the abstract definition of contextual signals which define a signal for a proposition which varies over the context space.

Definition 8.1.1. A *contextual signal* over a context $\mathcal{C}$ is a function

$$s : \mathcal{C} \to \mathbb{R}_{\geq 0} \to \mathbb{T}.$$

We denote by $\textsc{CtxSig}(\mathcal{C}) \triangleq [\mathcal{C} \to \mathbb{R}_{\geq 0} \to \mathbb{T}]$ the collection of all contextual signals over the context $\mathcal{C}$.

Definition 8.1.2. A contextual signal s over context $\mathcal{C}$ is said to be a *contextual signal for a property φ in model $\mathcal{M}$* if, for any concrete context $C \in \mathcal{C}$, the partially applied signal $s(C)$ is a signal for φ in the contextualized model $C \parallel \mathcal{M}$ in the sense of Definition 6.1.2.

This definition treats a contextual signal for a proposition φ as a map

$$\mathcal{C} \to (\mathbb{R}_{\geq 0} \to \mathbb{T}) : C \mapsto s(C)$$

defining signals for φ at every possible instance of an uncertain context $\mathcal{C}$. We may alternatively take a spatial view at each time point, viewing a contextual signal as a map

$$m : \mathbb{R}_{\geq 0} \to (\mathcal{C} \to \mathbb{T}) : t \mapsto C \mapsto s(C, t)$$

defined by $m(t)(C) = s(C)(t)$ which at each time $t \in \mathbb{R}_{\geq 0}$ gives a truth value for φ at each point C in the space of potential contexts. Therefore contextual signals can be seen as spatio-temporal signals over the context space[1].

Based on our semantics for $\mathcal{LBUC}$, it is possible to directly define an optimal contextual signal for a proposition φ over a given context space $\mathcal{C}$ which optimally captures the truth value of φ on each concrete context $C \in \mathcal{C}$.

Proposition 8.1.3. *Given a context space $\mathcal{C}$, the contextual signal $w^{\mathcal{C}}_{\varphi}$ defined by*

$$w^{\mathcal{C}}_{\varphi}(C)(t) = \begin{cases} \mathbf{T} & \text{if } (\mathcal{M} \parallel C, t) \models \varphi \\ \mathbf{F} & \text{if } (\mathcal{M} \parallel C, t) \models \neg\varphi \\ \mathbf{U} & \text{otherwise} \end{cases}$$

for $C \in \mathcal{C}$ and $t \in \mathbb{R}_{\geq 0}$ is the optimal contextual signal for φ. That is, for any contextual signal s,

$$s \text{ is a contextual signal for } \varphi \quad \iff \quad s \sqsubseteq w^{\mathcal{C}}_{\varphi}.$$

Proof. Firstly, suppose that s is a contextual signal for φ. Then if $s(C)(t) = b \in \mathbb{B}$, $(\mathcal{M} \parallel C, t) \models b$ and hence $w^{\mathcal{C}}_{\varphi}(t) = b$ showing that $s \sqsubseteq w^{\mathcal{C}}_{\varphi}$.

Conversely, assume that $s \sqsubseteq w^{\mathcal{C}}_{\varphi}$. Then, for any $C \in \mathcal{C}$, for any $t \in \mathbb{R}_{\geq 0}$, if $s(C)(t) = b \in \mathbb{B}$ we have $b = s(C)(t) \sqsubseteq w^{\mathcal{C}}_{\varphi}(t)$ and hence $w^{\mathcal{C}}_{\varphi}(t) = b$. But then we must have $\mathcal{M} \parallel C \models (\varphi \Leftrightarrow b)$. Therefore $s(C)$ is a signal for φ in the model $\mathcal{M} \parallel C$ for every $C \in \mathcal{C}$ and hence s is a contextual signal for $\mathcal{M}$ in context $\mathcal{C}$. □

[1] In contrast to normal spatio-temporal signals which capture real variation in the concentrations of agents across spatial domain captured in the model, here we consider a conceptual space of potential contexts, which are assumed to be established and fixed at time 0 and do not interact with each other over time.

This optimal contextual signal is, however, impossible to compute in general so our aim henceforth will be to approximate it.

Whilst a contextual signal provides more information by allowing us to explore how the truth value of a proposition varies over the context space, it can also provide a tighter view of the truth of a proposition over the whole context space, by generating a refined signal:

Definition 8.1.4. A signal w is a *refined signal* for φ in context $\mathcal{C}$ for model $\mathcal{M}$ provided it satisfies

$$w(t) = \mathbf{T} \qquad \Longrightarrow \qquad \forall C \in \mathcal{C},\, (\mathcal{M} \,\|\, C, t) \models \quad \varphi \tag{8.1}$$

$$w(t) = \mathbf{F} \qquad \Longrightarrow \qquad \exists C \in \mathcal{C},\, (\mathcal{M} \,\|\, C, t) \models \neg\varphi. \tag{8.2}$$

We note that this does **not** imply either,

1. w is a signal for $\mathcal{C} \triangleright \varphi$ in $\mathcal{M}$, or
2. w is a signal for φ in $\mathcal{M} \,\|\, \mathcal{C}$.

The former statement (Item 1) does not hold since the the refined signal considers evolution of the system after the the context $\mathcal{C}$ is already been introduced whereas a signal for $\mathcal{C} \triangleright \varphi$ tracks the instantaneous truth value of φ after each time point at which $\mathcal{C}$ may be introduced. The latter statement (Item 2) does not hold since the refined signal is too tight to be a signal for the φ over this composed system since it can be refuted by a single concrete context, whereas a signal for φ would have to account for each individual trajectory of the composed system.

Compared to the normal definition of a signal for a proposition, the looser refutation condition for refined signals means that they cannot be composed directly (for example, we cannot decide if $\mathcal{F}_{[a,b]}\,\varphi$ satisfies Eq. (8.2) from a refined signal for φ since φ being false on *some* context for each of the time points in $t + [a, b]$ does not imply that there is a single context on which φ is false at *all* of these time points). On the other hand, contextual signals contain more than enough information to be determined compositionally since they independently determine signals at each point in the context space and can be condensed into refined signal as needed by the following definition.

Definition 8.1.5. Given a contextual signal $s : \mathcal{C} \to \mathbb{R}_{\geq 0} \to \mathbb{T}$, the *refined signal associated with* s is the signal defined by

$$\text{refined}(s) \triangleq \bigwedge_{C \in \mathcal{C}} s(C).$$

Firstly we see that refinement is a monotone map.

Proposition 8.1.6. *For any context $\mathcal{C}$, the refinement map* $\text{refined} : \textsc{CtxSig}(\mathcal{C}) \to (\mathbb{R}_{\geq 0} \to \mathbb{T})$ *is monotone.*

Proof. For any pair of context signals s, w with $s \sqsubseteq w$ and $t \in \mathbb{R}_{\geq 0}$, we have that

$$\text{refined}(s)(t) = \bigwedge_{C \in \mathcal{C}} s(C)(t) \sqsubseteq \bigwedge_{C \in \mathcal{C}} w(C)(t) = \text{refined}(w)(t)$$

since $s(C)(t) \sqsubseteq w(C)(t)$ for all $C \in \mathcal{C}$ and since it can easily been seen that arbitrary conjunctions are monotone under $\sqsubseteq$. □

We then see that refining a contextual signal for a proposition does in fact yield a refined signal for that proposition.

Proposition 8.1.7. *Given a contextual signal s for a proposition φ in context $\mathcal{C}$,* $\text{refined}(s)$ *is a refined signal for φ in context $\mathcal{C}$.*

Proof. If $\text{refined}(s)\,(t) = \mathbf{T}$ then for any $C \in \mathcal{C}$, $s(C)(t) = \mathbf{T}$ and hence, since s_C is a signal for φ in $\mathcal{M} \parallel C$, $\mathcal{M} \parallel C \models_t \varphi$. Conversely, if $\text{refined}(s)\,(t) = \mathbf{F}$, then for some $C \in \mathcal{C}$, $s(C)(t) = \mathbf{F}$ and hence $(\mathcal{M} \parallel C) \models_t \neg\varphi$. Therefore w is a refined signal for φ. □

Moreover, we see that refining the optimal contextual signal $w^{\mathcal{C}}_{\varphi}$ (Proposition 8.1.3) gives an optimal refined signal for a proposition over a given context.

Proposition 8.1.8. *Given the optimal contextual signal $w^{\mathcal{C}}_{\varphi}$ for a proposition φ in context $\mathcal{C}$, the signal* $\text{refined}\left(w^{\mathcal{C}}_{\varphi}\right)$ *is the optimal refined signal for φ in context $\mathcal{C}$, that is, for any signal s,*

$$s \text{ is a refined signal for } \varphi \text{ in context } \mathcal{C} \iff s \sqsubseteq \text{refined}\left(w^{\mathcal{C}}_{\varphi}\right).$$

Proof. Firstly, suppose that $s \sqsubseteq \text{refined}\left(w^{\mathcal{C}}_{\varphi}\right)$. Then, for any $t \in \mathbb{R}\geq 0$, if $s(t) = b \in \mathbb{B}$, we must have $\text{refined}\left(w^{\mathcal{C}}_{\varphi}\right)(t) = \bigwedge_{C \in \mathcal{C}} w^{\mathcal{C}}_{\varphi}(C)(t) = b$. If $b = \mathbf{T}$, for any $C \in \mathcal{C}$ we must have $w^{\mathcal{C}}_{\varphi}(C)(t) = b = \mathbf{T}$ and hence $\mathcal{M} \parallel C \models_t \varphi$. On the other hand, if $b = \mathbf{F}$, we must have some $C \in \mathcal{C}$ with $w^{\mathcal{C}}_{\varphi}(C)(t) = b = \mathbf{F}$ and hence $\mathcal{M} \parallel C \models_t \varphi$. Therefore s is a refined signal for φ in context $\mathcal{C}$.

Conversely, suppose that s is a refined signal for φ in context $\mathcal{C}$ and consider and arbitrary $t \in \mathbb{R}_{\geq 0}$. If $s(t) = \mathbf{T}$ then for any $C \in \mathcal{C}$ we must have $\mathcal{M} \parallel \mathcal{C} \models_t \varphi$ so that $w^{\mathcal{C}}_{\varphi}(C)(t) = \mathbf{T}$ and hence

$$\text{refined}\left(w^{\mathcal{C}}_{\varphi}\right)(t) = \bigwedge_{C \in \mathcal{C}} (C)(t) = \mathbf{T} \sqsupseteq \mathbf{T} = s(t).$$

On the other hand $s(t) = \mathbf{F}$ then we must have some $C \in \mathcal{C}$ such that $\mathcal{M} \parallel \mathcal{C} \models_t \varphi$ so that $w^{\mathcal{C}}_{\varphi}(C)(t) = \mathbf{T}$ and hence

$$\text{refined}\Big(w^{\mathcal{C}}_{\varphi}\Big)(t) = \bigwedge_{C \in \mathcal{C}} (C)(t) = \mathbf{F} \sqsupseteq \mathbf{F} = s(t).$$

Therefore $s \sqsubseteq \text{refined}\Big(w^{\mathcal{C}}_{\varphi}\Big)$, completing the proof. □

8.1.2 Context Trees

We now need to move from our abstract treatment of the context space as a real continuous space, to a more concrete representation as a tree of subdivided contexts. To this end we first introduce the notion of a subcontext of a context.

Definition 8.1.9. Given contexts $\mathcal{C}$ and $\mathcal{D}$ then $\mathcal{C}$ *is a subcontext of* $\mathcal{D}$ if $\mathcal{C} \subseteq D$ as sets.

We may then expand a context $\mathcal{C}$ into a tree of progressively refined subcontexts, representing spatial subdivisions of the context space.

Definition 8.1.10. A *context tree for context* $\mathcal{C}$ is a collection $\mathcal{N}_{\mathcal{C}}$ of subcontexts of $\mathcal{C}$ satisfying:

1. $\mathcal{C} \in \mathcal{N}_{\mathcal{C}}$;
2. $\mathcal{N}_{\mathcal{C}}$ is a finitely branching tree of sets;
3. $\text{int}(\mathcal{D}) \neq \varnothing$ for all $\mathcal{D} \in \mathcal{N}_{\mathcal{C}}$;
4. $\mathcal{D} = \bigcup_{\mathcal{E} \in \text{children}(\mathcal{D})} \mathcal{E}$ for each $\mathcal{D} \in \mathcal{N}_{\mathcal{C}}$;
5. for every pair of siblings $\mathcal{E} \neq \mathcal{F} \in \text{children}(\mathcal{D})$ with $\mathcal{E} \neq F$, $\text{int}(\mathcal{E}) \cap \text{int}(\mathcal{F}) = \varnothing$.

This permits the use of both finite contextual trees, which subdivide the space into a finite number of regions, and infinite, which may be iteratively expanded to progressively subdivide the space as we increase our exploration depth. We note an important property of finite context trees: their leaf nodes form a partition of the context into subcontexts which overlap only on their boundaries.

Proposition 8.1.11. *Given any finite context tree* $\mathcal{N}_{\mathcal{C}}$ *over context* $\mathcal{C}$ *we have that*

(a) $\mathcal{C} = \bigcup \text{leaves}(\mathcal{N}_{\mathcal{C}})$*;*

(b) for all $\mathcal{D}, \mathcal{E} \in \text{leaves}(\mathcal{N}_{\mathcal{C}})$, $\mathcal{D} \neq \mathcal{E} \implies \text{int}(\mathcal{D}) \cap \text{int}(\mathcal{E}) = \varnothing$.

Proof. Both properties follow by induction on the depth of the tree. □

In general, however, we are primarily interested in monitoring infinite context trees which allow us to progressively refine the space as required in a given region; these may be represented by an implementation as lazy infinite data structures. The simplest way of generating such a context tree is subdividing the context in a grid pattern along each dimension of uncertainty at each level of the tree. This is similar to the representation of 2D patterns as a quad-tree (as used in, for example, the spatio-temporal logic SpaTeL [187]).

Definition 8.1.12. Consider a context given by an bond-calculus process $\mathcal{C}$ in the form

$$\mathcal{C} \equiv \mathcal{C}_C \parallel \mathcal{C}_S$$

with uncertain component $\mathcal{C}_C \triangleq \|_{j=1}^{n} [a_j, b_j] C_j$ where $b_j > a_j$ for all j, and certain component $\mathcal{C}_S \triangleq \|_{j=1}^{m} s_j S_j$. Then the *context tree generated by* $\mathcal{C}$ is the infinite tree with root $\mathcal{C}$ and children

$$\mathcal{C}_{l_1,\ldots,l_n} \triangleq \left(\bigparallel_{j=1}^{n} I_{j,l_j} X_j \right) \parallel \mathcal{C}_S \qquad l_j \in \{0,1\}$$

where

$$I_{j,l} \triangleq \begin{cases} \left[a_j, \frac{1}{2}(a_j + b_j)\right] & \text{if } l = 0 \\ \left[\frac{1}{2}(a_j + b_j), b_j\right] & \text{if } l = 1 \end{cases},$$

and the branch rooted at each child $\mathcal{C}_{l_1,\ldots,l_n}$ is the context tree generated by $\mathcal{C}_{l_1,\ldots,l_n}$.

In the above definition we separate the uncertain component $\mathcal{C}_C$ of the context from the static component $\mathcal{C}_S$. Static context dimensions which simply represent a fixed jump in a given direction are not amenable to context subdivision, motivating their exclusion such that the dimension of the context tree will only be the number of genuinely uncertain context variables n.

8.1.3 Signal Trees

We can now define a signal tree over a context tree, which defines a signal for each node of the context tree.

Definition 8.1.13. Given a context tree $\mathcal{N}_\mathcal{C}$, a *signal tree* s *over* $\mathcal{N}_\mathcal{C}$ is an monotone function

$$s : \mathcal{N}_\mathcal{C} \to \mathbb{R}_{\geq 0} \to \mathbb{T}$$

with respect to the information ordering $\sqsubseteq$ (that is, if $\mathcal{D} \subseteq \mathcal{E}$ then $s(\mathcal{D})(t) \sqsupseteq s(\mathcal{E})(t)$) such that for all $\mathcal{D}, \mathcal{E} \in \mathcal{N}_\mathcal{C}$ if $\mathcal{D} \cap \mathcal{E} \neq \varnothing$ then for any $t \in \mathbb{R}_{\geq 0}$ either $s(\mathcal{D})(t) \sqsubseteq s(\mathcal{E})(t)$ or $s(\mathcal{E})(t) \sqsubseteq s(\mathcal{D})(t)$. We denote by $\textsc{SigTree}(\mathcal{N}_\mathcal{C})$ the collection of all signal trees over the context tree $\mathcal{N}_\mathcal{C}$.

Depending on the underlying context tree, we are interested in both finite and infinite signal trees which may be computed and operated on lazily.

Definition 8.1.14. Given a context tree $\mathcal{N}_\mathcal{C}$, a signal tree $s : \mathcal{N}_\mathcal{C} \to \mathbb{R}_{\geq 0} \to \mathbb{T}$ is called *finite* / *infinite* if the underlying context tree $\mathcal{N}_\mathcal{C}$ is a finite / infinite set respectively.

Then we define a signal tree for a property φ over a context $\mathcal{N}_\mathcal{C}$ tree, as a signal tree which gives signal for φ on each context in $\mathcal{N}_\mathcal{C}$.

Definition 8.1.15. Given a proposition φ and a model $\mathcal{M}$, a signal tree s is a *signal tree for φ in $\mathcal{M}$ over $\mathcal{N}_\mathcal{C}$* if $s(\mathcal{D})$ is a signal for φ for every context $\mathcal{D} \in \mathcal{N}_\mathcal{C}$.

Now, for every contextual signal over $\mathcal{C}$ we attempt to directly define a signal tree approximation over $\mathcal{N}_\mathcal{C}$ as follows.

Definition 8.1.16. Given a contextual signal $w : \mathcal{C} \times \mathbb{R}_{\geq 0}$ and a context tree $\mathcal{N}_\mathcal{C}$, the *signal tree w_* of s over $\mathcal{N}_\mathcal{C}$* is defined by

$$w_*(\mathcal{D}) = \bigcap_{D \in \mathcal{D}} w(D).$$

This also allows us to use to find the optimal signal tree over a given context tree for a given proposition.

Proposition 8.1.17. *Given a context tree $\mathcal{N}_\mathcal{C}$ over context we have that*

$$(w_\varphi)_*(\mathcal{D})(t) = \begin{cases} \mathbf{T} & \text{if } (\mathcal{M} \parallel \mathcal{D}, t) \models \varphi \\ \mathbf{F} & \text{if } (\mathcal{M} \parallel \mathcal{D}, t) \models \neg\varphi \\ \mathbf{U} & \text{otherwise} \end{cases} \tag{8.3}$$

for $t \in \mathbb{R}_{\geq 0}$ and that $(w_\varphi)_*$ is the optimal signal tree for φ over $\mathcal{N}_\varphi$. That is, for any signal tree s over $\mathcal{N}_\mathcal{C}$,

$$s \text{ is a signal tree for } \varphi \iff s \sqsubseteq w_\varphi. \tag{8.4}$$

Proof. Firstly for any $b \in \mathbb{B}$ we have that

$$\begin{aligned}
& (w_\varphi)_*(\mathcal{D})(t) = \bigcap_{D \in \mathcal{D}} w_\varphi(\mathcal{D})(t) = b \\
\iff \quad & \text{for all } D \in \mathcal{D},\ w_\varphi(D)(t) && \text{by the definition of } \bigcap \\
\iff \quad & \text{for all } D \in \mathcal{D},\ (\mathcal{M} \parallel D, t) \models (\varphi \Leftrightarrow b) && \text{by Proposition 8.1.3} \\
\iff \quad & (\mathcal{M} \parallel \mathcal{D}, t) \models (\varphi \Leftrightarrow b) && \text{by } \mathcal{LBUC} \text{ semantics (5.2.2)}
\end{aligned}$$

showing that Eq. (8.3) holds. But then Eq. (8.4) follows directly from the definitions. □

Conversely, we define a contextual signal based on a given signal tree as follows.

Definition 8.1.18. Given a signal tree s, *the contextual signal* s^* *of* s is defined by

$$s^*(D) = \bigcup_{\mathcal{D} \in \mathcal{N}_\mathcal{C} : D \in \mathcal{D}} s(\mathcal{D}).$$

The union of signals in the above definition is well defined since a signal tree is guaranteed to be consistent on all overlapping context nodes.

We now see that these two relationships define a Galois correspondence [132] between contextual signals and signal trees over a particular context tree.

Proposition 8.1.19. *Given a context tree* $\mathcal{C}$ *over context* $\mathcal{C}$*, the maps* F^*, F_* *defined by*

$$\begin{aligned}
F^* &: \textsc{SigTree}(\mathcal{N}_\mathcal{C}) \to \textsc{CtxSig}(\mathcal{C}) &&: s \mapsto s^* \\
F_* &: \textsc{CtxSig}(\mathcal{C}) \to \textsc{SigTree}(\mathcal{N}_\mathcal{C}) &&: w \mapsto w_*
\end{aligned}$$

form a monotone Galois correspondence $F = (F^*, F_*)$*. That is,* F^* *and* F_* *are monotone maps and for every signal tree* s *over* $\mathcal{C}$ *and contextual signal* w *over* $\mathcal{C}$*,*

$$F^* s = s^* \sqsubseteq w \quad \iff \quad s \sqsubseteq w_* = F_* w.$$

Proof. Firstly, it can easily be seen that F^* and F_* are both monotone functions.

Then suppose that $s^* \sqsubseteq w$. If for some $\mathcal{D} \in \mathcal{N}_\mathcal{C}$ and $t \in \mathbb{R}_{\geq 0}$ we have that $s(\mathcal{D})(t) = b \in \mathbb{B}$, for any $D \in \mathcal{D}$ we must have that

$$b = s(\mathcal{D})(t) \sqsubseteq s^*(D)(t) \sqsubseteq w(D)(t)$$

and so $w(D)(t) = b$, and consequently $w_*(\mathcal{D})(t) = b$ by the definition of w_*, showing that $s \sqsubseteq w_*$.

Conversely suppose that $s \sqsubseteq w_*$. Then for any $D \in \mathcal{C}$, if $s^*(D)(t) = b \in \mathbb{B}$ we must have $s(\mathcal{D})(t) = b$ for some $\mathcal{D} \in \mathcal{N}_\mathcal{C}$ with $D \in \mathcal{D}$. Then

$$b = s(\mathcal{D})(t) \sqsubseteq w_*(\mathcal{D})(t)$$

and hence $w_*(\mathcal{D})(t) = b$. But then $D \in \mathcal{D}$ so, by the definition of s^*, $w(\mathcal{D})(t) = b$, proving that $s^* \sqsubseteq w$, and thus (F^*, F_*) form a Galois correspondence. □

This reassures us that we may use signal trees as a sound approximation of contextual signals. We can also deduce that a signal tree for a given proposition corresponds to a contextual signal for that proposition.

Proposition 8.1.20. *Given a signal tree s and a proposition φ, s is a signal tree for φ iff s^* is a contextual signal for φ.*

Proof. This follows since

$$\begin{aligned}
s \text{ is a signal tree for } \varphi &\iff s \sqsubseteq (w_\varphi)_* && \text{by Proposition 8.1.17} \\
&\iff s^* \sqsubseteq w_\varphi && \text{by Proposition 8.1.19} \\
&\iff s^* \text{ is a contextual signal for } \varphi && \text{by Proposition 8.1.3.}
\end{aligned}$$ □

As with contextual signals, signal trees for complex proportions may be defined compositionally by composing the signal trees of formulae following the structure of the tree.

Proposition 8.1.21. *Given context tree $(\mathcal{N}_\mathcal{C}, \prec_\mathcal{C})$ and propositions φ and ψ respectively having signal trees s_φ and s_ψ, we have the following signal trees for complex propositions:*

- *The proposition $\neg\varphi$ has a signal tree $s_{\neg\varphi}$ given by*

$$s_{\neg\varphi}(\mathcal{D}) = \neg s_\varphi(\mathcal{D}).$$

- *The proposition $\varphi \vee \psi$ has a signal tree $s_{\varphi\vee\psi}$ given by*

$$s_{\varphi\vee\psi}(\mathcal{D}) = s_\varphi(\mathcal{D}) \vee s_\psi(\mathcal{D}).$$

- *The proposition $\varphi \,\mathcal{U}_J\, \psi$ has a signal tree $s_{\neg\varphi}$ given by*

$$s_{\varphi\mathcal{U}_J\psi}(\mathcal{D}) = s_\varphi(\mathcal{D}) \,\mathcal{U}_J\, s_\psi(\mathcal{D}).$$

where $\mathcal{D} \in \mathcal{N}_\mathcal{C}$ is an arbitrary node of the context tree.

Proof. We will give the proof for $\varphi \,\mathcal{U}_J\, \psi$ since the signal trees for the other operators follows similarly. Since s and w are respectively signal trees for φ and ψ, for all $D \in \mathcal{D}$, $s(\mathcal{D})$ and $w(\mathcal{D})$ are respectively signals for φ and ψ on D, and hence, $s(\mathcal{D}) \,\mathcal{U}_J\, w(\mathcal{D})$ is a signal for $\varphi \,\mathcal{U}_J\, \psi$ by Proposition 6.1.8, proving that $s \,\mathcal{U}_J\, w$ is a signal tree for $\varphi \,\mathcal{U}_J\, \psi$. □

Thus, given signal trees for atomic propositions (we will see how these may be computed in following sections), we are able to propagate the signals trees upwards to compute the signal trees for complex propositions as in our regular signal monitoring algorithms.

8.1.4 Refining Signal Trees

In this section we will see how we may compute refined signals from signal trees.

To start with we consider the special case of finite signal trees, for which we may explicitly compute the associated refined signals via the following result:

Lemma 8.1.22. *Given sets A, B such that $A \neq \varnothing$ is nonempty and convex and* $\operatorname{int}(A) \cap \operatorname{int}(B) = \varnothing$ *then* $\operatorname{cl}(A) \cap \operatorname{int}(B) = \varnothing$.

Proof. Suppose we have some $a \in \operatorname{cl}(A) \cap \operatorname{int}(B)$. By convexity and nonemptiness $\operatorname{cl}(A) = \operatorname{cl}(\operatorname{int}(A))$ so $a \in \operatorname{cl}(\operatorname{int}(A))$. Since $a \in \operatorname{int}(B)$, for some open neighbourhood U of A, $U \subseteq \operatorname{int}(B)$, but as $a \in \operatorname{cl}(\operatorname{int}(A))$, $\varnothing \neq U \cap \operatorname{int}(A) \subseteq \operatorname{int}(B) \cap \operatorname{int}(A)$, a contradiction completing the proof. □

Proposition 8.1.23. *For any finite signal tree $s : \mathcal{N}_{\mathcal{C}} \times \mathbb{R}_{\geq 0} \to \mathbb{T}$ we have that*

$$\operatorname{refined}(s^*) \triangleq \bigwedge_{\mathcal{D} \in \operatorname{leaves}(\mathcal{N}_{\mathcal{C}})} s(\mathcal{D}).$$

Proof. Take any $t \in \mathbb{R}_{\geq 0}$.

If $\operatorname{refined}(s^*)(t) = \mathbf{T}$ then for any node $\mathcal{D} \in \mathcal{N}_{\mathcal{C}}$ with $\operatorname{children}(\mathcal{D}) = \varnothing$, pick some $D \in \operatorname{int}(\mathcal{D}) \neq \varnothing$. Then we must have $s^*(D)(t) = \mathbf{T}$ and hence there must some $\mathcal{E} \in \mathcal{N}_{\mathcal{C}}$ with $D \in \mathcal{E}$ and $s(\mathcal{E})(t) = \mathbf{T}$. But then $\mathcal{E}$ must be an ancestor of $\mathcal{E}$, since, if we assume not, there must be some leaf descendent $\mathcal{E}' \subseteq E$ with $D \in \mathcal{E}'$ by Proposition 8.1.11 (a). But as $\mathcal{E}' \neq \mathcal{D}$ by our assumption, Proposition 8.1.11 (b) tells us that $\operatorname{int}(\mathcal{D}) \cap \operatorname{int}(\mathcal{E}') = \varnothing$ and so by Lemma 8.1.22, $\varnothing = \operatorname{cl}(\mathcal{E}') \cap \operatorname{int}(\mathcal{D}) \ni b$, a contradiction. But now we know that $\mathcal{E}$ is an ancestor of $\mathcal{D}$ and so $s(\mathcal{D})(t) \sqsupseteq s(\mathcal{E})(t) \sqsupseteq \mathbf{T}$ and hence $s(\mathcal{D})(t) = \mathbf{T}$, showing that $\bigwedge_{\mathcal{D} \in \operatorname{leaves}(\mathcal{N}_{\mathcal{C}})} s(\mathcal{D})(t) = \mathbf{T}$.

If $\operatorname{refined}(s^*)(t) = \mathbf{F}$ then for some $D \in \mathcal{C}$, $s^*(D)(t) = \mathbf{F}$ and so by the definition of s^* we must have some $\mathcal{D}_0 \in \mathcal{N}_{\mathcal{C}}$ with $D \in \mathcal{D}_0$ and $s(\mathcal{D}_0)(t) = \mathbf{F}$. Then, picking any

leaf descendent $\mathcal{D}'_0 \subseteq \mathcal{D}_0$, $s(\mathcal{D}'_0)(t) \sqsupseteq s(\mathcal{D})(t) = \mathbf{F}$, showing that $s(\mathcal{D}'_0)(t) = \mathbf{F}$ and hence $\bigwedge_{\mathcal{D}_0 \in \text{leaves}(\mathcal{N}_\mathcal{C})} s(\mathcal{D})(t) = \mathbf{F}$. □

We will now see how we may extract refined signals from an infinite signal tree by truncating it at a given finite depth.

Definition 8.1.24. Given a context tree $\mathcal{N}_\mathcal{C}$, we define the *depth d truncation of $\mathcal{N}_\mathcal{C}$* as

$$\mathcal{N}_\mathcal{C}^{\leq d} \triangleq \Big\{\ D \in \mathcal{N}_\mathcal{C} \ \Big|\ \text{depth}(\mathcal{D}) \leq d\ \Big\}.$$

Given a signal tree s over $\mathcal{N}_\mathcal{C}$, we define the *depth d truncation $s_{\leq d}$ of s* as the domain restriction of s to $\mathcal{N}_\mathcal{C}^{\leq d}$, that is

$$s_{\leq d} : \mathcal{N}_\mathcal{C}^{\leq d} \to \mathbb{R}_{\geq 0} \to \mathbb{T} : \mathcal{D} \mapsto s(\mathcal{D}).$$

Proposition 8.1.25. *Given context trees $\mathcal{N}_\mathcal{C}^1 \subseteq \mathcal{N}_\mathcal{C}^2$, if s_1 and s_2 are signal trees over $\mathcal{N}_\mathcal{C}^1$ and $\mathcal{N}_\mathcal{C}^2$ respectively and $s_1(\mathcal{D}) = s_2(\mathcal{D})$ for all $\mathcal{D} \in \mathcal{N}_\mathcal{C}^1$, then $s_1^* \sqsubseteq s_2^*$.*

Proof. Define the signal tree $s_3 : \mathcal{N}_\mathcal{C}^2 \to \mathbb{R}_{\geq 0} \to \mathbb{T}$ by $s_3(\mathcal{D}) = s_1(\mathcal{E})$ where $\mathcal{E}$ is the least ancestor of $\mathcal{D}$ in $\mathcal{N}_\mathcal{C}^2$ such that $\mathcal{E} \in \mathcal{N}_\mathcal{C}^1$. Then we can see both that $s_1^* = s_3^*$ and $s_3 \sqsubseteq s_2$, showing by monotonicity (Proposition 8.1.19) that $s_1^* = s_3^* \sqsubseteq s_2^*$. □

Corollary 8.1.26. *Given a signal tree s and natural numbers $n, m \in \mathbb{N}$ such that $n \leq m$ we have that $s^*_{\leq n} \sqsubseteq s^*_{\leq m} \sqsubseteq s^*$.*

We may then use truncations of a signal tree to define refined signals to different depths of refinement.

Definition 8.1.27. Given a signal tree s we define the *d refined signal of s* as the signal

$$\begin{aligned} s^d &\triangleq \text{refined}\Big(s^*_{\leq d}\Big) = \bigwedge \text{leaves}\Big(s^*_{\leq d}\Big) \\ &= \bigwedge \Big\{\ s(\mathcal{D}) \ \Big|\ \mathcal{D} \in \mathcal{N}_\mathcal{C}, (\text{children}(\mathcal{D}) = \varnothing \vee \text{height}(\mathcal{D}) = d)\ \Big\}. \end{aligned}$$

The d-refined signal may be computed recursively over the signal tree as $s^d = \text{refined}(n, \mathcal{C}, s)$ where

$$\begin{aligned} \text{refined}(d, \mathcal{D}, s) &\triangleq s(\mathcal{D}) \quad \text{whenever } d = 0 \text{ or } \text{children}(\mathcal{D}) = \varnothing \\ \text{refined}(d+1, \mathcal{D}, s) &\triangleq \bigwedge_{\mathcal{E} \in \text{children}(\mathcal{D})} \text{refined}(d, \mathcal{E}, s)\,. \end{aligned}$$

This then allows us to use a signal tree s for a given property to define a sequence of progressively refined signals for the property as follows.

Proposition 8.1.28. *Given a signal tree s for a property φ and natural numbers $n, m \in \mathbb{N}_{\geq 0}$ such that $n \geq m$ we have that*

$$\text{refined}(s^*) \sqsupseteq s^n \sqsupseteq s^m \sqsupseteq s^0, \tag{8.5}$$

that the above signals are all refined signals for φ whilst s^0 is a signal for φ.

Proof. The inequalities Eq. (8.5) follow directly by Corollary 8.1.26 and monotonicity of refinement (Proposition 8.1.6). But then clearly $s^0 = s(\mathcal{C})$ is a signal for φ and the all of the other signal in Eq. (8.5) must be refined signals for φ by the optimality of $\text{refined}\left(w_\varphi^{\mathcal{C}}\right) \sqsupseteq \text{refined}(s^*)$. □

8.2 Signal Tree monitoring using Flow*

In this section we investigate methods of using Flow* flowpipes to generate signal trees for atomic propositions and context operators and then define how these signal trees may be composed to achieve general signal tree monitoring over flowpipes.

To this end we start by defining a tree of flowpipes specialized to each node of a context tree.

Definition 8.2.1. A *flowpipe tree* f over context tree $\mathcal{N}_{\mathcal{C}}$ and time domain $T \in \mathbb{IR}_{\geq 0}$ consists of a partial map

$$f : \mathcal{N}_{\mathcal{C}} \rightharpoonup \left[\left(\mathbf{S} \times T \to \widehat{\text{MIX}}\right) \times \mathcal{P}(\mathbf{S})\right]$$

which maps each node $\mathcal{D} \in \mathcal{N}_{\mathcal{C}}$ to a pair $f(\mathcal{D}) = (f_{\mathcal{D}}, \mathbf{S}_{\mathcal{D}})$ of a flowpipe $f_{\mathcal{D}}$ and a restricted space domain $\mathbf{S}_{\mathcal{D}} \subseteq \mathbf{S}$. The flowpipe tree f is a *flowpipe tree for model $\mathcal{M}$ over time domain I* if for every node $\mathcal{D} \in \mathcal{N}$, every trajectory $\mathbf{x}$ of the composed system $\mathcal{M} \parallel \mathcal{D}$, and every time point $t \in T$, $\mathbf{x}(t) \in f_{\mathcal{D}}(\mathbf{S}_{\mathcal{D}}, t)$.

This means that for each node $\mathcal{D}$ of the tree we may enclose the trajectories in this context by evaluating the flowpipe $f_{\mathcal{D}}$ on the associated space domain $\mathbf{S}_{\mathcal{D}}$. This is a partial map, reflecting the fact that it is not always possible to generate a flowpipe covering the entirety of a given initial system.

Given a flowpipe tree for a model $\mathcal{M}$ over context tree $\mathcal{N}_{\mathcal{C}}$ it is possible to define an associated signal tree as follow. In the case of an unknown flowpipe $f(\mathcal{D}) = \bot$ at context node $\mathcal{D}$, the associated signal in any signal tree will be the empty signal, that is,

$$\texttt{signalTreeF}(\varphi, f, \mathcal{A}, T)(\mathcal{D}) = ((T, \mathbf{U})) = ()$$

whenever $f(\mathcal{D}) = \mathbf{U}$. Whereas if, on the other hand, $f(\mathcal{D}) = (f_{\mathcal{D}}, S_{\mathcal{D}})$ we can define the signal tree $\texttt{signalTreeF}(\rho, f, \mathcal{A}, T)$ by

$$\texttt{signalTreeF}(\rho, f, \mathcal{A}, T)(\mathcal{D}) \triangleq \texttt{signalF}(\rho, f_{\mathcal{D}}, \mathcal{A}, \mathbf{S}_{\mathcal{D}}, T)$$

that is, by monitoring ρ over each flowpipe in the flowpipe tree over its associated space domain.

The signal trees for complex operators can be defined by composing them according to the operators in Proposition 8.1.21, so that,

$$\begin{aligned}
\texttt{signalTreeF}(\neg\varphi, f, \mathcal{A}, T) &\triangleq \neg\texttt{signalTreeF}(\varphi, f, \mathcal{A}, T) \\
\texttt{signalTreeF}(\varphi \wedge \psi, f, \mathcal{A}, T) &\triangleq \texttt{signalTreeF}(\varphi, f, \mathcal{A}, T) \wedge \texttt{signalTreeF}(\psi, f, \mathcal{A}, T) \\
\texttt{signalTreeF}(\varphi \vee \psi, f, \mathcal{A}, T) &\triangleq \texttt{signalTreeF}(\varphi, f, \mathcal{A}, T) \vee \texttt{signalTreeF}(\psi, f, \mathcal{A}, T) \\
\texttt{signalTreeF}(\mathcal{F}_K\,\varphi, f, \mathcal{A}, T) &\triangleq \mathcal{F}_K(\texttt{signalTreeF}(\varphi, f, \mathcal{A}, T)) \\
\texttt{signalTreeF}(\mathcal{G}_K\,\varphi, f, \mathcal{A}, T) &\triangleq \mathcal{G}_K(\texttt{signalTreeF}(\varphi, f, \mathcal{A}, T)) \\
\texttt{signalTreeF}(\varphi\,\mathcal{U}_K\,\psi, f, \mathcal{A}, T) &\triangleq \texttt{signalTreeF}(\varphi, f, \mathcal{A}, T)\,\mathcal{U}_K\,\texttt{signalTreeF}(\psi, f, \mathcal{A}, T).
\end{aligned}$$

We must now define the signal tree for the context operator $\mathcal{D} \triangleright \varphi$ over the context tree $\mathcal{N}_{\mathcal{C}}$. It would be possible to define this based on $\texttt{signalF}$, similarly to atomic propositions, however, this would offer no opportunity refine our knowledge of the inner proposition φ under the context $\mathcal{D}$. Instead, we use may apply signal tree monitoring to define a refined signal tree as follows. For a context tree node $\mathcal{E} \in \mathcal{N}_{\mathcal{C}}$ such that $f(\mathcal{E}) = (g_{\mathcal{E}}, \mathbf{S}_{\mathcal{E}})$ we first define an interval uncertain bond-calculus model

$$\Phi_{\mathcal{E},T} \triangleq (g_{\mathcal{E}}(\mathbf{S}_{\mathcal{E}}, T), \mathcal{A})$$

which encompasses the possible state of the system in outer context $\mathcal{E}$ before the inner contextual jump occurs. We then define a derived context tree $(\mathcal{N}_{\mathcal{D},\mathcal{E},T}, \mathcal{D} \parallel \Phi_{\mathcal{E},T})$ with nodes $\mathcal{N}_{\mathcal{D},\mathcal{E},T} = \left\{ D \parallel \Phi_{\mathcal{E},T} \;\middle|\; D \in \mathcal{N}_{\mathcal{D}} \right\}$. We then monitor a signal tree s for the inner proposition φ over the contextualized signal tree $\mathcal{N}_{\mathcal{D},\mathcal{E},T}$. Taking the n^{th} refinement s^n of s then gives a refined signal for φ assuming the context $\mathcal{D}$ has been applied at some time in T. Now if we take $b = s_n(0)$ this gives a sound three-valued truth value for φ after any context jump in $\mathcal{D}$ occurs at any time point $t \in T$. If $b \in \mathbb{B}$ then we can use the signal $((T, b))$, otherwise we may recursively bisect T to refine the signal to a desired precision as before. Thus, we may deduce the signal $\texttt{signalTreeF}(\mathcal{C} \triangleright \varphi, f, \mathcal{A}, T)(\mathcal{D})$ based on the n-refined signal for the context $\mathcal{D}$, and hence define a signal tree for the context operator.

Since the inner signal tree is refined into a single three-valued truth value every time a context operator is encountered, the precision of monitoring depends on the level of refinement use at each of the context operators in a proposition. This give us a way of subdividing every source of time-invariant uncertainty within a proposition: for example, in a property $\mathcal{G}_{[a,b]}(\mathcal{C} \triangleright \varphi)$, the inner property φ will be monitored under a subdivisible context $\mathcal{C}$ in a system $\Phi_{\mathcal{E},T}$ whose uncertainties depend on the node $\mathcal{E}$ and the time domain T both of which may be further subdivided, and the precision of Flow* flowpipe construction, which may be improved by changing Flow*'s integration parameters. This means we always have a means to improve the precision of monitoring if necessary, however, it is not always obvious which combination of these factors gives the best tradeoff of precision for monitoring time.

Now we know how a signal tree follows from a flowpipe tree, it remains to decide how we compute a flowpipe tree for a given system in context. The most direct method is to compute a *physical* flowpipe tree by applying Flow* to directly compute a flowpipe for the contextualized system.

Definition 8.2.2 (Physical Flowpipe Tree)**.** Given a context tree $\mathcal{N}_{\mathcal{C}}$, we define the physical flowpipe tree f such that for each $\mathcal{D} \in \mathcal{N}$, $f(\mathcal{D}) = (f_{\mathcal{D}}, [-1,1]^n)$ where $f_{\mathcal{D}}$ is a flowpipe computed for the contextualized system $\mathcal{M} \parallel \mathcal{D}$ and $[-1,1]^n$ is its full (preconditioned) space domain.

The above method is, however, a rather costly way of generating a signal tree since it requires us to repeat the expensive process of flowpipe construction for each region of the context tree. We can save much of this effort by making use of the symbolic nature of the flowpipes generated by Flow*, so we may construct a single flowpipe which tracks the functional dependency of the state at each time point on the whole range of initial conditions, and restrict it to a sub-domain corresponding to a particular context, allowing us to monitor the signal for different regions of context space based on of a single symbolic flowpipe. To this end we must represent the initial conditions of the system $\mathcal{M} \triangleq (\Pi, \mathcal{A})$ under context $\mathcal{C}$ as a Taylor model with symbolic variables corresponding to the position of the system within context space. That is, if $\Pi \equiv \|_{j=1}^{n} I_j X_j$ and $\mathcal{C} \equiv \|_{k=1}^{n} J_k C_k$, we may represent the initial conditions of the system in context as a Taylor model

$$T(\boldsymbol{\lambda}, \boldsymbol{\mu}) \triangleq \mathop{\Big\|}_{j=1}^{n} \lambda_j X_j + \mathop{\Big\|}_{k=1}^{m} \mu_k C_k$$

with vectors of symbolic parameters $\boldsymbol{\lambda} = (\lambda_j)_j$ and $\boldsymbol{\mu} = (\mu_j)_j$ where $\lambda_j \in I_j$ and $\mu_k \in J_k$

for all j, k respectively[2]. Taylor model based verified integration is able to solve the functional dependency represented by this initial condition to produce an output flowpipe $\Pi(\boldsymbol{\lambda}, \boldsymbol{\mu}, t)$ which guarantee that for any choice of λ_j, μ_k, and t, the trajectories starting in $T(\boldsymbol{\lambda}, \boldsymbol{\mu})$ will be contained in $\Pi(\boldsymbol{\lambda}, \boldsymbol{\mu}, t)$ at time t. This allows us to treat such a flowpipe as a function $F : \mathcal{C} \times \mathbb{R}_{\geq 0} \to \widehat{\text{MIX}}$ giving us the range in the uncertainty of the output mixture of each time point given the context in which the system is placed. This then lets us define a symbolic flowpipe tree of the contextual system as follows.

Definition 8.2.3 (Symbolic Flowpipe Tree)**.** Given a flowpipe $\Pi : \mathcal{C} \times \mathbb{R}_{\geq 0} \to \text{MIX}$ and a context tree $\mathcal{N}_{\mathcal{C}}$, we may define a *symbolic flowpipe tree* for the system by

$$f(\mathcal{D}) = (F, \mathcal{D}).$$

Remark 8.2.4. In the Taylor model initial condition T we may skip any constant parameters λ_j or μ_k for which $I_j \in \mathbb{R}$ or $J_k \in \mathbb{R}$ respectively, and instead include these constants directly in the coefficients of the Taylor model. We also note that this Taylor model possesses no remainder part, since Flow* can handle this uncertainty better via symbolic parameters than the remainder interval.

In practice, this construction is complicated somewhat further by the fact that Flow* uses flowpipes which are preconditioned by replacing the coordinate space of the original system with the dynamically transformed coordinate space $[-1, 1]^n$. However, the functional dependency is still tracked by the preconditioned Taylor model, and we are able to construct a flowpipe tree based on the transformed context space, by translating subdivisions of the context to subdivisions of the flowpipe's space domain as follows.

Definition 8.2.5 (Preconditioned Symbolic Flowpipe Tree)**.** Given a preconditioned flowpipe

$$\Pi : [-1, 1]^n \times \mathbb{R}_{\geq 0} \to \widehat{\text{MIX}}$$

we may define a symbolic flowpipe tree by

$$f(\mathcal{D}) = (\Pi, g^{-1}(\mathcal{D})).$$

where the bijective map $g : [-1, 1]^n \to \mathcal{C}$ is defined by $g(\mathbf{x}) = \Pi(\mathbf{x}, 0)$.

[2]We distinguish these two sets of symbolic parameters because whilst the uncertainty in the context sets represents a universal quantification under actual possible scenarios, we do not know this is the case for the uncertainty in the base system. Indeed, when building a flowpipe tree for a contextual operator $\mathcal{C} \triangleright \varphi$ based on a Taylor model initial condition $T(\boldsymbol{\lambda}, \boldsymbol{\mu})$, each possible assignment to $\boldsymbol{\mu}$ corresponds to a real context to which the system could potentially be placed, whilst assignments to $\boldsymbol{\lambda}$ corresponds to points in the range of an over-approximated flowpipe for the system which may or may not be realized by actual trajectories of the original system, so should not be subdivided as part of context tree.

One limitation of this symbolic approach comes from the fact that symbolic subdivision will sometimes produce looser results than purely physical subdivision since flowstar is not able to symbolically track the state for contexts further up the tree (particularly in the case that trajectories bifurcate into a non-convex reach set), and may fail completely if the initial context is too large to handle. Therefore we are motivated to pursue a mixed approach whereby we start by using physical subdivision, implemented via mixed flowpipe trees.

Definition 8.2.6 (Mixed Flowpipe Tree)**.** A *mixed flowpipe tree* for model $\mathcal{M}$ over context tree $\mathcal{N}_{\mathcal{C}}$ is any flowpipe tree f over $\mathcal{N}_{\mathcal{C}}$ such that for every node $\mathcal{D} \in \mathcal{N}_{\mathcal{C}}$, exactly one of the following three cases holds:

1. $f(\mathcal{E})$ is undefined for $\mathcal{E} = \mathcal{D}$ and each of its ancestors $\mathcal{E}$;
2. $f(\mathcal{D}) = (f_{\mathcal{D}}, [-1, 1]^n)$ where $f_{\mathcal{D}}$ is a flowpipe for $\mathcal{D} \parallel \mathcal{M}$ with space domain $[-1, 1]^n$;
3. for some ancestor $\mathcal{E}$ of $\mathcal{D}$, case (2) holds and the subtree rooted at $\mathcal{E}$ is the symbolic flowpipe tree generated by $f_{\mathcal{E}}$.

Thus a mixed flowpipe tree consists of a (possibly empty) layer of nodes for which the flowpipe is undefined, followed by a layer of physical flowpipes, and finally, a layer of symbolically restricted flowpipes. In practice, this may be computed recursively, by proceeding from the top of the tree attempting to compute flowpipes for each node until at some stage we switch to symbolically subdividing the flowpipe. Whilst in general it is difficult to tell when best to switch from physical to symbolic subdivision, the most basic approach is start performing symbolic subdivision as soon as Flow* successfully produces a flowpipe for a given property for the full duration of the property which we are monitoring. This generates a *flowpipe forest* consisting of symbolic flowpipe trees each of which is based on a maximal physical flowpipe.

8.3 Masked Contextual Monitoring

Whilst signal trees give us a powerful way of exploring the truth of a proposition over a context space, their naive application has the potential for inefficiency due to the exponential increase in the number of spatial subregions as we move down a context tree, despite the fact that often only a small proportion of these are necessary to the overall verification of a given property. This can occur since, as we move down each branch of

the tree, if we naively expand each node of the tree at a given depth, we monitor the truth value of each atomic proposition at each time point from scratch on each subregion, even if it might have been possible to decide its truth value for the whole branch further up the tree. Additionally, in monitoring a signal tree for a complex proposition, we may spend a large amount of time monitoring a particular atomic proposition on regions of the context space on which it does not contribute to the truth of the overall proposition. To tackle both of these problems we are motivated to introduce *mask trees*, which specify a different mask for each node of the context tree, implementing a spatio-temporal extension of masking capable of avoiding monitoring certain time regions on whole branches of the signal tree and consequently excluding whole regions of the combined context space / time domain from the monitoring algorithm. This gives us a flexible framework to enable both *downtree masking*, propagating truth values down the signal tree to avoid recomputation of truth values decided at higher levels of the tree, and a full mask tree extension of the masked monitoring algorithm developed in Chapter 7.

8.3.1 Mask Trees and Signal Tree Monitoring Contexts

We first define a mask tree, assigning a mask to each node of a context tree.

Definition 8.3.1. Given a context tree $\mathcal{N}_{\mathcal{C}}$ a *mask tree* over $\mathcal{N}_{\mathcal{C}}$ is a monotone function $m : \mathcal{N}_{\mathcal{C}} \to \mathbb{R}_{\geq 0} \to \mathbb{B}$ under implication order $\geq$; that is: for any $\mathcal{D}, \mathcal{E} \in \mathcal{N}_{\mathcal{C}}$ such that $\mathcal{D} \supseteq \mathcal{E}$, $m(\mathcal{D})(t) \Leftarrow m(\mathcal{E})(t)$ for all $t \in \mathbb{R}_{\geq 0}$.

We can then use a mask tree on an existing signal tree to recursively restrict the signal at each node of the tree.

Definition 8.3.2. Given a signal tree $s : \mathcal{N}_{\mathcal{C}} \to \mathbb{R}_{\geq 0} \to \mathbb{T}$ and a mask tree $m : \mathcal{N}_{\mathcal{C}} \to \mathbb{R}_{\geq 0} \to \mathbb{B}$ we define the *restriction of s to m* as the signal tree $s|_m : \mathcal{N}_{\mathcal{C}} \to \mathbb{R}_{\geq 0} \to \mathbb{T}$ defined by

$$s|_m(\mathcal{D}) = s(\mathcal{D})|_{m(\mathcal{D})} \quad \text{for all } \mathcal{D} \in \mathcal{N}_{\mathcal{C}}.$$

We will now proceed to extend our masked monitoring algorithm to signal tree monitoring. This then motivates us to define signal tree monitoring contexts which will play the same role as monitoring contexts in determining the suitability of mask trees for monitoring each operator of STL.

Definition 8.3.3. Given a context tree $\mathcal{N}_{\mathcal{C}}$, a *signal tree monitoring context over* $\mathcal{N}_{\mathcal{C}}$ is

defined according to the grammar:

$$\begin{aligned}\mathcal{S}([\cdot]) \;::=\; & [\cdot] \;\mid\; s \vee \mathcal{S}([\cdot]) \;\mid\; s \wedge \mathcal{S}([\cdot]) \;\mid\; \neg\, \mathcal{S}([\cdot]) \\ & \mid\; \mathcal{F}_I(\mathcal{S}([\cdot])) \;\mid\; \mathcal{G}_I(\mathcal{S}([\cdot])) \;\mid\; s\, \mathcal{U}_I\, \mathcal{S}([\cdot])\,.\end{aligned}$$

where each s is a signal tree over $\mathcal{N}_{\mathcal{C}}$.

The following definition allows us to interpret a signal tree monitoring context as a tree of monitoring contexts for each node in its context tree.

Definition 8.3.4. Given a signal tree monitoring context $\mathcal{S}([\cdot])$ over a context tree $\mathcal{N}_{\mathcal{C}}$, we define a monitoring context $\mathcal{S}_{\mathcal{D}}$ for each $\mathcal{D} \in \mathcal{N}_{\mathcal{C}}$ by the following rules:

$$\begin{aligned}[\cdot]_{\mathcal{D}} &\equiv [\cdot] \\ (s \vee \mathcal{S}([\cdot]))_{\mathcal{D}} &\equiv s(\mathcal{D}) \vee \mathcal{S}_{\mathcal{D}}([\cdot]) \\ (s \wedge \mathcal{S}([\cdot]))_{\mathcal{D}} &\equiv s(\mathcal{D}) \wedge \mathcal{S}_{\mathcal{D}}([\cdot]) \\ \mathcal{F}_I(\mathcal{S}([\cdot]))_{\mathcal{D}} &\equiv \mathcal{F}_I(\mathcal{S}_{\mathcal{D}}([\cdot])) \\ \mathcal{G}_I(\mathcal{S}([\cdot]))_{\mathcal{D}} &\equiv \mathcal{G}_I(\mathcal{S}_{\mathcal{D}}([\cdot])) \\ (s\, \mathcal{U}_I\, \mathcal{S}([\cdot]))_{\mathcal{D}} &\equiv s(\mathcal{D})\, \mathcal{U}_I\, \mathcal{S}_{\mathcal{D}}([\cdot]).\end{aligned}$$

This then leads to define when a mask tree is sufficient or optimal for monitoring in a given masked monitoring context.

Definition 8.3.5. Given a context tree $\mathcal{N}_{\mathcal{C}}$, a mask tree $m : \mathcal{N}_{\mathcal{C}} \to \mathbb{R}_{\geq 0} \to \mathbb{B}$ is *sufficient for a signal tree monitoring context* $\mathcal{S}([\cdot])$ *under a mask tree* $n : \mathcal{N}_{\mathcal{C}} \to \mathbb{R}_{\geq 0} \to \mathbb{B}$ if for any signal tree $s : \mathcal{N}_{\mathcal{C}} \to \mathbb{R}_{\geq 0} \to \mathbb{B}$ we have that $\mathcal{S}(s)|_n = \mathcal{S}(s|_m)|_n$.

Furthermore, we say that m is *optimal* for $\mathcal{S}([\cdot])$ under n if m is the least sufficient mask for $\mathcal{S}([\cdot])$ under n with respect to the implication order $\Rightarrow$.

We now has the following result, that tells us that sufficiency and optimality of a contextual signal correspond to nodewise sufficiency and optimality over the context tree.

Proposition 8.3.6. *Given any mask tree m and any signal tree monitoring context $\mathcal{S}([\cdot])$, m is sufficient / optimal for $\mathcal{S}([\cdot])$ iff for all $\mathcal{D} \in \mathcal{N}_{\mathcal{C}}$, $m(\mathcal{D})$ is sufficient / optimal for $\mathcal{S}_{\mathcal{D}}([\cdot])$ for all $\mathcal{D} \in \mathcal{N}_{\mathcal{C}}$.*

Proof.

Sufficiency

$$
\begin{aligned}
& m \text{ is sufficient for } \mathcal{S}([\cdot]) \text{ under mask tree } n && (8.6)\\
\iff\ & \text{for every signal tree } s, && (8.7)\\
& \quad \mathcal{S}(s)|_n = \mathcal{S}(s|_m)|_n && (8.8)\\
\iff\ & \text{for every signal tree } s, \text{ for all } \mathcal{D} \in \mathcal{N}_{\mathcal{C}}, && (8.9)\\
& \quad \mathcal{S}(s)|_n(\mathcal{D}) = \mathcal{S}(s|_m)|_n(\mathcal{D}) && (8.10)\\
\iff\ & \text{for every signal tree } s, \text{ for all } \mathcal{D} \in \mathcal{N}_{\mathcal{C}}, && (8.11)\\
& \quad \mathcal{S}(s)(\mathcal{D})|_{n(\mathcal{D})} = \mathcal{S}\Big(s(\mathcal{D})|_{m(\mathcal{D})}\Big)\Big|_{n(\mathcal{D})} && (8.12)\\
\iff\ & \text{for all } \mathcal{D} \in \mathcal{N}_{\mathcal{C}}, \text{ for every signal } w, && (8.13)\\
& \quad w|_{n(\mathcal{D})} = \mathcal{S}\Big(w|_{m(\mathcal{D})}\Big)\Big|_{n(\mathcal{D})} && (8.14)\\
\iff\ & \text{for all } \mathcal{D} \in \mathcal{N}_{\mathcal{C}}, && (8.15)\\
& \quad m(\mathcal{D}) \text{ is sufficient for } \mathcal{S}([\cdot]) \text{ under } n(\mathcal{D}). && (8.16)
\end{aligned}
$$

In the above, the key biimplication Eq. (8.13) follows trivially in the $\Leftarrow$ direction, and for the $\Rightarrow$ direction, if we have any signal w with

$$\mathcal{S}_{\mathcal{D}}\Big(w\Big)\Big|_{n(\mathcal{D})} \neq \mathcal{S}_{\mathcal{D}}\Big(w|_{m(\mathcal{D})}\Big)\Big|_{n(\mathcal{D})},$$

then the signal tree s defined by

$$s(\mathcal{E})(t) = \begin{cases} w(t) & \text{if } \mathcal{E} \subseteq \mathcal{D} \\ \mathbf{U} & \text{otherwise} \end{cases}$$

has that

$$\mathcal{S}_{\mathcal{D}}\Big(s(\mathcal{D})\Big)\Big|_{n(\mathcal{D})} = \mathcal{S}_{\mathcal{D}}\Big(w\Big)\Big|_{n(\mathcal{D})} \neq \mathcal{S}_{\mathcal{D}}\Big(w|_{m(\mathcal{D})}\Big)\Big|_{n(\mathcal{D})} = \mathcal{S}_{\mathcal{D}}\Big(s(\mathcal{D})|_{m(\mathcal{D})}\Big)\Big|_{n(\mathcal{D})}$$

proving the impliction by contrapositive.

Optimality If m is optimal, then for any $\mathcal{D} \in \mathcal{N}_{\mathcal{C}}$, if we have some sufficient mask l with $m(\mathcal{D}) \Rightarrow l$, we can define a mask tree m' by

$$m'(\mathcal{E}) = \begin{cases} m(\mathcal{E}) \wedge l & \text{if } \mathcal{E} = \mathcal{D} \\ m(\mathcal{E}) & \text{otherwise.} \end{cases}$$

But then $m \Rightarrow m'$, m' is sufficiency by the first half of the proposition, and hence $m = m'$ by the optimality of m.

Conversely, suppose that $m(\mathcal{D})$ is optimal for all $\mathcal{D}$ and take any sufficient mask tree m' with $m \Rightarrow m'$. Then for any $\mathcal{D} \in \mathcal{N}_{\mathcal{C}}$, $m(\mathcal{D}) \Rightarrow m'(\mathcal{D})$ and $m'(\mathcal{D})$ is sufficient by the first half of the proposition, so by the optimality of $m(\mathcal{D})$ we must have $m'(\mathcal{D}) = m(\mathcal{D})$. But then $m' = m$, proving the optimality of m. □

This directly allows us to define optimal and sufficient masks for each context within the contextual monitoring process by reinterpreting the masking operators $m_s^\wedge$, $\mathcal{P}_K$, etc from Chapter 7 over context trees, by applying the appropriate masking rules to each node of a signal tree.

8.3.2 Masked Signal Tree Monitoring Algorithm

We are now ready to draw upon our existing monitoring rules from Section 7.5 and Section 8.2 to define the contextual monitoring algorithm for $\mathcal{LBUC}$. This consists of the following rules for the standard STL operators:

$$\texttt{signalTreeMF}(\varphi, f, \mathcal{A}, m)(\mathcal{D}) \triangleq \texttt{signalM}(\varphi, f(\mathcal{D}), \mathcal{A}, m(\mathcal{D}))$$

$$\texttt{signalTreeMF}(\neg\varphi, f, \mathcal{A}, m) \triangleq \neg\texttt{signalTreeMF}(\varphi, f, \mathcal{A}, m)$$

$$\texttt{signalTreeMF}(\varphi \wedge \psi, f, \mathcal{A}, m) \triangleq s_\varphi \wedge s_\psi \quad \text{where} \begin{cases} s_\varphi \triangleq \texttt{signalTreeMF}\Big(\varphi, f, \mathcal{A}, m\Big) \\ s_\psi \triangleq \texttt{signalTreeMF}\Big(\psi, f, \mathcal{A}, m_{s_\varphi}^\wedge\Big) \end{cases}$$

$$\texttt{signalTreeMF}(\varphi \vee \psi, f, \mathcal{A}, m) \triangleq s_\varphi \vee s_\psi \quad \text{where} \begin{cases} s_\varphi \triangleq \texttt{signalTreeMF}(\varphi, f, \mathcal{A}, m) \\ s_\psi \triangleq \texttt{signalTreeMF}\Big(\psi, f, \mathcal{A}, m_{s_\varphi}^\vee\Big) \end{cases}$$

$$\texttt{signalTreeMF}(\mathcal{F}_K\,\varphi, f, \mathcal{A}, m) \triangleq \mathcal{F}_K(\texttt{signalTreeMF}(\varphi, f, \mathcal{A}, \mathcal{P}_K\,m))$$

$$\texttt{signalTreeMF}(\mathcal{G}_K\,\varphi, f, \mathcal{A}, m) \triangleq \mathcal{G}_K(\texttt{signalTreeMF}(\varphi, f, \mathcal{A}, \mathcal{P}_K\,m))$$

$$\texttt{signalTreeMF}(\varphi\,\mathcal{U}_{[a,b]}\,\psi, f, \mathcal{A}, m) \triangleq s_\varphi\,\mathcal{U}_{[a,b]}\,s_\psi \quad \text{where} \begin{cases} s_\varphi \triangleq \texttt{signalTreeMF}(\varphi, f, \mathcal{A}, m) \\ s_\psi \triangleq \texttt{signalTreeMF}\Big(\psi, f, \mathcal{A}, \mathcal{H}_{[0,a]}\Big(m_{s_\varphi}^\wedge\Big)\Big) \end{cases}$$

The masked signal tree $\texttt{signalTreeMF}\Big(\mathcal{D} \triangleright \varphi, f, \mathcal{A}, m\Big)$ for a context operator $\mathcal{D} \triangleright \varphi$ can be monitored in a similar way to the unmasked signal tree except that at each node we must apply the mask to restrict the time domain for the initial jump and recursively use masked monitoring for the inner property φ. Thus we may compute the masked signal tree's component signal $\texttt{signalTreeMF}\Big(\mathcal{D} \triangleright \varphi, f, \mathcal{A}, m\Big)(\mathcal{E})$ on each node $\mathcal{E}$ of the context

tree as follows. If firstly we assume that the corresponding node of the mask tree is unitary so that $m(\mathcal{E}) = (T)$ for some time interval T, then we may define the uncertain bond-calculus model $\Phi_{\mathcal{E},T}$ and the derived context tree $\mathcal{N}_{\mathcal{D},\mathcal{E},T}$ as in the unmasked case. Then we recursively perform *masked contextual monitoring* of φ over $\mathcal{N}_{\mathcal{D},\mathcal{E},T}$ using the singleton initial mask tree (0) (that is, the mask tree which is 0 all the way down) to compute the inner signal tree s and its n^{th} refined signal s^n. Thus as in the unmasked case we may find the signal value on T and bisect T to give the signal $s_T \triangleq \texttt{signalTreeMF}(\mathcal{D} \triangleright \varphi, f, \mathcal{A}, m)(\mathcal{E})$. Now, for a general mask $m = (T_i)_i$ we may decompose it into unitary segments $m = \bigvee_i (T_i)$ and use this to compute

$$\texttt{signalTreeMF}(\mathcal{D} \triangleright \varphi, f, \mathcal{A}, m)(\mathcal{E}) \triangleq \bigcup_i s_{T_i}$$

giving us the required masked signal tree in the general case.

As a final application of masking to the contextual monitoring process, we can apply appropriate masks recursively down the signal tree so that at each level we only attempt monitoring on the time region on which the signal has not been determined further up the tree. To this end we define the *unknown mask* of a signal.

Definition 8.3.7. Given a signal s, the *unknown mask* of s is defined by $m_s^{\mathbf{U}} = m_s^{\wedge} \wedge m_s^{\vee}$. That is, $m_s^{\mathbf{U}}$ is defined such that $m_s^{\mathbf{U}}(t) = \mathbf{T}$ iff $s(t) = \mathbf{U}$.

Now, given any node $\mathcal{E} \in \mathcal{N}$ with parent $\mathcal{D}$, this allows us to use the following revised rule for the signal tree for φ at $\mathcal{E}$

$$\texttt{signalTreeMF}(\varphi, f, \mathcal{A}, m)(\mathcal{E}) = s_{\mathcal{D}} \cup \texttt{signalTreeMF}\Big(\varphi, f, \mathcal{A}, m \wedge m_s^{\mathbf{U}}\Big)(\mathcal{E})$$

where $s_{\mathcal{D}} \triangleq \texttt{signalTreeMF}(\varphi, f, \mathcal{A}, m)(\mathcal{D})$. This allows us to use the knowledge from the $s_{\mathcal{D}}$ of the parent in computing the signal at node $\mathcal{E}$, since $s_{\mathcal{D}}$ is also signal for φ at node $\mathcal{E}$, and hence we may restrict the monitoring at node $\mathcal{E}$ by the mask $m_s^{\mathbf{U}}$ which accounts for the part of the signal we already know. Applying this rule at each stage of signal tree expansion allows us to compute the whole signal tree recursively, avoiding repeating monitoring at any point of the time domain once its truth value has first been found on a given branch of the context space. Overall, this makes for a much more adaptive contextual monitoring algorithm since the masks for each depth will shrink as we move down the tree so that only the trickiest regions of the context space will be monitored at higher depths with nonempty masks.

8.4 Demonstration and Performance Evaluation

In this section we will demonstrate the use of context signals to monitor properties and systems involving ranges of uncertain parameters to a greater precision than is possible using three-valued signals which quantify over the full range of uncertainty in the system. We will return to the Predator-Prey Role Reversal model from Section 6.5 to demonstrate and measure the performance of contextual monitoring for both refining the signal over relatively small ranges of uncertainty and for exploring the truth of a propositions over large uncertain regions which lie outside the scope of our previous model-checking algorithms.

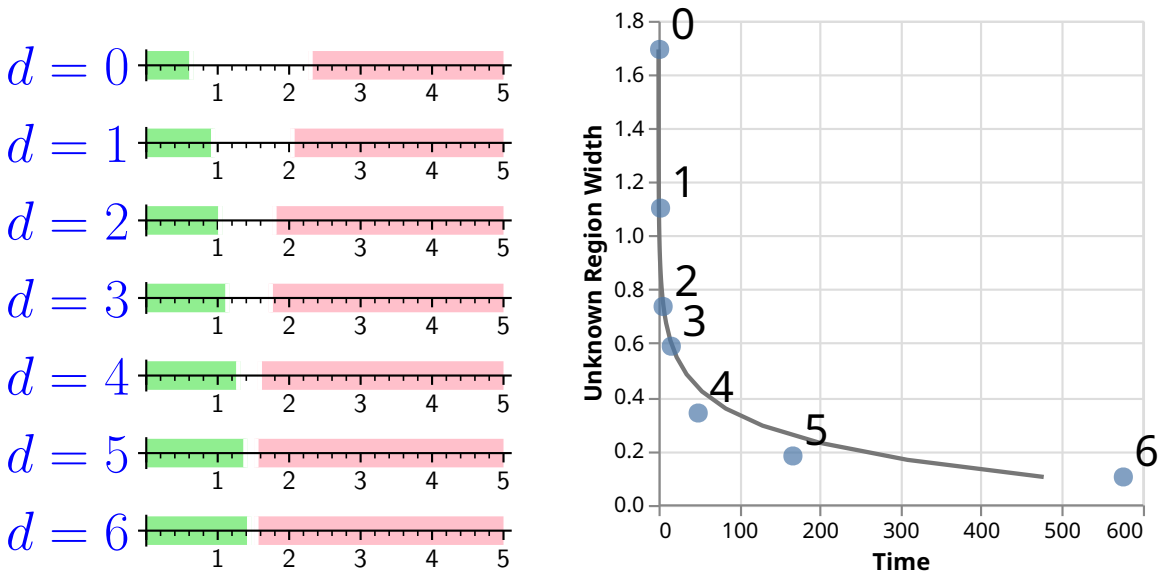
8.4.1 Context Signal Refinement

We begin by considering the small box of uncertain initial conditions $\Pi_2 \equiv [1.0, 1.2]$ WHELK$\|$ $[4, 6]$ LOBSTER and the property,

$$\varphi_3 \triangleq \mathcal{F}_{[0.1,0.2]}(\neg P)$$

which states that we will be in the elliptical region of the phase space described by P at some point between 0.1 and 0.2 time units in the future. We can use contextual monitoring to generate a signal tree for φ_3 over the space of uncertain contexts generated by the initial conditions Π_2, which may be lazily expanded to produce a refined signal for the property at a variety of depths d, as shown in Fig. 8.1a. If we expand it at depth $d = 0$, this produces a standard signal for φ_3 which matches the result of our normal verified monitoring procedure, whilst greater depths yield progressively more precise refined signals. Fig. 8.1b shows the relationship between the decreasing uncertainty of the refined signal (as measured by the width of the unknown region) and the overall monitoring time. We can see an inverse exponential relationship between the unknown region width and the monitoring time demonstrated in an exponential regression curve. This shows how the first few levels of refinement can rapidly improve the quality of the signal whilst it becomes increasingly difficult to completely eliminate the remaining uncertainty by expanding the lower levels of the tree.

As well as improving the quality of the overall signal, contextual monitoring gives us a richer view of how the truth of the proposition various over the parameter space at different moments in time. At an instant in time, the signal tree provides regions of the parameter space for which the property is definitely true or false. We may also visualize the structure of the signal tree as a two-level nested tree map including black boxes for the levels of the tree at which each physical flowpipe is computed, and grey boxes for

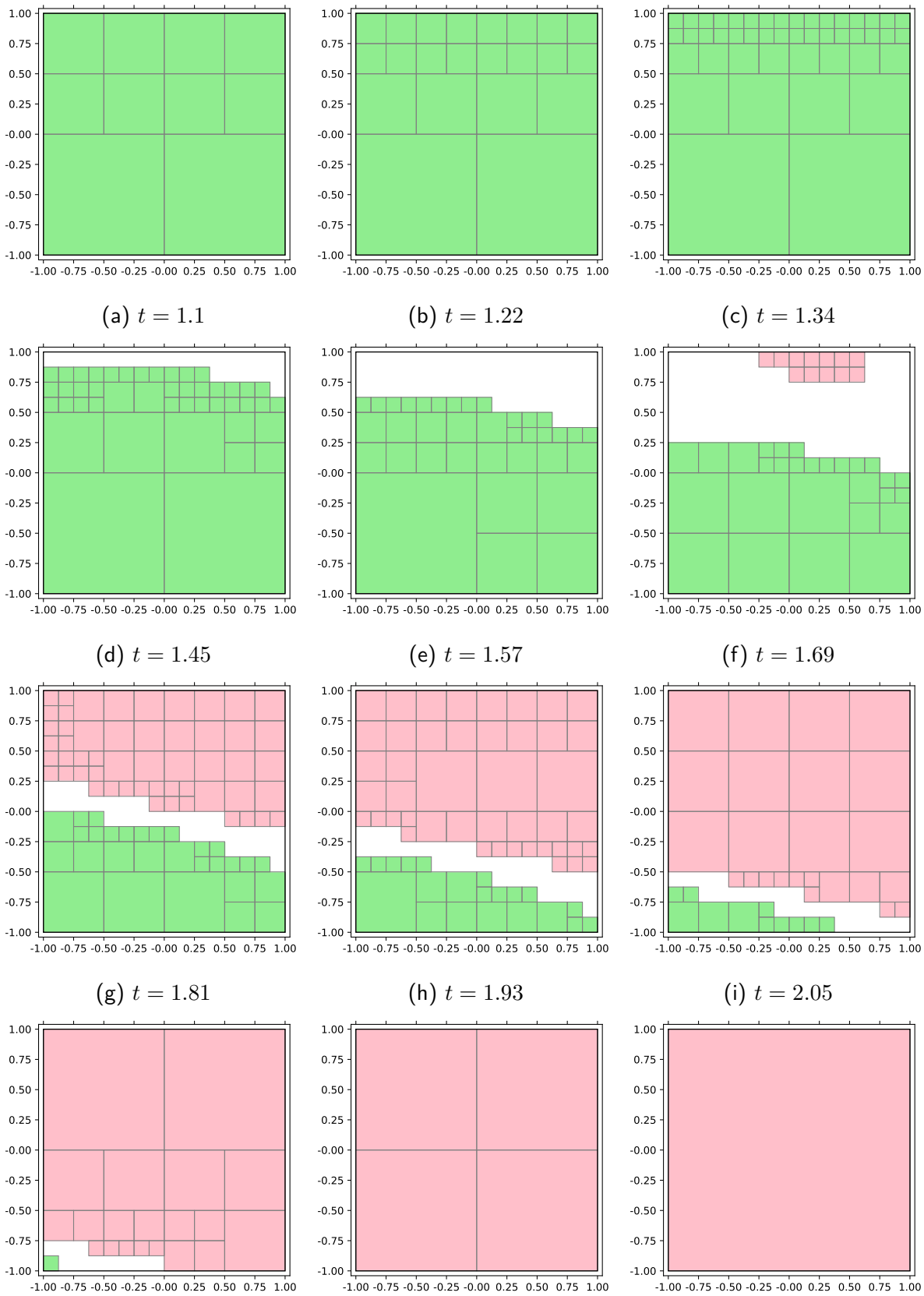


(a) The refined signals for φ_3 at depths d.

(b) Scatter plot of total monitoring time against unknown region width at each depth.

Figure 8.1: Results of contextual monitoring for φ_3 given uncertain initial mixture Φ_2.

the level of symbolic subdivision at which the signal value is first determined. In Fig. 8.2 we can apply this visualization to a sequence of different time slices of the signal tree. For example, the time point $t = 1.81$ is illustrated in Fig. 8.2g which shows that a single flowpipe covers the whole parameter space, whilst inside of this symbolic subdivision at a variety of depths determines a region for which the property is true in the south-west of the diagram (coloured green) and a region for which it is true in the north-east (coloured red), with an unknown region in the middle. Overall, Fig. 8.2 shows the evolution of this spatial configuration over time from $t = 1.1$ (Fig. 8.2a) where the signal is true over the whole space to $t = 2.4$ (Fig. 8.2l) where the signal is false over the whole space; this sequence corresponds to the refined signals shown in Fig. 8.1a. It is worth noting that the improvement in these refined signals comes from two distinct sources: the true interval of the signal expands because subdividing the space increases the precision of verified monitoring, allowing the signal to be determined on each region whereas the overall region is too large to monitor in one go (as in Fig. 8.2a), whereas the false region of the signal expands because we can determine the overall falsehood of the property as soon as we find one subregion on which it fails (as in e.g. Fig. 8.2f).



(a) $t = 1.1$ (b) $t = 1.22$ (c) $t = 1.34$

(d) $t = 1.45$ (e) $t = 1.57$ (f) $t = 1.69$

(g) $t = 1.81$ (h) $t = 1.93$ (i) $t = 2.05$

(j) $t = 2.16$ (k) $t = 2.28$ (l) $t = 2.40$

Figure 8.2: Slices of Signal Tree for φ_3 over time.

8.4.2 Nested Contextual Refinement

As we have seen in the previous subsection, refinement offers a way of improving the quality of monitoring results by subdividing the context set to explore different regions of the context space. This applies not only to systems with uncertainty in the initial conditions of the overall bond-calculus model but also to uncertain context operators embedded as part of a larger logical property. Moreover, for a given property there may be different levels of contextual uncertainty which we may independently refine. For example, suppose we want to monitor the contextual property,

$$\varphi_4 \triangleq [0,2]\,\textsc{Lobster} \triangleright \mathcal{F}_{[0,5]}\,\mathcal{G}_{[0,5]}\,P$$

for 5 time units over the context space defined via the set of uncertain initial conditions $\Pi_2 \equiv [1.0, 1.2]\,\textsc{Whelk} \parallel [4,6]\,\textsc{Lobster}$.

In order to improve the precision of monitoring, we can attempt to refine the *outer* context Π_2 or the *inner* context $[0,2]\,\textsc{Lobster}$. We measure the tightness of monitoring by the least time point $t = k$ for which the signal is true (or equivalently, the least k for which the signal is sufficient to verify the property $\mathcal{F}_{[0,k]}\,\varphi_4$), whilst the time cost of monitoring will depend of the levels of outer and inner context refinement applied. Table 8.1 examines this result of this trade-off by monitoring the property with different levels of outer and inner context refinement. Firstly, in Table 8.1a we see that refining both the outer and inner context results in an increase in the precision of monitoring and a corresponding decrease in k, however, the greatest impact comes with a combination of outer and inner refinement. In Table 8.1b we see that the total verification time also scales with the the combined outer and inner refinement level, leading to a rather large

Level	0	1	2
0	4.7675	4.6112	4.6112
1	4.6112	4.1423	4.1423
2	4.5331	3.986	3.9078

(a) Least known $\mathbf{T}$ value (k) in signal.

Level	0	1	2
0	147.68	123.08	405.22
1	429.63	381.81	1,089.16
2	972.21	822.72	2,046.74

(b) Total verification time (sec).

Table 8.1: Monitoring results and times for property under different levels of context refinement. The level of refinement for the inner context increase along rows, whilst level of refinement for the initial conditions increases down columns. All runs were carried out with context refinement parameter $\varepsilon = 0.1$.

increase in total verification time when we combine 2 levels of refinement corresponding to the outer and inner context respectively. We do however see that 1 level of inner and outer refinement gives us most of the improvement in monitoring precision at a much more reasonable cost, and is more effective than outer or inner refinement alone. We also see that this increase in total verification time is not always monotonic as observed when moving from 0^{th} to 1^{st} level refinement of the outer context; this can be explained by the fact that in this case refining the outer context appears to provide a more cost-effective increase in monitoring precision the adaptive refinement of the time-domain of the inner context operator it avoids. We also note that these costs depend heavily on the property at hand; whilst we pay the full cost to improve the monitoring result whose satisfaction hangs in the balance (with large regions of uncertainty in the time-domain), the combination of masking and adaptive context signal computation prevent us from paying these costs when refinement is not necessary to give an unambiguous monitoring result.

8.4.3 Context Space Exploration

As well as using contextual signal to produced signal over a context space representing a reasonably small interval of uncertainty around a fixed set of initial conditions, we can also explore how the behaviour of the system varies over a much wider range of initial conditions, to determine the behaviour of the system in different regions of the parameter space. To this end we consider the uncertain initial mixture,

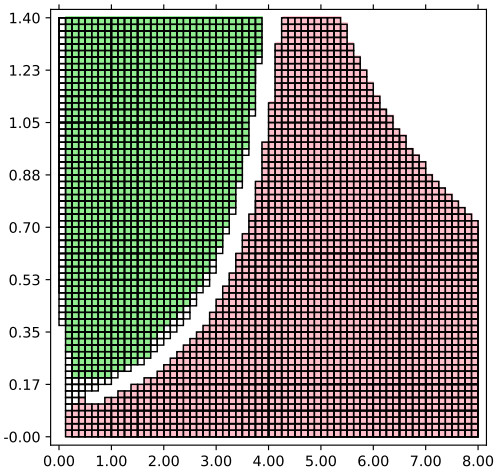
$$\Pi_3 \triangleq [0, 1.4]\,\textsc{Whelk} \parallel [0, 8]\,\textsc{Lobster}.$$

As demonstrated by the system's phase portrait (Fig. 6.6) this covers a wide range of qualitatively different behaviours (encompassing both boxes of initial conditions Π_1 and Π_2 considered in Section 6.5). This makes it difficult to monitor non-trivial logical propositions over the whole context space since each atomic propositions is universally quantified over the whole context space, defeating the ability of complex logical propositions to handle properties which are true for different reasons in different regions of the context space. Moreover, Flow* is unable to construct a single Flowpipe covering the whole context space as the over-approximation error rapidly blows up over time. These difficulties motivate us to apply the techniques of this chapter and use flowpipe trees and signal trees for the properties φ_1 and φ_2 monitored under a variety of parameter sets (Fig. 8.3) to give a more refined view of the behaviour of the system over the context space.

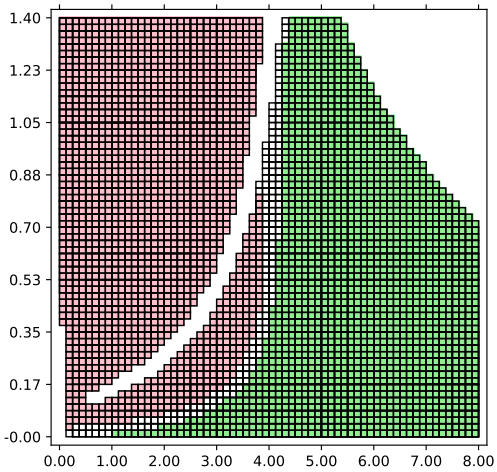
Firstly, as illustrated in Figs. 8.4a and 8.4b, we may use a physical flowpipe tree which computes a new flowpipe for each node of the tree at our desired depth of expansion. We

	Parameter Set		
	p_1	p_2	p_3
Step Size	$5 \cdot 10^{-2}$	$4 \cdot 10^{-2}$	$3 \cdot 10^{-2}$
Order	6	7	8
Remainder Estimation	$1 \cdot 10^{-4}$	$1 \cdot 10^{-5}$	$1 \cdot 10^{-6}$
Cutoff Threshold	$1 \cdot 10^{-5}$	$1 \cdot 10^{-6}$	$1 \cdot 10^{-7}$

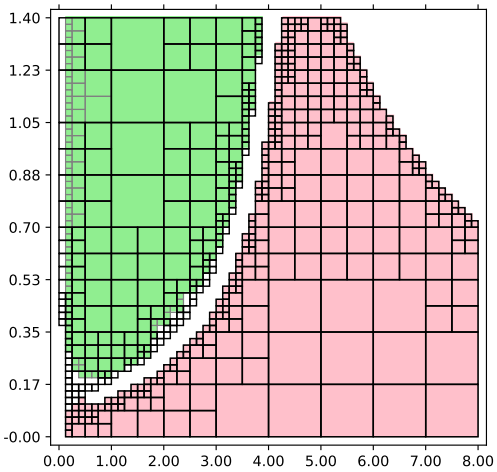
Figure 8.3: Different Flow* parameter sets for contextual monitoring.



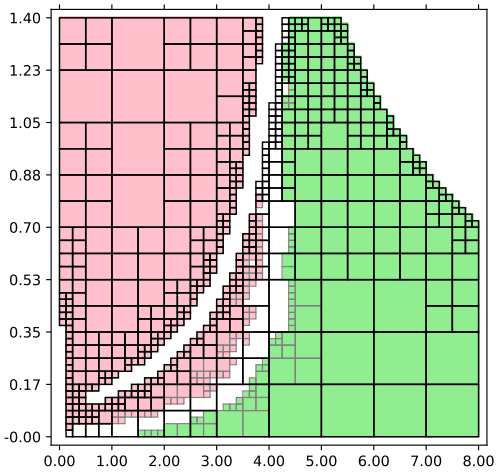
(a) Property φ_1 with physical Flowpipe tree.



(b) Property φ_2 with physical Flowpipe tree.



(c) Property φ_1 with mixed Flowpipe tree.



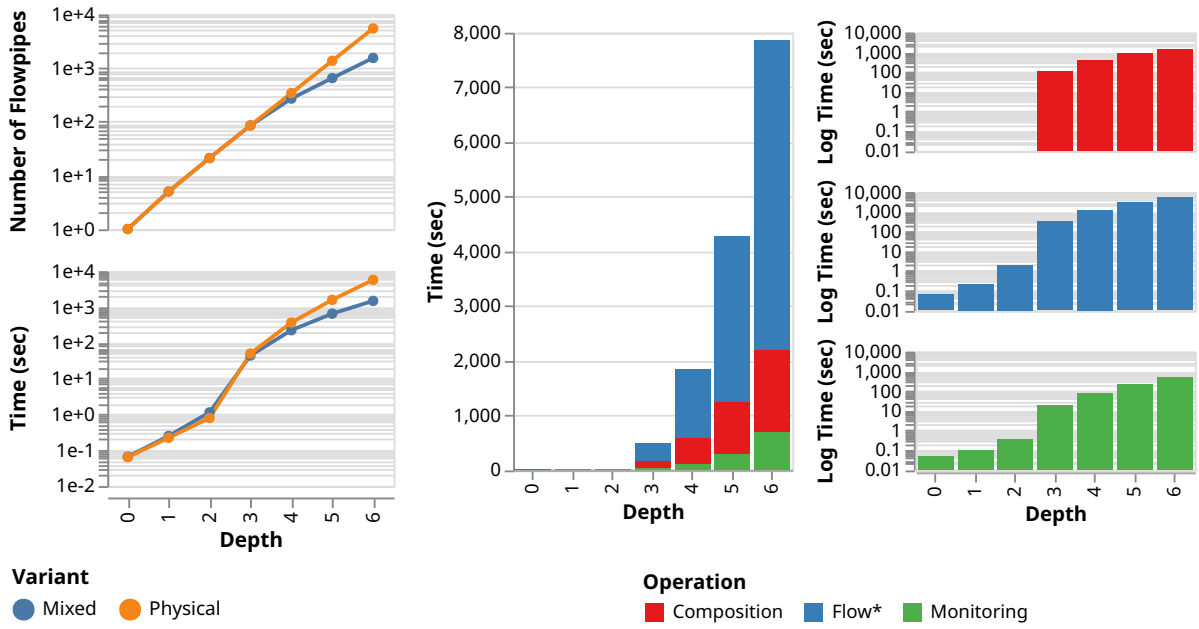
(d) Property φ_2 with mixed Flowpipe tree.

Figure 8.4: Signal Trees slice for properties φ_1 and φ_2 based on physical and symbolic Flowpipe trees under Flow* parameter set p_1.

can see that this gives a reasonably precise view of the behaviour of the system over the whole phase space, although in some regions such as the north-east corner of the context space and along the main separatrix of the phase portrait, Flow* was unsuccessful and did not produce a flowpipe, so no monitoring results were recorded. We are also able to use a mixed flowpipe tree, as shown in Figs. 8.4c and 8.4d, switching from physical to symbolic subdivision to refine the signal as soon as a flowpipe is first successfully computed. We firstly note that many of the flowpipes of the mixed signal tree are expanded at a higher level of the tree, with symbolic subdivision being used to determine the value of the signal for their children, avoiding the need to recompute the whole flowpipe. We can see that for φ_1 the monitoring results are almost as good as the physical flowpipe tree, whilst for φ_2 we lose a little more precision in the centre of the phase portrait. This loss of precision occurs when Flow* succeed for a given region but the flowpipes produced do not always provide enough precision for monitoring; this could be improved via more sophisticated heuristics for choosing when to switch from physical to symbolical subdivision during the construction of the mixed flowpipe tree.

These visual properties of the signal tree are borne out when we analyse the computational cost of the verification procedure. In Fig. 8.5a we see that as we increase the depth at which we expand the signal tree, the number of flowpipe computations and the consequent overall monitoring time increases exponentially, whilst the mixed flowpipe tree can result in an order of magnitude reduction in the number of flowpipe computations and the consequent overall monitoring time (the exact number of flowpipes computations required in the mixed tree depends heavily on the dynamics of the system at hand and may often be finite as in Section 8.4.1). The growth of the cost of each stage of the monitoring process is shown in Fig. 8.5b. We see that, as in Section 6.4, the Flow* verified integration and the associated composition of the preconditioned Taylor models comprised the majority of the total verification time, each taking an order of magnitude longer than the actual monitoring process. However, the exponential growth coefficient of the Flow* verification time noticeably levels off at greater depths due to the combination of downtree masking and symbolic subdivision.

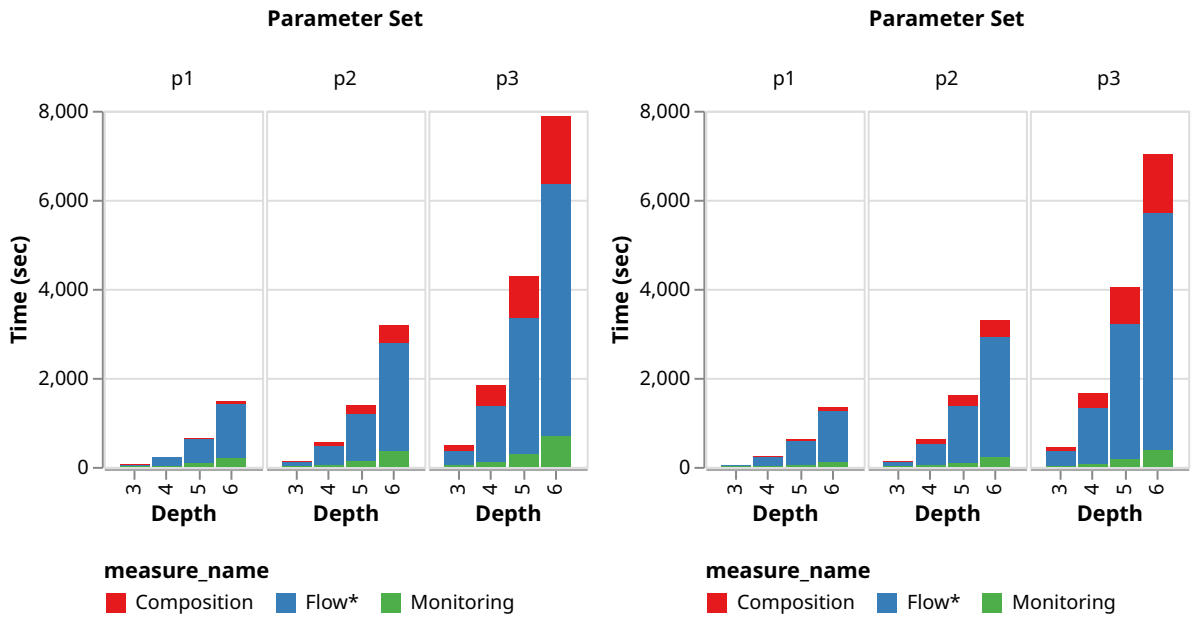
In addition to the depth of expansion, we can also improve the precision of monitoring by changing the parameters of the underlying Flow* verified integration process, such as the step size and Taylor model polynomial order. In order to investigate how these parameters interact with context signal refinement, we will compare the results of contextual monitoring in three different parameter sets (Fig. 8.3): the parameter set p_1 used thus far in this section and two additional parameter sets, p_2 and p_3, which successively



(a) Time and Number of Flowpipes for different monitoring algorithm variants.

(b) Breakdown of overall verification time by operation.

Figure 8.5: Monitoring costs for property φ_1 under as expansion depth increases.

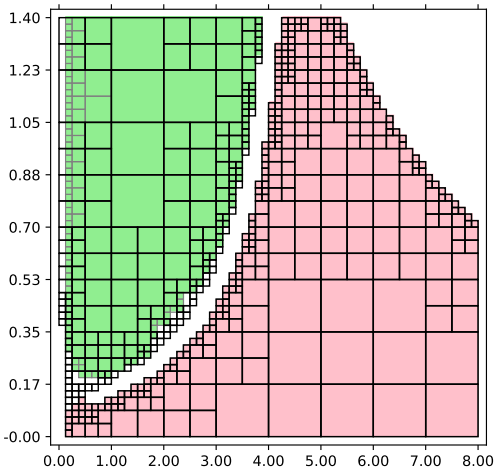


(a) Property φ_1.

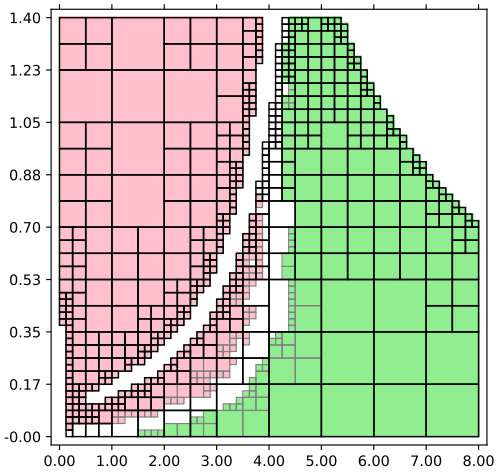
(b) Property φ_2.

Figure 8.6: Variation of masked monitoring costs for different parameter sets.

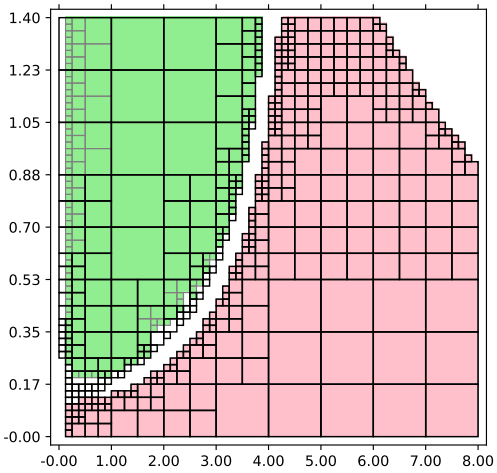
refine the key integration parameters to flowpipe over-estimation error. We can see the context signals resulting from the mixed flowpipe trees generated using each of these different parameter sets in Fig. 8.7. Comparing each parameter set, we see that whilst the overall picture is consistent the parameter sets p_2 and p_3 are often able to give the same monitoring results as p_1 at a higher level of the context tree (visible in the presence of fewer, larger black boxes in e.g. the north-west corner the context space as we move down each column of Fig. 8.7) whilst we are also able to determine the truth value on new regions of the context space. This increase in precision does, however, come at the cost of greater time costs, as evident in Fig. 8.6. This shows that each refined parameter set comes with a substantial increase in cost, whilst delivering a moderate increase in coverage of the context space, not dissimilar to each level of context tree refinement. Moreover, some regions of the context space remain unexplored using either highly refined parameters without context refinement or at higher depths of context refinement with unrefined Flow* parameters, suggesting that for the most precise view of the behaviour of the system we need both context tree refinement and well chosen Flow* integration parameters. We also note that both parameter sets p_2 and p_3 used with symbolic subdivision give a considerably better overall coverage of the context space than p_1 combined with physical subdivision in 56% and 138% of the time cost respectively.



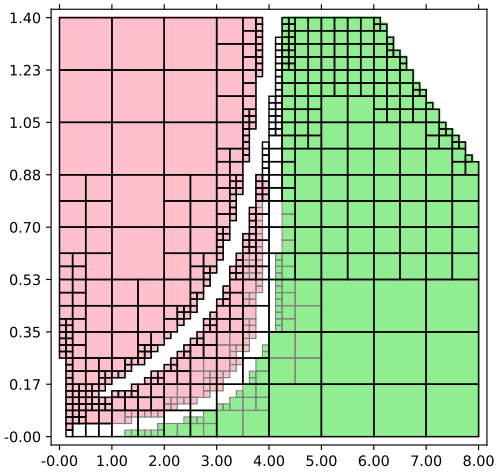
(a) Property φ_1 under parameter set p_1.



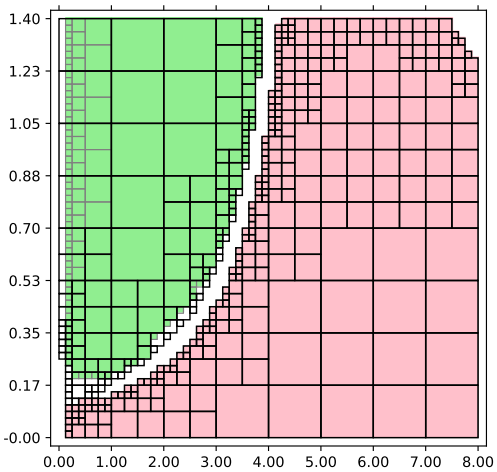
(b) Property φ_2 under parameter set p_1.



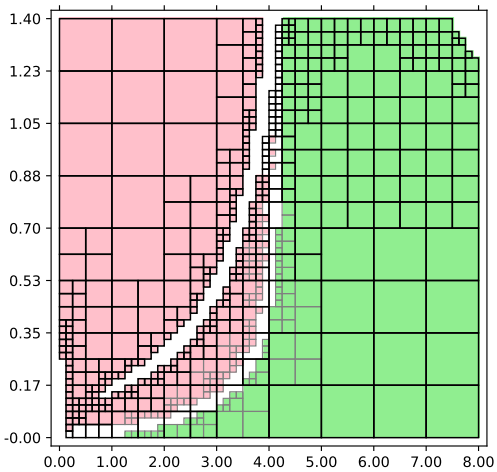
(c) Property φ_1 under parameter set p_2.



(d) Property φ_2 under parameter set p_2.



(e) Property φ_1 under parameter set p_3.



(f) Property φ_2 under parameter set p_3.

Figure 8.7: Signal Trees slice for properties φ_1 and φ_2 based on Flow* verified integration with parameter sets p_1, p_2, and p_3, over context space given by the uncertain initial mixture Π_3.

Chapter 9

Implementation Overview

In this chapter we will give a brief overview of our implementations of tools for the bond-calculus and $\mathcal{LBUC}$. In Section 9.1 we describe our implementation of the bond-calculus language. In Section 9.2 we describe our integration of the bond-calculus and $\mathcal{LBUC}$ with Sagemath and Python and their use in an interactive browser-based Jupyter environment. In Section 9.3 we describe our interface to Flow*'s C++ API and our use of Cython. Finally, in Section 9.4 we briefly describe our methodology for benchmarking and testing.

Both of these tools are open source, licensed under the GPLv3: the bond-calculus implementation is available online at[1], whilst $\mathcal{LBUC}$ is available at[2] along with many of the models from this thesis.

9.1 Bond-Calculus Language

The core bond-calculus language is implemented as collection of Haskell libraries and exposed as an interactive text-based environment in the `bondwb` tool similarly to the existing `cpiwb`[3] tool for Continuous π. By using Haskell's type system we are able to define bond-calculus models their semantics generically over different base numerical types following an appropriate type class. Thus we are able to use different instantiations of this semantics including a numerical semantics, a parametric symbolic semantics, and the interval semantics.

Models are represented and stored textually in `.bond` files so, for example, the role-reversal model from Section 6.5 is captured by the bond-calculus model,

[1] `https://github.com/twright/bondwb`
[2] `https://github.com/twright/Logic-of-Behaviour-in-Uncertain-Contexts`
[3] `https://github.com/continuouspi/cpiwb`

```
species Whelk = dieWhelk -> 0
              + growWhelk -> (Whelk | Whelk)
              + beWhelk -> Whelk;

species Lobster = dieLobster -> 0
                + beLobster -> Lobster
                + growLobster -> (Lobster | Lobster);

kinetic law NLGrowth(g; x) = g*x*(1 - x);

kinetic law NLDecay(d; x) = d*x*(1 + x);

kinetic law RRPredation(g, h; x, y) = g*x*(h - x)*y;

affinity network N(k, b, c, e, f) = {
    growWhelk at rate NLGrowth(b);
    dieWhelk || beLobster at rate RRPredation(c, k);
    dieLobster at rate NLDecay(e);
    beWhelk || growLobster at rate RRPredation(f, k);
}

process Pi = [0.4] Whelk || [1.0] Lobster
             with network N(0.6, 0.3, 0.05, 2, 0.3);
```

Interval uncertain parameters may be used in place of any number in the model, so we may, for example, use the alternative uncertain initial process,

```
process Pi = [0.3, 0.5] Whelk || [1.0] Lobster
             with network N(0.6, 0.3, [0.05, 0.8], 2, 0.3);
```

We support ODE extraction by generating Python scripts representing system variables using the Sagemath [336] computer algebra system. This allows us to use Sagemath and the SymPy library [263] to simplify the symbolic systems generated by our ODE extraction algorithms, ensuring that these systems are produced in a compact, efficient, and readable form; this is especially important to ensure effective use of interval methods such as Flow* which are sensitive to the structure of symbolic expressions. Stochastic simulation is supported by extracting Chemical Reaction Networks associated with the system in

the form of models for the StochPy library[4] [250], which supports a range of stochastic simulation algorithms.

9.2 Python Interface and $\mathcal{LBUC}$

We also support a Python interface to the bond-calculus with extensive integration with the Sagemath [336] computer algebra system as the `lbuc.bondcalculus` module whilst $\mathcal{LBUC}$ is implemented as an embedded Domain Specific Language inside Python presented as the `lbuc.logic` module. These are designed to be used interactively in a Jupyter notebook [224] in conjunction using either a Sagemath or Python language kernel.

For example, one can load an existing bond-calculus model file by

```
m = BondModel("WhelksAndLobsters.bond")
```

create new bond-calculus processes by

```
p = m.process("[1.0 .. 1.2] Whelk || [4 .. 6] Lobster "
              "with network N(0.8, 0.6, 0.3, 0.05, 2)")
```

and extract a system of ODEs (in the form of a `System` object) by

```
s = m.as_system
```

Intervals are represented using Sagemath's interval libraries whilst polynomials and more general symbolic expressions are also represented using appropriate Sagemath expressions. Systems of ODEs may also be directly specified in Sagemath syntax. We provide extensive support for plotting system dynamics based on Sagemath's graphics libraries. For example, we may plot a streamline plot via a method call,

```
s.streamline_plot((s.v('Whelk'), 0, 1.5), (s.v('Lobster'), 0, 8))
```

whilst we can use our Flow* integration to compute a flowpipe for 5 time units via

```
flowpipe = s.reach(5)
```

and use a variety of methods to plot over-approximations of the flowpipe e.g. using intervals,

```
flowpipe.sage_interval_plot('Whelk', 'Lobster')
```

[4] `http://stochpy.sourceforge.net/`

All plots are generated as Sagemath `Graphics` objects making it easy to combine them to create composite plots such as Fig. 6.7.

$\mathcal{LBUC}$ properties may be directly specified as Python objects using Sagemath symbolic expressions. For example, a polynomial atomic proposition may be represented by an expression,

```
P = Atomic((var("Whelk") - 1)**2 + var("Lobster")**2 > 0.2)
```

whilst a contextual property $\varphi \triangleq [0,2]\,\textsc{Lobster} \triangleright \mathcal{F}_{[0,5]}\,\mathcal{G}_{[0,5]}\,P$ is represented by

```
phi = "[0, 2] Lobster" >> F([0, 5], G([0, 5], P))
```

We may then monitor the property `phi` over the system `s` for 5 time units via a call,

```
phi.signal_for_system(s, 5)
```

or a contextual signal via a call,

```
phi.context_signal_for_system(s, 5)
```

The full range of time-bounded monitoring algorithms described in this thesis are implemented as part of this library, along with an extensive collection of methods for visualizing signals and signal trees have been used to generate many of the diagrams in this thesis. Signal trees are implemented as lazy infinite data structures, so we may incrementally query a contextual signal at increasing levels of refinement until we achieve sufficient precision, without having to repeat our analysis from scratch.

A full tutorial for the tool is provided as a notebook[5] in the $\mathcal{LBUC}$ repository. Our use of Python and Jupyter notebooks increases accessibility of these tools for biological modelling, since provides a familiar interactive environment whilst making it easy to combine our verification techniques with further model analysis using Sagemath or standard Python scientific libraries including SciPy [218], Pandas [341], and Matplotlib [203], which are emerging as an increasingly popular platform for scientific programming. Other biological modelling formalisms are also explored related approaches including PySB [248] which embeds rule-based models within Python and BIOCHAM-4 [159] which provides an interactive analysis environment for BIOCHAM models via a custom Jupyter kernel. Also related is CORA [7] which implements a range of hybrid systems state representations and reachability methods as a MATLAB toolbox to aid the rapid prototyping of new reachability methods.

[5] `https://github.com/twright/Logic-of-Behaviour-in-Uncertain-Contexts/blob/master/Introduction.ipynb`

9.3 Flow* Wrapper and Verified Monitoring

By using Python for the majority of the *LBUC* codebase we gain the productivity advantages of a relatively high-level language, whilst allowing interactive use via a Jupyter notebook environment. However, Python is an inappropriate language for performant implementations of low-level numerical code such as interval arithmetic, Taylor model manipulations, and root-finding methods whilst for these operations we would like to use the C++ libraries including as part of Flow*. Therefore, whilst the majority of the high-level signal-based algorithms of *LBUC* are pure Python core, for monitoring atomic propositions we defers to a CPython C Extension: the `flowstar` module. This is written in the Cython language [34] which provides a restricted variant of Python with annotations which can be directly compiled to efficient C or C++ code.

In particular, we have developed a Python wrapper of Flow* which makes it possible to perform verified integration on continuous systems expressed in Python with standard Sage symbolic expressions and interact with the resulting flowpipe. This directly interacts with Flow*'s C++ API to avoid the need to generate Flow* model files and for flowpipes to be written to disk and parsed. Our algorithms for monitoring atomic propositions are also implemented at the Cython level, using Flow*'s C++ libraries to perform Taylor model and interval operations. To a certain extent, this judicious separation between languages gives us the best of both worlds since the majority of our execution time typically takes place within the optimized Cython code, meaning the overall performance of our algorithms should be comparable with methods implemented in C++. Our implementation uses Flow* 2.1.0 with patches to use native floating point arithmetic in the place of MPFR and to remove memory leaks.

9.4 Models, Benchmarks, and Testing

All of the models described in this thesis have been implemented in using the bond-calculus and verified using our implementation of *LBUC*. *LBUC*'s Python embedding has been used to explore these models interactively to test and to use standard Sage and Python plotting libraries to visualise system dynamics, signals, and contextual signals. For the verification of unbounded-time properties in Section 6.5, we verified semi-algebraic invariants using a Python implementation of the Liu, Zhan, and Zhao decision procedure [242] and the QEPCAD B quantifier elimination tool [78] (via its Sagemath interface); a fully automated integration of invariant synthesis methods and our verified monitoring algorithm remains

future work.

All benchmarks and performance measurements in this thesis have been carried out using $\mathcal{LBUC}$'s implementation and Python embedding and run on an AMD Ryzen 7 3700U with 32 GB of RAM under Fedora Linux. Our benchmarks were carried in Python code which directly invokes and times relevant test systems and properties. Thus, we were able to create automatic and reproducible benchmarks without resorting to a separate scripting language. We also extended our verification algorithms with granular instrumentation based on the Python `instrument`[6] library to gather detailed timing information for the individual operations and model-checking stages, giving us more a clearer picture of our algorithm's performance characteristics. Data analysis and visualization were carried out using the Python Pandas [341] and Altair libraries [351].

An extensive suite of automated tests is also used for both the bond-calculus and $\mathcal{LBUC}$ to provide some assurance of the correctness of our implementation and to avoid regressions. These involve many sample models and properties beyond those featured in this thesis, including comparing the results of verified monitoring procedure and contextual monitoring procedure to directly computed signals for model systems possessing explicit solutions. The bond-calculus makes extensive use of the property-based random testing using QuickCheck [128] to ensures that our implementation conforms to properties expected from our formal semantics (including, in particular, termination of our normalization procedures).

[6] `https://github.com/wearpants/instrument`

Chapter 10

Conclusion

In this thesis we have introduced a new modelling and analysis framework for biological systems under uncertainty. We firstly introduced affinity patterns and a new continuous semantics for the bond-calculus which is compositional in both process concentrations and affinities. This was demonstrated by exploring different modelling styles for gene regulatory networks at the molecular and network levels. We then extended the bond-calculus semantics to handle uncertainty in both the concentrations of species and the affinity rate law parameters of a model, and defined the $\mathcal{LBUC}$ logic for expressing spatio-temporal properties of bond-calculus models in uncertain contexts. We defined verified monitoring algorithms for STL and $\mathcal{LBUC}$ properties under uncertainty using Flow* flowpipes. These algorithms feature tight integration with Flow*'s symbolic flowpipe representation to precisely monitor complex atomic propositions and masks which avoid unnecessary symbolic operations by restricting monitoring of atomic propositions to time regions relevant to the overall property. Finally, we extended this to spatio-temporal monitoring for $\mathcal{LBUC}$ over the *context space* induced by model or contextual uncertainty, and introduced refinement techniques to quantify results over the context space and adaptive symbolic/physical flowpipe refinement techniques to improve results whilst minimizing the number of verified integrations required.

In this chapter we give our conclusions. In Section 10.1 we discuss our results and reflect upon the different components of this thesis and discuss related approaches whilst in Section 10.2 we discuss the potential for future work extending our approach.

10.1 Evaluation and Related Work

We will now reflect on the effectiveness of the bond-calculus, $\mathcal{LBUC}$, and our verification methods, with reference to relevant related work.

10.1.1 The Bond-Calculus and $\mathcal{LBUC}$

One part of this thesis has involved expanding and evaluating the effectiveness of the bond-calculus as a framework for modelling complex patterns of interaction in biological systems at the molecular and network levels. In Chapter 3 we introduced a new continuous semantics for the bond-calculus which directly extends the compositional approach of Continuous π's semantics to multiway interactions and general kinetic laws. This resolves a long-standing limitation in this compositional approach to process algebra semantics, identified in [232, Chapter 7] and in [358], that the interaction tensor $\oplus_M$ used in the Continuous π semantics relies upon multilinearity to achieve compositionality and hence was fundamentally tied to mass action kinetics. Our compositional semantics for complex reaction types also significantly expands the applicability of associated logics, meaning that $\mathcal{LBUC}$ is readily applicable to high-level models of biological networks with general kinetic laws as well as the lower-level mass action molecular models which Continuous π and hence $\mathcal{LBC}$ focused on. Moreover, the fact that bond-calculus models are compositional in both mixture and affinity networks directly increases the expressiveness of $\mathcal{LBUC}$ to include affinity network contexts (Section 5.3.5) and differential contexts (Section 5.3.4).

Our combination of site-based multiway communication and mobility is another interesting aspect of the bond-calculus. The main other process algebra which has explored this combination in a biological setting is the link-calculus [54] which uses a rather different multiway communication mechanism based on the formation of link-chains, however, so far this calculus has focused on a qualitative semantics and applications to membrane interactions [55]. The bond-calculus and the link-calculus share the ability to model *open multiway interactions* which can be dynamically joined by a variable number of agents whilst the bond-calculus also models multilevel multiway interactions which may be joined both by agents in different chemical species and multiple components of a single species in an allosteric interaction. Our particular form of site-based communication has interesting consequences for our continuous semantics since it is possible for a single chemical species to experience arbitrary arity multiway interactions *with itself*, as demonstrated in Section 4.1.2. This is handled via the interaction exponential Eq. (3.1), which generalizes the separate rules for unary reactions and homogeneous and heterogeneous binary reactions

in previous continuous process algebra semantics [97, 234].

Handling uncertain initial conditions and rate parameters is the most significant extension introduced in this thesis. This has some overlap in its applications to stochastic calculi such as Stochastic π [303] and Bio-PEPA [126] which are able to simulate systems with reactions occurring at stochastic rates, whilst the ProPPA process algebra [179] provides parameter inference for systems whose (time-invariant) reaction rate parameters are drawn from distributions. The non-deterministic model of uncertainty we pursue is, however, rather different since it corresponds to worst-case verification of uncertain continuous systems rather than the average case behaviour of stochastic systems. Various works have demonstrated the independent importance of both types of uncertainty by considering models which combine stochastic and nondeterministic uncertainty [62, 64, 70, 356].

The $\mathcal{LBUC}$ extended our handling of uncertainty by making it possible to express temporal properties of bond-calculus models in uncertain contexts. At the most basic level, the purely temporal subset of this language makes it perform verified three-valued monitoring for STL over bond-calculus models with uncertain initial conditions or rate parameters, to ensure that monitoring results are robust under the range of uncertainty expressed in the model. The addition of uncertain contexts allows us to specify the reactive behaviour of a model to contextual uncertainty at the mixture or affinity network levels. This allows us to reason about robustness under timed perturbations to a model's environment and enables us to reason about open systems whose environment grows and varies over time.

$\mathcal{LBUC}$ also allows us to express spatio-temporal experiments which examine the behaviour of a system through properties expressing experimental protocols, probing the logical dependency of a system's behaviour on sequences of external perturbations. Uncertain context operators are able to encode a range of key experimental operations such as instantaneously or gradually adding new reactants or triggering new reaction rules, however, other important experimental operations such as splitting and mixing reaction vessels cannot currently be expressed within $\mathcal{LBUC}$. Our approach is worth comparing to [1] which introduces a dedicated calculus for describing biological experimental protocols with deterministic and stochastic semantics. Whilst their approach more explicitly models experimental operations, as a temporal logic $\mathcal{LBUC}$ is able to express precise timing requirements and complex logical dependencies between perturbations and model behaviour.

Given our focus on optimizing our core monitoring algorithms, we mostly relied upon

small but computationally challenging examples. Whilst these examples were useful for validating our basic methods and investigating monitoring performance, these models have not yet fully explored the expressiveness and verification abilities of $\mathcal{LBUC}$ for more realistic biological models and properties. This motivates further investigation of $\mathcal{LBUC}$ in more practical biological modelling case studies. Additionally, our interest in performance led us to focus on properties at the borderline of satisfaction — the worst case for verified monitoring performance — whilst arguably properties which are more robustly satisfied are more typical of practical modelling and specification; thanks to masking and the other adaptive techniques in this thesis, much of the expense of verification can be avoided in these cases. Some additional validation of our methods' applicability to biological models has been given outside of this thesis in the BSc dissertation of Fiorista [163] who evaluated the effectiveness of bond-calculus modelling and $\mathcal{LBUC}$ verification for a range of signalling pathway motifs from [350]. $\mathcal{LBUC}$ should also be applicable to extending the wide range of biological systems and properties to which $\mathcal{LBC}$ has previously been applied [20–22] to take into account parameter uncertainty.

10.1.2 Verified Monitoring over Flowpipes

Throughout this thesis we focused on exact formal methods based on the Flow* verified integrator. This proved a powerful technique, allowing us to soundly account for all sources of uncertainty within the underlying model and the verification process whilst handling whole sets of input states at once. The exact nature of our methods gives us strong reassurances that the verification results of our logic hold robustly under contextual uncertainty and numerical errors. This is especially important in the context of $\mathcal{LBUC}$ verification since the worst case behaviour of the system on a small region of the context space can completely change the truth of the overall property whilst incorrect monitoring results can be compounded during the monitoring of complex properties.

Exact formal verification of nonlinear continuous and hybrid systems remains an active and challenging research area, and it is not yet clear which combination of methods will ultimately dominate in enabling practical verification at scale, as the current leading methods are usually limited to systems with at most 10-20 variables [205]. Whilst very competitive in its ability to handle complex dynamics and large uncertain sets, Flow* poses additional challenges to verified monitoring since expensive symbolic operations are necessary to extract information from its preconditioned Taylor model flowpipes, which motivated our focus on core STL monitoring performance. The techniques of adaptive

monitoring and masks made significant progress towards this challenge by restricting these symbolic operations to the regions of time most relevant to the property at hand. This means the cost of applying verified monitoring for a system is usually reasonable in comparison to the base verified integration cost, meaning verified monitoring should be applicable in most settings in which Flow* is currently used. This cost is also often *less* than the cost of other analysis techniques implemented in Flow* such as plotting and reach-avoidance checking since these perform symbolic flowpipe operations upfront over the whole time domain, whilst our methods work in a property-directed manner, avoiding work which is not relevant to the property at hand.

Handling context operators brings with it additional challenges, as already encountered by numerical model-checking procedures for $\mathcal{LBC}$ [19], since the combination of contextual and temporal operators creates *higher-order properties* which quantify over multiple potential executions of the system. Such properties are key to the expressiveness of the logic but can lead to an exponential increase in monitoring time in the case of alternation of temporal and contextual modalities. In $\mathcal{LBUC}$ we further add to these challenges of uncertainty in contexts and initial conditions. In these cases, however, our use of formal methods and Flow* pay us back for their baseline complexity, by allowing us to soundly handle whole sets of states in a single run. This allows us to monitor context operators using a relatively small number of Flow* verified integration runs to adaptively cover a given time domain and to handle uncertain contexts and initial conditions.

Contextual signal monitoring and refinement extends the range of applicability of these techniques to wider ranges of uncertainty (in both initial conditions and uncertain contexts) by allowing us to combine trees of signals monitored on different regions of the context space. Supported by our use of Flow* flowpipes to symbolically encode functional dependencies between our position in context space to the eventual state of the system, contextual monitoring is able to subdivide the context space without recomputing the flowpipe on every subregion. Thus it is often possible to evaluate an entire signal tree from a single run of Flow*. This technique only goes so far given the exponential increase we observed in monitoring cost with expansion depth. This problem is especially challenging when the context space is too large for Flow* to handle symbolically or at separatrices of the system which lead to divergent trajectories. We did, however, frequently observe a significant increase in monitoring precision from the first few levels of expansion, and the adaptive nature of this method allows us to avoid additional expansion once a conclusive verification result has been achieved. Our ability to handle large context spaces could be improved by smarter subdivision schemes which subdivide the context space into regions

with similar dynamics [324] or by better heuristics to focus subdivision and physical flowpipe computations on relevant regions of the context space.

The use of masking is also especially useful when monitoring context operators as demonstrated in Section 7.6.2. In this case it saves not just the cost of symbolic flowpipe evaluation, but whole Flow* verified integration runs for time regions not relevant to the property at hand. This application of masking is not specific to our use of verified monitoring and could also be applied to existing signal-based numerical monitoring algorithms for $\mathcal{LBC}$. Whilst we introduced contextual masking in Section 8.3 in order to bring the previously established advantages of masked monitoring to contextual and refined signal monitoring, additional benchmarking would be helpful to fully explore its performance implications and better evaluate its ability to prune regions of the combined context space-time domain, as well as the impact of downtree masking.

The most closely related techniques to masks are online monitoring algorithms, which perform monitoring on partial traces of running systems. Both [146] and the earlier incremental marking procedure [254, 279] attempt to avoid unnecessary monitoring of atomic propositions. Masks take a quite different approach, computing non-contiguous regions of interest for property in a top-down manner throughout the monitoring process, whereas [146] dynamically tracks contiguous time horizons of propositions. Masks also play a quite different role in our verification process, given that they are used to prevent unnecessary symbolic flowpipe operations or reachability computations arising from context operators, whereas online monitoring seeks to reduce the cost of monitoring over long time horizons or to allow early termination. The use of masks can also be compared with the numerical trace based monitoring methods of Banks and Stark [19] which offer support for early termination and short-circuiting, however, [20] found signal-based monitoring to be significantly faster overall [20, Chapter 6 and Appendix B]. Trace-based methods are also not applicable to verified monitoring over Flow* flowpipes given their continuous time domains.

Despite the progress made in this thesis, our exact formal methods based approach is still limited in its applicability to more practical biological models by both the high cost verified integration for nonlinear systems and the complexity of $\mathcal{LBUC}$ monitoring. To some extent, we hope that this will be improved by future advancements in continuous reachability methods, mirroring the large advances made for linear systems reachability throughout the 2000s [168]. The applicability of our methods could also be extended via compositional reasoning on decomposed models as we proposed in Section 10.2.2. It would also be worth further comparison of our methods to more pragmatic trajectory-

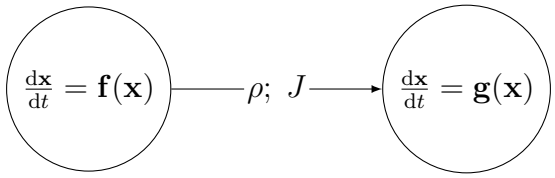
based approaches including sensitivity analysis [23], discrepancy functions [155], statistical methods [63, 249], or quantitative measures of satisfaction [146, 150].

10.2 Future Work

We will finally discuss some potential extensions to the methods of the work in this thesis.

10.2.1 Hybrid Semantics

The focus throughout this thesis has been on continuous systems, supported via the continuous ODE semantics of the bond-calculus, where all discrete interactions are modelled as external influences via $\mathcal{LBUC}$ context operators. This allows us to model many interesting forms of hybrid behaviour since, given a single hybrid automaton jump of form



with guard condition ρ and some jump map J, we can use a $\mathcal{LBUC}$ property

$$\varphi\, \mathcal{U}_{[0,\infty)}\ (\rho \wedge \mathcal{C} \triangleright \mathcal{G}_{[0,\infty)}\, \psi)$$

to ensure that this system obeys φ before the jump and ψ thereafter (with an appropriate context $\mathcal{C}$ encoding the jump from $\mathbf{f}$ to $\mathbf{g}$). $\mathcal{LBUC}$ also allows us to express complex logical dependencies of behaviour on jumps outside of the scope of STL properties over hybrid systems. This approach is, however, limited in its ability to express arbitrary behaviour in the discrete component of the systems since a given $\mathcal{LBUC}$ property can only express finite sequences of mode switches. Additionally, from a modelling perspective it may often be more natural to directly employ discreteness when modelling components of a system. This would make it possible to independently model intrinsic hybrid behaviour within a system and discrete events arising from interactions with its external environment.

This motivates the extension of the bond-calculus to hybrid systems. This could take the form both of discrete "control species" which occur as finite populations and of discrete mutations of the affinity network which change the reactivity of sites. Such an extended hybrid bond-calculus could then be given a hybrid semantics drawing on our existing semantics or alternative hybrid process algebra semantics such as [68, 172]. $\mathcal{LBUC}$ could also be coupled with other existing process algebras such as HyPE [174],

since the general structure of the logic is applicable to any calculus with a compositional structure. The basic verification algorithms developed throughout this thesis should be applicable to hybrid automata since these are well supported by Flow*. This may, however, require new techniques to effectively handle hybrid automaton jumps since the over-approximated flowpipes Flow* generates may not always provide enough information for effective verification and additional care will be required to handle the application of context operators at different modes of the system.

An alternative approach would be to extend $\mathcal{LBUC}$ with a least fixed point operator μ similarly to the modal μ-calculus, making it possible to express unbounded sequences of contextual jumps triggered within certain time windows or by certain preconditions. From a theoretical perspective, it would be interesting to investigate the extent to which this would close the gap in expressiveness between context operators and hybrid automata, and what relationship the resulting logic would have to dynamic logics such as d$\mathcal{L}$ [295] and STd$\mathcal{L}$ [4]. This logic would likely defy a general purpose model-checking procedure in the style of $\mathcal{LBUC}$, however, proof-rules based on a combination of verification methods similar to those proposed in Section 6.3 may be applicable in special cases.

10.2.2 Bounded Guarantee Operators

The uncertain context operator in $\mathcal{LBUC}$ can be seen as an intermediate form between the context operator $\Pi \triangleright \varphi$ of $\mathcal{LBC}$, which introduces a specific process Π as a context, and the full guarantee $\psi \triangleright \varphi$ operator of ambient logic [103] which uses a logical property ψ to specify a class of context properties. Whilst verification of general guarantee operators remains very difficult, given the unboundedness of the potential contexts over which they quantify, our verification methods could be extended to a more general class of *bounded guarantee operators*,

$$(\mathcal{C} : \psi) \triangleright \varphi$$

which assert that the property φ holds for any context model $\mathcal{M} \in \mathcal{C}$ satisfying the precondition ψ.

Such bounded guarantees include properties of the form $(\mathcal{C} : \rho) \triangleright \varphi$ where $\rho \triangleq f(\mathbf{x}) > 0$ is an atomic proposition. Such a property introduces a non-rectangular context set described via the state formula $(\mathbf{x} \in \mathcal{C}) \wedge (f(\mathbf{x}) > 0)$. The verification methods of Chapter 6 could be directly extended to this class of properties by over-approximating this set initial condition via a Taylor model. The more general class of bounded guarantee operators $(\mathcal{C} : \psi) \triangleright \varphi$ with arbitrary $\mathcal{LBUC}$ properties as preconditions could also be

tackled by applying the contextual monitoring and contextual masking techniques from Chapter 8 to identity the region of $\mathcal{C}$ for which ψ holds and then monitor φ under an appropriate spatial mask. This could allow a form of *assume-guarantee reasoning* where verification results for lower-dimensional subsystems may be used restrict the uncertainty of higher-dimensional composed systems. This style of compositional reasoning is also worth comparing to [29] which proposed a compositional approach to synthetic biological circuit design based on STL parameter synthesis. Related methods which aid hybrid systems reachability analysis via decomposition have also been proposed in [111] and [240].

Bibliography

[1] Alessandro Abate et al. “Experimental Biological Protocols with Formal Semantics”. In: *Computational Methods in Systems Biology*. Ed. by Milan Češka and David Šafránek. Cham: Springer International Publishing, 2018, pp. 165–182.

[2] Gary K Ackers, Alexander D Johnson, and Madeline A Shea. “Quantitative model for gene regulation by lambda phage repressor”. In: *Proceedings of the National Academy of Sciences* 79.4 (1982), pp. 1129–1133.

[3] Arend Aerts et al. “Temporal logic falsification of cyber-physical systems: An input-signal-space optimization approach”. In: *2018 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*. IEEE. 2018, pp. 214–223.

[4] Hammad Ahmad and Jean-Baptiste Jeannin. “A program logic to verify signal temporal logic specifications of hybrid systems”. In: *Proceedings of the 24th International Conference on Hybrid Systems: Computation and Control*. 2021, pp. 1–11.

[5] Ozgur Akman et al. “Modelling biological clocks with Bio-PEPA: stochasticity and robustness for the Neurospora Crassa circadian network”. In: *Computational Methods in Systems Biology*. Springer. 2009, pp. 52–67.

[6] Ozgur E. Akman et al. “Complementary approaches to understanding the plant circadian clock”. In: Proceedings Third Workshop *From Biology To Concurrency and back,* Paphos, Cyprus, 27th March 2010. Ed. by Emanuela Merelli and Paola Quaglia. Vol. 19. Electronic Proceedings in Theoretical Computer Science. Open Publishing Association, 2010, pp. 1–19.

[7] Matthias Althoff. “Cora 2015 manual”. In: *TU Munich* 85748 (2016).

[8] Rajeev Alur, Tomás Feder, and Thomas A Henzinger. “The benefits of relaxing punctuality”. In: *Journal of the ACM (JACM)* 43.1 (1996), pp. 116–146.

[9] Rajeev Alur et al. “The algorithmic analysis of hybrid systems”. In: *Theoretical computer science* 138.1 (1995), pp. 3–34.

[10] Bogdan Aman and Gabriel Ciobanu. “Bonding calculus”. In: *Natural Computing* 17.4 (2018), pp. 823–832.

[11] David Angeli. “A tutorial on chemical reaction network dynamics”. In: *European journal of control* 15.3-4 (2009), pp. 398–406.

[12] Yashwanth Annpureddy et al. “S-Taliro: A tool for temporal logic falsification for hybrid systems”. In: *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*. Springer. 2011, pp. 254–257.

[13] Adam Arkin, John Ross, and Harley H McAdams. “Stochastic kinetic analysis of developmental pathway bifurcation in phage λ-infected Escherichia coli cells”. In: *Genetics* 149.4 (1998), pp. 1633–1648.

[14] Eugene Asarin et al. “Approximate Reachability Analysis of Piecewise-Linear Dynamical Systems”. In: *Hybrid Systems: Computation and Control*. Ed. by Nancy Lynch and Bruce H. Krogh. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, pp. 20–31.

[15] Giorgio Bacci, Davide Grohmann, and Marino Miculan. “A framework for protein and membrane interactions”. In: *Proceedings Third Workshop on Membrane Computing and Biologically Inspired Process Calculi, MeCBIC 2009, Bologna, Italy, 5th September 2009.* 2009, pp. 19–33. URL: https://doi.org/10.4204/EPTCS.11.2.

[16] Giorgio Bacci, Davide Grohmann, and Marino Miculan. “Bigraphical models for protein and membrane interactions”. In: *Proceedings Third Workshop on Membrane Computing and Biologically Inspired Process Calculi, MeCBIC 2009, Bologna, Italy, 5th September 2009.* 2009, pp. 3–18. URL: https://doi.org/10.4204/EPTCS.11.1.

[17] Giorgio Bacci and Marino Miculan. “Undecidability of Model checking in Brane Logic”. In: *Electronic Notes in Theoretical Computer Science* 192.3 (2008), pp. 23–37.

[18] Kyungmin Bae and Jia Lee. “Bounded model checking of signal temporal logic properties using syntactic separation”. In: *Proceedings of the ACM on Programming Languages* 3.POPL (2019), p. 51.

[19] C.J. Banks and I. Stark. “A logic of behaviour in context”. In: *Information and Computation* 236 (2014). Special Issue on Hybrid Systems and Biology, pp. 3–18. URL: http://www.sciencedirect.com/science/article/pii/S0890540114000108.

[20] Chris Banks. “Spatio-temporal Logic for the Analysis of Biochemical Models”. Supervised by Ian Stark. PhD thesis. Edinburgh, 2016.

[21] Chris Banks et al. “Stochastic Modelling of the Kai-based Circadian Clock”. In: *Electronic Notes in Theoretical Computer Science* 296 (2013), pp. 43–60.

[22] Christopher J Banks, Daniel D Seaton, and Ian Stark. “Analysis of a post-translational oscillator using process algebra and spatio-temporal logic”. In: *International Conference on Computational Methods in Systems Biology*. Springer. 2015, pp. 222–238.

[23] Christopher J. Banks and Ian Stark. “A More Sensitive Context”. In: arXiv:1702.03288, arXiv:1702.03288 (Feb. 2017).

[24] Roberto Barbuti et al. “A calculus of looping sequences for modelling microbiological systems”. In: *Fundamenta Informaticae* 72.1-3 (2006), pp. 21–35.

[25] Roberto Barbuti et al. “A survey of gene regulatory networks modelling methods: from differential equations, to Boolean and qualitative bioinspired models”. In: *Journal of Membrane Computing* (2020), pp. 1–20.

[26] Roberto Barbuti et al. “Spatial calculus of looping sequences”. In: *Theoretical Computer Science* 412.43 (2011), pp. 5976–6001.

[27] Roberto Barbuti et al. “The Calculus of Looping Sequences”. In: *Formal Methods for Computational Systems Biology*. Ed. by Marco Bernardo, Pierpaolo Degano, and Gianluigi Zavattaro. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 387–423.

[28] A. Barkai and C. McQuaid. “Predator-Prey Role Reversal in a Marine Benthic Ecosystem”. In: *Science* 242.4875 (Oct. 1988), pp. 62–64. URL: https://doi.org/10.1126/science.242.4875.62.

[29] Ezio Bartocci, Luca Bortolussi, and Laura Nenzi. “A temporal logic approach to modular design of synthetic biological circuits”. In: *International Conference on Computational Methods in Systems Biology*. Springer. 2013, pp. 164–177.

[30] Ezio Bartocci and Pietro Lió. “Computational modeling, formal analysis, and tools for systems biology”. In: *PLoS computational biology* 12.1 (2016), e1004591.

[31] Ezio Bartocci et al. “A formal methods approach to pattern recognition and synthesis in reaction diffusion networks”. In: *IEEE Transactions on Control of Network Systems* 5.1 (2016), pp. 308–320.

[32] Ezio Bartocci et al. “Specification-based monitoring of cyber-physical systems: a survey on theory, tools and applications”. In: *Lectures on Runtime Verification*. Springer, 2018, pp. 135–175.

[33] Gregory Batt et al. “Robustness analysis and tuning of synthetic gene networks”. In: *Bioinformatics* 23.18 (2007), pp. 2415–2422.

[34] S. Behnel et al. “Cython: The Best of Both Worlds”. In: *Computing in Science Engineering* 13.2 (Mar. 2011), pp. 31–39.

[35] Andreea Beica, Calin C Guet, and Tatjana Petrov. “Efficient reduction of kappa models by static inspection of the rule-set”. In: *International Workshop on Hybrid Systems Biology*. Springer. 2015, pp. 173–191.

[36] Nuel D. Belnap. “A Useful Four-Valued Logic”. In: *Modern Uses of Multiple-Valued Logic*. Ed. by J. Michael Dunn and George Epstein. Dordrecht: Springer Netherlands, 1977, pp. 5–37. URL: https://doi.org/10.1007/978-94-010-1161-7_2.

[37] Nikola Beneš et al. “A Model Checking Approach to Discrete Bifurcation Analysis”. In: *FM 2016: Formal Methods*. Ed. by John Fitzgerald et al. Cham: Springer International Publishing, 2016, pp. 85–101.

[38] Nikola Beneš et al. “Pithya: A Parallel Tool for Parameter Synthesis of Piecewise Multi-affine Dynamical Systems”. In: *Computer Aided Verification*. Ed. by Rupak Majumdar and Viktor Kunčak. Cham: Springer International Publishing, 2017, pp. 591–598.

[39] Johan Bengtsson et al. “UPPAAL—a tool suite for automatic verification of real-time systems”. In: *International hybrid systems workshop*. Springer. 1995, pp. 232–243.

[40] Soufiene Benkirane. “Process algebra for epidemiology: evaluating and enhancing the ability of PEPA to describe biological systems”. PhD thesis. 2011.

[41] Soufiene Benkirane et al. “Improved continuous approximation of PEPA models through epidemiological examples”. In: *Electronic Notes in Theoretical Computer Science* 229.1 (2009), pp. 59–74.

[42] Soufiene Benkirane et al. “Measles epidemics and PEPA: an exploration of historic disease dynamics using process algebra”. In: *International symposium on formal methods*. Springer. 2012, pp. 101–115.

[43] Johan van Benthem and Guram Bezhanishvili. “Modal logics of space”. In: *Handbook of spatial logics*. Springer, 2007, pp. 217–298.

[44] Jan A Bergstra and Jan Willem Klop. “Algebra of communicating processes”. In: *CWI Monograph series* 3 (1986), pp. 89–138.

[45] Martin Berz and Georg Hoffstätter. “Computation and application of Taylor polynomials with interval remainder bounds”. In: *Reliable Computing* 4.1 (1998), pp. 83–97.

[46] Martin Berz and Kyoko Makino. *COSY INFINITY version 8 reference manual*. Tech. rep. Technical Report MSUCL, 1997.

[47] Martin Berz and Kyoko Makino. “Verified integration of ODEs and flows using differential algebraic methods on high-order Taylor models”. In: *Reliable computing* 4.4 (1998), pp. 361–369.

[48] Martin Berz and Kyoko Makino. “Verified integration of ODEs and flows using differential algebraic methods on high-order Taylor models”. In: *Reliable Computing* 4.4 (1998), pp. 361–369.

[49] Lacramioara Bintu et al. “Transcriptional regulation by the numbers: applications”. In: *Current Opinion in Genetics & Development* 15.2 (2005). Chromosomes and expression mechanisms, pp. 125–135.

[50] Lacramioara Bintu et al. “Transcriptional regulation by the numbers: models”. In: *Current Opinion in Genetics & Development* 15.2 (2005). Chromosomes and expression mechanisms, pp. 116–124.

[51] Michael L Blinov et al. “BioNetGen: software for rule-based modeling of signal transduction based on the interactions of molecular domains”. In: *Bioinformatics* 20.17 (2004), pp. 3289–3291.

[52] Ralf Blossey, Luca Cardelli, and Andrew Phillips. “A compositional approach to the stochastic dynamics of gene networks”. In: *Transactions on Computational Systems Biology IV*. Springer, 2006, pp. 99–122.

[53] Ralf Blossey, Luca Cardelli, and Andrew Phillips. "Compositionality, stochasticity, and cooperativity in dynamic models of gene regulation". In: *HFSP journal* 2.1 (2008), pp. 17–28.

[54] Chiara Bodei, Linda Brodo, and Roberto Bruni. "Open Multiparty Interaction". In: *Recent Trends in Algebraic Development Techniques: 21st International Workshop, WADT 2012, Salamanca, Spain, June 7-10, 2012, Revised Selected Papers*. Ed. by Narciso Martí-Oliet and Miguel Palomino. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 1–23. URL: https://doi.org/10.1007/978-3-642-37635-1_1.

[55] Chiara Bodei et al. "A flat process calculus for nested membrane interactions". In: *Scientific Annals of Computer Science* 24.1 (2014), p. 91.

[56] Max Bodenstein and SC Lind. "Geschwindigkeit der Bildung des Bromwasserstoffs aus seinen Elementen". In: *Zeitschrift für Physikalische Chemie* 57.1 (1907), pp. 168–192.

[57] Michael A Boemo, Luca Cardelli, and Conrad A Nieduszynski. "The Beacon Calculus: A formal method for the flexible and concise modelling of biological systems". In: *PLoS computational biology* 16.3 (2020), e1007651.

[58] Sergiy Bogomolov et al. "Abstraction-Based Parameter Synthesis for Multiaffine Systems". In: *Hardware and Software: Verification and Testing*. Ed. by Nir Piterman. Cham: Springer International Publishing, 2015, pp. 19–35.

[59] Sergiy Bogomolov et al. "Conic abstractions for hybrid systems". In: *International Conference on Formal Modeling and Analysis of Timed Systems*. Springer. 2017, pp. 116–132.

[60] Sergiy Bogomolov et al. "Counterexample-guided refinement of template polyhedra". In: *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*. Springer. 2017, pp. 589–606.

[61] Luca Bortolussi. "Stochastic Concurrent Constraint Programming". In: *Electronic Notes in Theoretical Computer Science* 164.3 (2006). Proceedings of the 4th International Workshop on Quantitative Aspects of Programming Languages (QAPL 2006), pp. 65–80. URL: http://www.sciencedirect.com/science/article/pii/S1571066106004919.

[62] Luca Bortolussi and Nicolas Gast. “Mean field approximation of uncertain stochastic models”. In: *Dependable Systems and Networks (DSN), 2016 46th Annual IEEE/IFIP International Conference on*. IEEE. 2016, pp. 287–298.

[63] Luca Bortolussi, Dimitrios Milios, and Guido Sanguinetti. “Smoothed model checking for uncertain Continuous-Time Markov Chains”. In: *Information and Computation* 247 (2016), pp. 235–253. URL: https://www.sciencedirect.com/science/article/pii/S0890540116000055.

[64] Luca Bortolussi, Dimitrios Milios, and Guido Sanguinetti. “U-Check: Model Checking and Parameter Synthesis Under Uncertainty”. In: *Quantitative Evaluation of Systems*. Ed. by Javier Campos and Boudewijn R. Haverkort. Cham: Springer International Publishing, 2015, pp. 89–104.

[65] Luca Bortolussi and Laura Nenzi. “Specifying and monitoring properties of stochastic spatio-temporal systems in signal temporal logic”. In: *Proceedings of the 8th International Conference on Performance Evaluation Methodologies and Tools*. 2014, pp. 66–73.

[66] Luca Bortolussi and Alberto Policriti. “Hybrid dynamics of stochastic programs”. In: *Theoretical Computer Science* 411.20 (2010), pp. 2052–2077.

[67] Luca Bortolussi and Alberto Policriti. “Hybrid dynamics of stochastic programs”. In: *Theoretical Computer Science* 411.20 (2010). Hybrid Automata and Oscillatory Behaviour in Biological Systems, pp. 2052–2077. URL: http://www.sciencedirect.com/science/article/pii/S0304397510001039.

[68] Luca Bortolussi and Alberto Policriti. “Hybrid Semantics for Stochastic π-Calculus”. In: *Algebraic Biology*. Ed. by Katsuhisa Horimoto et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 40–55.

[69] Luca Bortolussi and Alberto Policriti. “Stochastic Concurrent Constraint Programming and Differential Equations”. In: *Electronic Notes in Theoretical Computer Science* 190.3 (2007). Proceedings of the Fifth Workshop on Quantitative Aspects of Programming Languages (QAPL 2007), pp. 27–42. URL: http://www.sciencedirect.com/science/article/pii/S1571066107005567.

[70] Luca Bortolussi and Guido Sanguinetti. “A Statistical Approach for Computing Reachability of Non-linear and Stochastic Dynamical Systems”. In: *Quantitative Evaluation of Systems*. Ed. by Gethin Norman and William Sanders. Cham: Springer International Publishing, 2014, pp. 41–56.

[71] Luca Bortolussi et al. “CARMA: collective adaptive resource-sharing Markovian agents”. In: *arXiv preprint arXiv:1509.08560* (2015).

[72] Marius Bozga et al. “Kronos: A model-checking tool for real-time systems”. In: *Formal Techniques in Real-Time and Fault-Tolerant Systems*. Ed. by Anders P. Ravn and Hans Rischel. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998, pp. 298–302.

[73] Jeremy T Bradley, Stephen T Gilmore, and Jane Hillston. “Analysing distributed internet worm attacks using continuous state-space approximation of process algebra models”. In: *Journal of Computer and System Sciences* 74.6 (2008), pp. 1013–1032.

[74] Jeremy T. Bradley. “A Ticking Clock: Performance Analysis of a Circadian Rhythm with Stochastic Process Algebra”. In: *Computer Performance Engineering*. Ed. by Nigel Thomas and Carlos Juiz. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 79–94.

[75] Fred Brauer, Carlos Castillo-Chavez, and Carlos Castillo-Chavez. *Mathematical models in population biology and epidemiology*. Vol. 2. Springer, 2012.

[76] Davide Bresolin. “HyLTL: A Temporal Logic for Model Checking Hybrid Systems”. In: *Electronic Proceedings in Theoretical Computer Science* 124 (2013), pp. 73–84.

[77] Luboš Brim et al. “Parameter synthesis by parallel coloured CTL model checking”. In: *International Conference on Computational Methods in Systems Biology*. Springer. 2015, pp. 251–263.

[78] Christopher W Brown. “QEPCAD B: a program for computing with semi-algebraic sets using CADs”. In: *ACM SIGSAM Bulletin* 37.4 (2003), pp. 97–108.

[79] Roberto Bruni and Ivan Lanese. “Parametric synchronizations in mobile nominal calculi”. In: *Theoretical Computer Science* 402.2 (2008). Trustworthy Global Computing, pp. 102–119. URL: `http://www.sciencedirect.com/science/article/pii/S0304397508003484`.

[80] R Bundschuh, F Hayot, and C Jayaprakash. “Fluctuations and slow variables in genetic networks”. In: *Biophysical journal* 84.3 (2003), pp. 1606–1615.

[81] Ralf Bundschuh, F Hayot, and C Jayaprakash. “The role of dimerization in noise reduction of simple genetic networks”. In: *Journal of Theoretical Biology* 220.2 (2003), pp. 261–269.

[82] Fabrizio Banci Buonamici et al. “Spatial logics and model checking for medical imaging”. In: *International Journal on Software Tools for Technology Transfer* (2019), pp. 1–23.

[83] Jerry R Burch et al. “Symbolic model checking: 1020 states and beyond”. In: *Information and computation* 98.2 (1992), pp. 142–170.

[84] RM Burstall. “Program proving as hand simulation with a little induction. Information Processing 74”. In: *Proceedings of IFIP Congress*. Vol. 74. 1974, pp. 308–312.

[85] Nadia Busi, Maurizio Gabbrielli, and Gianluigi Zavattaro. “Replication vs. Recursive Definitions in Channel Based Calculi”. In: *Automata, Languages and Programming: 30th International Colloquium, ICALP 2003, Eindhoven, The Netherlands, June 30-July 4, 2003. Proceedings*. Vol. 2719. Springer. 2003, p. 133.

[86] Luis Caires and Etienne Lozes. “Elimination of quantifiers and undecidability in spatial logics for concurrency”. In: *International Conference on Concurrency Theory*. Springer. 2004, pp. 240–257.

[87] Luis Caires and Luis Monteiro. “Verifiable and executable logic specifications of concurrent objects in $\mathcal{L}_pi$”. In: *European Symposium on Programming*. Springer. 1998, pp. 42–56.

[88] Luis Caires and Luca Cardelli. “A spatial logic for concurrency (part I)”. In: *Information and Computation* 186.2 (2003), pp. 194–235.

[89] Luis Caires and Luca Cardelli. “A spatial logic for concurrency—II”. In: vol. 322. 3. Elsevier, 2004, pp. 517–565.

[90] Cristiano Calcagno, Luca Cardelli, and Andrew D Gordon. “Deciding validity in a spatial logic for trees”. In: *Proceedings of the 2003 ACM SIGPLAN international workshop on Types in languages design and implementation*. 2003, pp. 62–73.

[91] Muffy Calder, Stephen Gilmore, and Jane Hillston. “Modelling the Influence of RKIP on the ERK Signalling Pathway Using the Stochastic Process Algebra PEPA”. In: *Transactions on Computational Systems Biology VII*. Ed. by Corrado Priami et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 1–23. URL: https://doi.org/10.1007/11905455_1.

[92] Muffy Calder, Stephen Gilmore, and Jane Hillston. “Modelling the influence of RKIP on the ERK signalling pathway using the stochastic process algebra PEPA”. In: *Transactions on computational systems biology VII*. Springer, 2006, pp. 1–23.

[93] Muffy Calder et al. “Analysis of signalling pathways using the PRISM model checker”. In: (2005).

[94] Ferdinanda Camporesi, Jérôme Feret, and Kim Quyên Lý. “KaDE: A Tool to Compile Kappa Rules into (Reduced) ODE Models”. In: *Computational Methods in Systems Biology*. Ed. by Jérôme Feret and Heinz Koeppl. Cham: Springer International Publishing, 2017, pp. 291–299.

[95] Igor Cappello and Paola Quaglia. “A translation of beta-binders in a prioritized pi-calculus”. In: *Electronic Notes in Theoretical Computer Science* 229.1 (2009), pp. 109–125.

[96] Luca Cardelli. “Brane calculi”. In: *International Conference on Computational Methods in Systems Biology*. Springer. 2004, pp. 257–278.

[97] Luca Cardelli. “From Processes to ODEs by Chemistry”. In: *Fifth Ifip International Conference On Theoretical Computer Science – Tcs 2008*. Ed. by Giorgio Ausiello et al. Boston, MA: Springer US, 2008, pp. 261–281. URL: https://doi.org/10.1007/978-0-387-09680-3_18.

[98] Luca Cardelli. “On process rate semantics”. In: *Theoretical Computer Science* 391.3 (2008). Converging Sciences: Informatics and Biology, pp. 190–215. URL: http://www.sciencedirect.com/science/article/pii/S0304397507008559.

[99] Luca Cardelli. “Strand Algebras for DNA Computing”. In: *DNA Computing and Molecular Programming: 15th International Conference, DNA 15, Fayetteville, AR, USA, June 8-11, 2009, Revised Selected Papers*. Ed. by Russell Deaton and Akira Suyama. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 12–24. URL: https://doi.org/10.1007/978-3-642-10604-0_2.

[100] Luca Cardelli and Philippa Gardner. “Processes in space”. In: *Conference on Computability in Europe*. Springer. 2010, pp. 78–87.

[101] Luca Cardelli, Philippa Gardner, and Giorgio Ghelli. “A spatial logic for querying graphs”. In: *International Colloquium on Automata, Languages, and Programming*. Springer. 2002, pp. 597–610.

[102] Luca Cardelli and Giorgio Ghelli. “TQL: a query language for semistructured data based on the ambient logic”. In: *Mathematical structures in computer science* 14.3 (2004), pp. 285–327.

[103] Luca Cardelli and Andrew D Gordon. “Anytime, anywhere: Modal logics for mobile ambients”. In: *Proceedings of the 27th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*. ACM. 2000, pp. 365–377.

[104] Luca Cardelli and Andrew D. Gordon. “Mobile ambients”. In: *Theoretical Computer Science* 240.1 (2000), pp. 177–213. URL: http://www.sciencedirect.com/science/article/pii/S0304397599002315.

[105] Luca Cardelli and Radu Mardare. “Stochastic pi-calculus revisited”. In: *International Colloquium on Theoretical Aspects of Computing*. Springer. 2013, pp. 1–21.

[106] Luca Cardelli et al. “A Process Model of Actin Polymerisation”. In: *Electronic Notes in Theoretical Computer Science* 229.1 (2009). Proceedings of the Second Workshop From Biology to Concurrency and Back (FBTC 2008), pp. 127–144. URL: http://www.sciencedirect.com/science/article/pii/S1571066109000140.

[107] Nathalie Chabrier-Rivier, François Fages, and Sylvain Soliman. “The Biochemical Abstract Machine BIOCHAM”. In: *Computational Methods in Systems Biology*. Ed. by Vincent Danos and Vincent Schachter. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 172–191.

[108] Witold Charatonik et al. “The complexity of model checking mobile ambients”. In: *International Conference on Foundations of Software Science and Computation Structures*. Springer. 2001, pp. 152–167.

[109] Xin Chen. “Reachability Analysis of Non-Linear Hybrid Systems Using Taylor Models”. PhD thesis. Fachgruppe Informatik, RWTH Aachen University, 2015.

[110] Xin Chen, Erika Ábrahám, and Sriram Sankaranarayanan. “Flow*: An Analyzer for Non-linear Hybrid Systems”. In: *Computer Aided Verification*. Ed. by Natasha Sharygina and Helmut Veith. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 258–263.

[111] Xin Chen and Sriram Sankaranarayanan. “Decomposed reachability analysis for nonlinear systems”. In: *2016 IEEE Real-Time Systems Symposium (RTSS)*. IEEE. 2016, pp. 13–24.

[112] Alongkrit Chutinan and Bruce H Krogh. “Computational techniques for hybrid system verification”. In: *IEEE transactions on automatic control* 48.1 (2003), pp. 64–75.

[113] Vincenzo Ciancia et al. “Specifying and verifying properties of space”. In: *IFIP International Conference on Theoretical Computer Science*. Springer. 2014, pp. 222–235.

[114] Alessandro Cimatti et al. “NuSMV: A new symbolic model verifier”. In: *International conference on computer aided verification*. Springer. 1999, pp. 495–499.

[115] Alessandro Cimatti et al. “Verifying LTL properties of hybrid systems with k-liveness”. In: *International Conference on Computer Aided Verification*. Springer. 2014, pp. 424–440.

[116] Gabriel Ciobanu. “From gene regulation to stochastic fusion”. In: *International Conference on Unconventional Computation*. Springer. 2008, pp. 51–63.

[117] Gabriel Ciobanu. “From gene regulation to stochastic fusion”. In: *International Conference on Unconventional Computation*. Springer. 2008, pp. 51–63.

[118] Gabriel Ciobanu. “General patterns of interaction in stochastic fusion”. In: *Natural Computing* 12.3 (Sept. 2013), pp. 429–439. URL: https://doi.org/10.1007/s11047-012-9346-5.

[119] Gabriel Ciobanu and Angelo Troina. “Rate-Based Stochastic Fusion Calculus and Continuous Time Markov Chains”. In: (2012).

[120] Federica Ciocchetta. “Bio-PEPA with Events”. In: *Transactions on Computational Systems Biology XI* (2009), pp. 45–68.

[121] Federica Ciocchetta. “The BlenX Language with Biological Transactions”. In: *Transactions on Computational Systems Biology IX*. Ed. by Corrado Priami. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 114–152. URL: https://doi.org/10.1007/978-3-540-88765-2_4.

[122] Federica Ciocchetta. “Transactions on Computational Systems Biology XI”. In: ed. by Corrado Priami, Ralph-Johan Back, and Ion Petre. Berlin, Heidelberg: Springer-Verlag, 2009. Chap. Bio-PEPA with Events, pp. 45–68. URL: http://dx.doi.org/10.1007/978-3-642-04186-0_3.

[123] Federica Ciocchetta and Maria Luisa Guerriero. “Modelling Biological Compartments in Bio-PEPA”. In: *Electronic Notes in Theoretical Computer Science* 227 (2009). Proceedings of the Second International Meeting on Membrane Computing and Biologically Inspired Process Calculi (MeCBIC 2008), pp. 77–95. URL: http://www.sciencedirect.com/science/article/pii/S1571066108005689.

[124] Federica Ciocchetta and Jane Hillston. "Bio-PEPA for Epidemiological Models". In: *Electronic Notes in Theoretical Computer Science* 261 (2010). Proceedings of the Fourth International Workshop on the Practical Application of Stochastic Modelling (PASM 2009), pp. 43–69. URL: http://www.sciencedirect.com/science/article/pii/S157106611000006X.

[125] Federica Ciocchetta and Jane Hillston. "Bio-PEPA: A framework for the modelling and analysis of biological systems". In: *Theoretical Computer Science* 410.33 (2009), pp. 3065–3084. URL: http://www.sciencedirect.com/science/article/pii/S0304397509001662.

[126] Federica Ciocchetta and Jane Hillston. "Bio-PEPA: An Extension of the Process Algebra PEPA for Biochemical Networks". In: *Electronic Notes in Theoretical Computer Science* 194.3 (Aug. 2008), pp. 103–117. URL: http://dx.doi.org/10.1016/j.entcs.2007.12.008.

[127] Federica Ciocchetta and Corrado Priami. "Biological Transactions for Quantitative Models". In: *Electronic Notes in Theoretical Computer Science* 171.2 (2007). Proceedings of the First Workshop on Membrane Computing and Biologically Inspired Process Calculi (MeCBIC 2006), pp. 55–67. URL: http://www.sciencedirect.com/science/article/pii/S1571066107002770.

[128] Koen Claessen and John Hughes. "QuickCheck: a lightweight tool for random testing of Haskell programs". In: *Proceedings of the fifth ACM SIGPLAN international conference on Functional programming*. 2000, pp. 268–279.

[129] Edmund M Clarke and E Allen Emerson. "Design and synthesis of synchronization skeletons using branching time temporal logic". In: *Workshop on Logic of Programs*. Springer. 1981, pp. 52–71.

[130] Jonathan Coates. "A Formalisation of Biochemical Process Languages in Lean". School of Informatics, University of Edinburgh.

[131] Matthew Collinson, Brian Monahan, and David Pym. "A logical and computational theory of located resource". In: *Journal of Logic and Computation* 19.6 (2009), pp. 1207–1244.

[132] Patrick Cousot. "Abstract interpretation based formal methods and future challenges". In: *Informatics*. Springer. 2001, pp. 138–156.

[133] Troels C Damgaard, Vincent Danos, and Jean Krivine. "A language for the cell". In: *Technical Report TR-2008-116*. 2008.

[134] Vincent Danos and Cosimo Laneve. “Formal molecular biology”. In: *Theoretical Computer Science* 325.1 (2004), pp. 69–110.

[135] Joëlle De Caluwé et al. “A compact model for the complex plant circadian clock”. In: *Frontiers in plant science* 7 (2016).

[136] Joëlle De Caluwé et al. “Modeling the photoperiodic entrainment of the plant circadian clock”. In: *Journal of theoretical biology* 420 (2017), pp. 220–231.

[137] Leonardo De Moura and Nikolaj Bjørner. “Z3: An efficient SMT solver”. In: *International conference on Tools and Algorithms for the Construction and Analysis of Systems*. Springer. 2008, pp. 337–340.

[138] Rocco De Nicola, Gian Luigi Ferrari, and Rosario Pugliese. “KLAIM: A kernel language for agents interaction and mobility”. In: *IEEE Transactions on software engineering* 24.5 (1998), pp. 315–330.

[139] Rocco De Nicola and Michele Loreti. “MoMo: A modal logic for reasoning about mobility”. In: *International Symposium on Formal Methods for Components and Objects*. Springer. 2004, pp. 95–119.

[140] Rocco De Nicola et al. “Model checking mobile stochastic logic”. In: *Theoretical Computer Science* 382.1 (2007), pp. 42–70.

[141] Rocco De Nicola et al. “SoSL: a service-oriented stochastic logic”. In: *Rigorous Software Engineering for Service-Oriented Systems*. Springer, 2011, pp. 447–466.

[142] Tadeáš Děd et al. “Formal biochemical space with semantics in Kappa and BNGL”. In: *Electronic Notes in Theoretical Computer Science* 326 (2016), pp. 27–49.

[143] Pierpaolo Degano et al. “Beta-binders for biological quantitative experiments”. In: *Electronic Notes in Theoretical Computer Science* 164.3 (2006), pp. 101–117.

[144] Andrea Degasperi and Muffy Calder. “Process algebra with hooks for models of pattern formation”. In: *Electronic Notes in Theoretical Computer Science* 268 (2010), pp. 31–47.

[145] L. Dematté, C. Priami, and A. Romanel. “The BlenX Language: A Tutorial”. In: *Proceedings of the Formal Methods for the Design of Computer, Communication, and Software Systems 8th International Conference on Formal Methods for Computational Systems Biology*. SFM’08. Bertinoro, Italy: Springer-Verlag, 2008, pp. 313–365. URL: http://dl.acm.org/citation.cfm?id=1786698.1786708.

[146] Jyotirmoy V Deshmukh et al. “Robust online monitoring of signal temporal logic”. In: *Formal Methods in System Design* 51.1 (2017), pp. 5–30.

[147] David L Dill. “Timing assumptions and verification of finite-state concurrent systems”. In: *International Conference on Computer Aided Verification*. Springer. 1989, pp. 197–212.

[148] Alexandre Donzé. “Breach, a toolbox for verification and parameter synthesis of hybrid systems”. In: *International Conference on Computer Aided Verification*. Springer. 2010, pp. 167–170.

[149] Alexandre Donzé, Gilles Clermont, and Christopher J Langmead. “Parameter synthesis in nonlinear dynamical systems: Application to systems biology”. In: *Journal of Computational Biology* 17.3 (2010), pp. 325–336.

[150] Alexandre Donzé and Oded Maler. “Robust Satisfaction of Temporal Logic over Real-Valued Signals”. In: *Formal Modeling and Analysis of Timed Systems*. Ed. by Krishnendu Chatterjee and Thomas A. Henzinger. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 92–106.

[151] Alexandre Donzé and Oded Maler. “Systematic simulation using sensitivity analysis”. In: *International Workshop on Hybrid Systems: Computation and Control*. Springer. 2007, pp. 174–189.

[152] Tommaso Dreossi. “Sapo: Reachability computation and parameter synthesis of polynomial dynamical systems”. In: *Proceedings of the 20th International Conference on Hybrid Systems: Computation and Control*. 2017, pp. 29–34.

[153] Tommaso Dreossi, Thao Dang, and Carla Piazza. “Parallelotope bundles for polynomial reachability”. In: *Proceedings of the 19th International Conference on Hybrid Systems: Computation and Control*. 2016, pp. 297–306.

[154] Denys Duchier and Céline Kuttler. “Biomolecular Agents as Multi-behavioural Concurrent Objects”. In: *Electronic Notes in Theoretical Computer Science* 150.1 (2006). Proceedings of the First International Workshop on Methods and Tools for Coordinating Concurrent, Distributed and Mobile Systems (MTCoord 2005), pp. 31–49. URL: `http://www.sciencedirect.com/science/article/pii/S1571066106000983`.

[155] Parasara Sridhar Duggirala et al. “C2E2: a tool for verifying annotated hybrid systems”. In: *HSCC*. 2015.

[156] Johan S Eklöf et al. “A spatial regime shift from predator to prey dominance in a large coastal ecosystem”. In: *Communications Biology* 3.1 (2020), pp. 1–9.

[157] François Fages and Aurélien Rizk. “On temporal logic constraint solving for analyzing numerical data time series”. In: *Theoretical Computer Science* 408.1 (2008), pp. 55–65.

[158] François Fages and Aurélien Rizk. “On the analysis of numerical data time series in temporal logic”. In: *International Conference on Computational Methods in Systems Biology*. Springer. 2007, pp. 48–63.

[159] François Fages and Sylvain Soliman. “On robustness computation and optimization in BIOCHAM-4”. In: *International Conference on Computational Methods in Systems Biology*. Springer. 2018, pp. 292–299.

[160] Georgios E. Fainekos and George J. Pappas. “Robust Sampling for MITL Specifications”. In: *FORMATS 2007: Proceedings of the Fifth International Conference on Formal Modeling and Analysis of Timed Systems*. Lecture Notes in Computer Science 4763. Springer, 2007, pp. 147–162.

[161] Cheng Feng and Jane Hillston. “PALOMA: A Process Algebra for Located Markovian Agents”. In: *Quantitative Evaluation of Systems: 11th International Conference, QEST 2014, Florence, Italy, September 8-10, 2014. Proceedings*. Ed. by Gethin Norman and William Sanders. Cham: Springer International Publishing, 2014, pp. 265–280. URL: https://doi.org/10.1007/978-3-319-10696-0_22.

[162] Raphael A. Finkel and Jon Louis Bentley. “Quad trees a data structure for retrieval on composite keys”. In: *Acta informatica* 4.1 (1974), pp. 1–9.

[163] Riccardo Fiorista. “Modelling Biochemical Components as Communicating Processes”. School of Informatics, University of Edinburgh.

[164] Karl Fogelmark and Carl Troein. “Rethinking Transcriptional Activation in the Arabidopsis Circadian Clock”. In: *PLOS Computational Biology* 10.7 (July 2014), pp. 1–12. URL: https://doi.org/10.1371/journal.pcbi.1003705.

[165] Walter Fontana and Leo W. Buss. ““The arrival of the fittest”: Toward a theory of biological organization”. In: *Bulletin of Mathematical Biology* 56.1 (1994), pp. 1–64. URL: http://www.sciencedirect.com/science/article/pii/S0092824005802058.

[166] Cédric Fournet and Georges Gonthier. "The Join Calculus: A Language for Distributed Mobile Programming". In: *Applied Semantics: International Summer School, APPSEM 2000 Caminha, Portugal, September 9–15, 2000 Advanced Lectures*. Ed. by Gilles Barthe et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 268–332. URL: https://doi.org/10.1007/3-540-45699-6_6.

[167] Goran Frehse. "PHAVer: algorithmic verification of hybrid systems past HyTech". In: *International Journal on Software Tools for Technology Transfer* 10.3 (2008), pp. 263–279.

[168] Goran Frehse et al. "SpaceEx: Scalable verification of hybrid systems". In: *International Conference on Computer Aided Verification*. Springer. 2011, pp. 379–395.

[169] Nathan Fulton et al. "KeYmaera X: An axiomatic tactical theorem prover for hybrid systems". In: *International Conference on Automated Deduction*. Springer. 2015, pp. 527–538.

[170] Dov Gabbay et al. "On the temporal analysis of fairness". In: *Proceedings of the 7th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*. 1980, pp. 163–173.

[171] David Gabelaia et al. "Combining spatial and temporal logics: expressiveness vs. complexity". In: *Journal of artificial intelligence research* 23 (2005), pp. 167–243.

[172] Vashti Galpin. "Hybrid semantics for Bio-PEPA". In: *Information and Computation* 236 (2014). Special Issue on Hybrid Systems and Biology, pp. 122–145. URL: https://www.sciencedirect.com/science/article/pii/S0890540114000170.

[173] Vashti Galpin. "Modelling a circadian clock with HYPE". In: *Proceedings of the 9th workshop on process algebra and stochastically timed activities (PASTA)*. 2010, pp. 92–98.

[174] Vashti Galpin, Luca Bortolussi, and Jane Hillston. "HYPE: a process algebra for compositional flows and emergent behaviour". In: *International Conference on Concurrency Theory*. Springer. 2009, pp. 305–320.

[175] Sicun Gao, Soonho Kong, and Edmund M Clarke. "dReal: An SMT solver for nonlinear theories over the reals". In: *International conference on automated deduction*. Springer. 2013, pp. 208–214.

[176] Laura Gardner and Alexander Deiters. “Light-controlled synthetic gene circuits”. eng. In: *Current opinion in chemical biology* 16.3-4 (Aug. 2012). S1367-5931(12)00059-2[PII], pp. 292–299. URL: https://doi.org/10.1016/j.cbpa.2012.04.010.

[177] Nil Geisweiller. “An attempt to give a clear semantics of the extension of PEPA for massively parallel processes and biological modelling”. In: *Process Algebra and Stochastically Timed Activities* (2006), pp. 36–43.

[178] Anastasios-Andreas Georgoulas. “Formal language for statistical inference of uncertain stochastic systems”. PhD thesis. University of Edinburgh, 2016.

[179] Anastasis Georgoulas, Jane Hillston, and Guid Sanguinetti. “ProPPA: Probabilistic Programming for Stochastic Dynamical Systems”. In: *ACM Transactions on Modeling and Computer Simulation (TOMACS)* 28.1 (2018), p. 3.

[180] Daniel T Gillespie. “Exact stochastic simulation of coupled chemical reactions”. In: *The Journal of Physical Chemistry* 81.25 (1977), pp. 2340–2361. URL: http://dx.doi.org/10.1021/j100540a008.

[181] Stephen Gilmore et al. “PEPA nets: a structured performance modelling formalism”. In: *Performance Evaluation* 54.2 (2003). Modelling Techniques and Tools for Computer Performance Evaluation, pp. 79–104. URL: https://www.sciencedirect.com/science/article/pii/S0166531603000695.

[182] Antoine Girard. “Reachability of uncertain linear systems using zonotopes”. In: *International Workshop on Hybrid Systems: Computation and Control*. Springer. 2005, pp. 291–305.

[183] Angel Goñi-Moreno and Martyn Amos. “A reconfigurable NAND/NOR genetic logic gate”. In: *BMC systems biology* 6.1 (2012), pp. 1–11.

[184] Radu Grosu et al. “Learning and detecting emergent behavior in networks of cardiac myocytes”. In: *Communications of the ACM* 52.3 (2009), pp. 97–105.

[185] Maria Luisa Guerriero et al. “Stochastic properties of the plant circadian clock”. In: *Journal of The Royal Society Interface* (2011). URL: http://rsif.royalsocietypublishing.org/content/early/2011/08/26/rsif.2011.0378.

[186] Jeremy Gunawardena. “Chemical reaction network theory for in-silico biologists”. In: (2003).

[187] Iman Haghighi et al. “SpaTeL: A Novel Spatial-Temporal Logic and Its Applications to Networked Systems”. In: *Proceedings of the 18th International Conference on Hybrid Systems: Computation and Control*. HSCC ’15. Seattle Washington: Association for Computing Machinery, 2015, pp. 189–198. URL: https://doi.org/10.1145/2728606.2728633.

[188] Iman Haghighi et al. “SpaTeL: a novel spatial-temporal logic and its applications to networked systems”. In: *Proceedings of the 18th International Conference on Hybrid Systems: Computation and Control*. 2015, pp. 189–198.

[189] Jeff Hasty et al. “Designer gene networks: Towards fundamental cellular control”. In: *Chaos: An Interdisciplinary Journal of Nonlinear Science* 11.1 (2001), pp. 207–220.

[190] Jeff Hasty et al. “Noise-based switches and amplifiers for gene expression”. In: *Proceedings of the National Academy of Sciences* 97.5 (2000), pp. 2075–2080.

[191] Matthew Hennessy and Robin Milner. “Algebraic Laws for Nondeterminism and Concurrency”. In: *J. ACM* 32.1 (Jan. 1985), pp. 137–161. URL: https://doi.org/10.1145/2455.2460.

[192] T.A. Henzinger, Pei-Hsin Ho, and H. Wong-Toi. “Algorithmic analysis of nonlinear hybrid systems”. In: *IEEE Transactions on Automatic Control* 43.4 (1998), pp. 540–554.

[193] Thomas A Henzinger, Pei-Hsin Ho, and Howard Wong-Toi. “HyTech: A model checker for hybrid systems”. In: *International Conference on Computer Aided Verification*. Springer. 1997, pp. 460–463.

[194] Thomas A. Henzinger et al. “Beyond HyTech: Hybrid Systems Analysis Using Interval Numerical Methods”. In: *Hybrid Systems: Computation and Control*. Ed. by Nancy Lynch and Bruce H. Krogh. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, pp. 130–144.

[195] AV. Hill. “The possible effects of the aggregation of the molecules of haemoglobin on its dissociation curves”. In: *J Physiol (Lond)* 40 (1910), pp. 4–7.

[196] J. Hillston. “Fluid Flow Approximation of PEPA Models”. In: *Proceedings of the Second International Conference on the Quantitative Evaluation of Systems*. QEST ’05. Washington, DC, USA: IEEE Computer Society, 2005, pp. 33–. URL: http://dx.doi.org/10.1109/QEST.2005.12.

[197] Jane Hillston. “Compositional Markovian modelling using a process algebra”. In: *Computations with Markov chains*. Springer, 1995, pp. 177–196.

[198] Jane Hillston. “Fluid flow approximation of PEPA models”. In: *Quantitative Evaluation of Systems, 2005. Second International Conference on the*. IEEE. 2005, pp. 33–42.

[199] Jane Hillston. “The nature of synchronisation”. In: *Proc. of 2nd Process Algebra and Performance Modelling Workshop*. Citeseer. 1994, p. 143.

[200] Jane Hillston and Leïla Kloul. “An efficient Kronecker representation for PEPA models”. In: *Process Algebra and Probabilistic Methods. Performance Modelling and Verification*. Springer, 2001, pp. 120–135.

[201] CAR Hoare. “Communicating Sequential Processes”. In: *Communications of the ACM* 21.8 (1978), pp. 666–677.

[202] Jan-Hendrik S Hofmeyr and Hofmeyr Cornish-Bowden. “The reversible Hill equation: how to incorporate cooperative enzymes into metabolic models”. In: *Bioinformatics* 13.4 (1997), pp. 377–385.

[203] J. D. Hunter. “Matplotlib: A 2D graphics environment”. In: *Computing in Science & Engineering* 9.3 (2007), pp. 90–95.

[204] Janine Imada and Brian J Ross. “Evolutionary synthesis of stochastic gene network models using feature-based search spaces”. In: *New Generation Computing* 29.4 (2011), pp. 365–390.

[205] Fabian Immler et al. “ARCH-COMP18 Category Report: Continuous and Hybrid Systems with Nonlinear Dynamics”. In: *ARCH18. 5th International Workshop on Applied Verification of Continuous and Hybrid Systems*. Ed. by Goran Frehse. Vol. 54. EPiC Series in Computing. EasyChair, 2018, pp. 53–70. URL: `https://easychair.org/publications/paper/gjfh`.

[206] Daisuke Ishii and Alexandre Goldsztejn. “HySIA: Tool for Simulating and Monitoring Hybrid Automata Based on Interval Analysis”. In: *International Conference on Runtime Verification*. Springer. 2017, pp. 370–379.

[207] Daisuke Ishii, Naoki Yonezaki, and Alexandre Goldsztejn. “Monitoring Bounded LTL Properties Using Interval Analysis”. In: *Electronic Notes in Theoretical Computer Science* 317 (2015). The Seventh and Eighth International Workshops on Numerical Software Verification (NSV), pp. 85–100.

[208] Daisuke Ishii, Naoki Yonezaki, and Alexandre Goldsztejn. “Monitoring temporal properties using interval analysis”. In: *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences* 99.2 (2016), pp. 442–453.

[209] Jean-Baptiste Jeannin and André Platzer. “dTL2: Differential Temporal Dynamic Logic with nested temporalities for hybrid systems”. In: *International Joint Conference on Automated Reasoning*. Springer. 2014, pp. 292–306.

[210] Mathias John. “Reaction Constraints for the Pi-Calculus-A Language for the Stochastic and Spatial Modeling of Cell-Biological Processes”. PhD thesis. Universität Rostock, 2010.

[211] Mathias John, Roland Ewald, and Adelinde M Uhrmacher. “A Spatial Extension to the π Calculus”. In: *Electronic notes in theoretical computer science* 194.3 (2008), pp. 133–148.

[212] Mathias John, Cédric Lhoussaine, and Joachim Niehren. “Dynamic Compartments in the Imperative Pi Calculus”. In: *Computational Methods in Systems Biology, 7th International Conference*. Vol. 5688. Bologna, Italy: Spinger, Aug. 2009, pp. 235–250. URL: https://hal.inria.fr/inria-00422970.

[213] Mathias John et al. “Biochemical reaction rules with constraints”. In: *European symposium on programming*. Springer. 2011, pp. 338–357.

[214] Mathias John et al. “Constructing and visualizing chemical reaction networks from pi-calculus models”. In: *Formal Aspects of Computing* 25.5 (Sept. 2013), pp. 723–742.

[215] Mathias John et al. “The Attributed Pi Calculus”. In: *Computational Methods in Systems Biology: 6th International Conference CMSB 2008, Rostock, Germany, October 12-15, 2008. Proceedings*. Ed. by Monika Heiner and Adelinde M. Uhrmacher. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 83–102. URL: https://doi.org/10.1007/978-3-540-88562-7_10.

[216] Mathias John et al. “The Attributed Pi-Calculus with Priorities”. In: *Transactions on Computational Systems Biology XII: Special Issue on Modeling Methodologies*. Ed. by Corrado Priami et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 13–76. URL: https://doi.org/10.1007/978-3-642-11712-1_2.

[217] Craig C Jolley, Koji L Ode, and Hiroki R Ueda. “A design principle for a post-translational biochemical oscillator”. In: *Cell reports* 2.4 (2012), pp. 938–950.

[218] Eric Jones, Travis Oliphant, Pearu Peterson, et al. *SciPy: Open source scientific tools for Python*. [Online; accessed 12-02-2018]. 2001. URL: http://www.scipy.org/.

[219] Hans Kamp. "Tense Logic and the Theory of Linear Order". PhD thesis. Ucla, 1968.

[220] Tomasz Kapela et al. "CAPD:: DynSys: a flexible C++ toolbox for rigorous numerical analysis of dynamical systems". In: *Communications in Nonlinear Science and Numerical Simulation* (2020), p. 105578.

[221] Edgar Kaucher. "Interval analysis in the extended interval space IR". In: *Fundamentals of Numerical Computation (Computer-Oriented Numerical Analysis)*. Springer, 1980, pp. 33–49.

[222] Hiroaki Kitano. "Biological robustness". In: *Nature Reviews Genetics* 5.11 (Nov. 2004), pp. 826–837. URL: https://doi.org/10.1038/nrg1471.

[223] Bartek Klin and Vladimiro Sassone. "Structural operational semantics for stochastic and weighted transition systems". In: *Information and Computation* 227 (2013), pp. 58–83. URL: https://www.sciencedirect.com/science/article/pii/S0890540113000497.

[224] Thomas Kluyver et al. *Jupyter Notebooks-a publishing format for reproducible computational workflows.* Vol. 2016. 2016.

[225] Saul Kripke. "Semantical Considerations Of The Modal Logic". In: *Studia Philosophica* 1 (2007).

[226] Jean Krivine, Robin Milner, and Angelo Troina. "Stochastic Bigraphs". In: *Electronic Notes in Theoretical Computer Science* 218 (2008). Proceedings of the 24th Conference on the Mathematical Foundations of Programming Semantics (MFPS XXIV), pp. 73–96. URL: http://www.sciencedirect.com/science/article/pii/S1571066108004003.

[227] Céline Kuttler. "Simulating Bacterial Transcription and Translation in a Stochastic π Calculus". In: *Transactions on Computational Systems Biology VI*. Ed. by Corrado Priami and Gordon Plotkin. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 113–149.

[228] Céline Kuttler, Cédric Lhoussaine, and Mirabelle Nebut. “Rule-Based Modeling of Transcriptional Attenuation at the Tryptophan Operon”. In: *Transactions on Computational Systems Biology XII: Special Issue on Modeling Methodologies*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 199–228.

[229] Céline Kuttler, Cédric Lhoussaine, and Joachim Niehren. “A stochastic pi calculus for concurrent objects”. In: *Ab* 4545 (2007), pp. 232–246.

[230] Céline Kuttler and Joachim Niehren. “Gene Regulation in the Pi Calculus: Simulating Cooperativity at the Lambda Switch”. In: *Transactions on Computational Systems Biology VII*. Ed. by Corrado Priami et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 24–55. URL: https://doi.org/10.1007/11905455_2.

[231] M. Kwiatkowska, G. Norman, and D. Parker. “PRISM 4.0: Verification of Probabilistic Real-time Systems”. In: *Proc. 23rd International Conference on Computer Aided Verification (CAV'11)*. Ed. by G. Gopalakrishnan and S. Qadeer. Vol. 6806. LNCS. Springer, 2011, pp. 585–591.

[232] Marek Kwiatkowski. “A formal computational framework for the study of molecular evolution”. Supervised by Ian Stark. PhD thesis. Edinburgh, 2010.

[233] Marek Kwiatkowski and Ian Stark. “On Executable Models of Molecular Evolution”. In: *Proceedings of the 8th International Workshop on Computational Systems Biology WCSB 2011*. TICSP Report 57. Tampere International Centre for Signal Processing, 2011, pp. 105–108.

[234] Marek Kwiatkowski and Ian Stark. “The Continuous π-Calculus: A Process Algebra for Biochemical Modelling”. In: *Computational Methods in Systems Biology: Process of the Sixth International Conference CMSB 2008*. Lecture Notes in Computer Science 5307. Springer-Verlag, 2008, pp. 103–122. URL: http://homepages.ed.ac.uk/stark/continuous-pi.html.

[235] Roberto Larcher, Corrado Priami, and Alessandro Romanel. “Modelling self-assembly in BlenX”. In: *Transactions on Computational Systems Biology XII*. Springer, 2010, pp. 163–198.

[236] Kim G. Larsen, Radu Mardare, and Bingtian Xue. “Concurrent Weighted Logic”. In: *Journal of Logical and Algebraic Methods in Programming* 84.6 (2015). “Special Issue on Open Problems in Concurrency Theory”, pp. 884–897. URL: https://www.sciencedirect.com/science/article/pii/S2352220815000590.

[237] Colas Le Guernic and Antoine Girard. “Reachability analysis of linear systems using support functions”. In: *Nonlinear Analysis: Hybrid Systems* 4.2 (2010), pp. 250–262.

[238] Axel Legay et al. “Statistical Model Checking”. In: *Computing and Software Science: State of the Art and Perspectives*. Ed. by Bernhard Steffen and Gerhard Woeginger. Cham: Springer International Publishing, 2019, pp. 478–504. URL: https://doi.org/10.1007/978-3-319-91908-9_23.

[239] T. Li et al. “Model Checking of Spatial Logic”. In: *2020 27th Asia-Pacific Software Engineering Conference (APSEC)*. 2020, pp. 169–177.

[240] Kristjan Liiva et al. “Compositional Taylor model based validated integration”. In: *2018 20th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC)*. IEEE. 2018, pp. 45–52.

[241] Youdong Lin and Mark A. Stadtherr. “Validated solutions of initial value problems for parametric ODEs”. In: *Applied Numerical Mathematics* 57.10 (2007), pp. 1145–1162. URL: http://www.sciencedirect.com/science/article/pii/S0168927406002030.

[242] Jiang Liu, Naijun Zhan, and Hengjun Zhao. “Computing semi-algebraic invariants for polynomial dynamical systems”. In: *Proceedings of the ninth ACM international conference on Embedded software*. ACM. 2011, pp. 97–106.

[243] Jiang Liu et al. “A calculus for hybrid CSP”. In: *Asian Symposium on Programming Languages and Systems*. Springer. 2010, pp. 1–15.

[244] Zhiyu Liu, Meng Jiang, and Hai Lin. “Specification mining and automated task planning for autonomous robots based on a graph-based spatial temporal logic”. In: *arXiv preprint arXiv:2007.08451* (2020).

[245] Zhiyu Liu et al. “Distributed communication-aware motion planning for multi-agent systems from stl and spatel specifications”. In: *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*. IEEE. 2017, pp. 4452–4457.

[246] J.C.W. Locke, A.J. Millar, and M.S. Turner. “Modelling genetic networks with noisy and varied experimental data: the circadian clock in Arabidopsis thaliana”. In: *Journal of Theoretical Biology* 234.3 (2005), pp. 383–393. URL: http://www.sciencedirect.com/science/article/pii/S0022519304005879.

[247] Rudolf J. Lohner. “On the Ubiquity of the Wrapping Effect in the Computation of Error Bounds”. In: *Perspectives on Enclosure Methods*. Ed. by Ulrich Kulisch, Rudolf Lohner, and Axel Facius. Vienna: Springer Vienna, 2001, pp. 201–216. URL: https://doi.org/10.1007/978-3-7091-6282-8_12.

[248] Carlos F Lopez et al. “Programming biological models in Python using PySB”. In: *Molecular Systems Biology* 9.1 (2013), p. 646. URL: https://www.embopress.org/doi/abs/10.1038/msb.2013.1.

[249] Ludovica Luisa Vissat et al. “Automatic verification of reliability requirements of spatio-temporal analysis using Three-Valued Spatio-Temporal Logic”. English. In: *Proceedings of the 11th EAI International Conference on Performance Evaluation Methodologies and Tools*. ACM, Dec. 2017, pp. 225–226.

[250] Timo R. Maarleveld, Brett G. Olivier, and Frank J. Bruggeman. “StochPy: A Comprehensive, User-Friendly Tool for Simulating Stochastic Biological Processes”. In: *PLOS ONE* 8.11 (Nov. 2013), pp. 1–10. URL: https://doi.org/10.1371/journal.pone.0079345.

[251] Kyoko Makino and Martin Berz. “Suppression of the wrapping effect by Taylor model-based verified integrators: Long-term stabilization by preconditioning”. In: *International Journal of Differential Equations and Applications* 10.4 (2011).

[252] Kyoko Makino and Martin Berz. “Taylor models and other validated functional inclusion methods”. In: *International Journal of Pure and Applied Mathematics* 6 (2003), pp. 239–316.

[253] Oded Maler and Dejan Nickovic. “Monitoring temporal properties of continuous signals”. In: *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems*. Springer, 2004, pp. 152–166.

[254] Oded Maler, Dejan Nickovic, and Amir Pnueli. “Checking Temporal Properties of Discrete, Timed and Continuous Behaviors”. In: *Pillars of Computer Science: Essays Dedicated to Boris (Boaz) Trakhtenbrot on the Occasion of His 85th Birthday*. Ed. by Arnon Avron, Nachum Dershowitz, and Alexander Rabinovich. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 475–505.

[255] Radu Mardare and Corrado Priami. *A decidable extension of Hennessy-Milner logic with spatial operators*. Tech. rep. 2006.

[256] SM Markov. “Extended interval arithmetic involving infinite intervals”. In: *Math. Balkanika, New Ser* 6 (1992), pp. 269–304.

[257] Svetoslav Markov. "On directed interval arithmetic and its applications". In: *J. UCS The Journal of Universal Computer Science*. Springer, 1996, pp. 514–526.

[258] Harley H McAdams and Adam Arkin. "Stochastic mechanisms in gene expression". In: *Proceedings of the National Academy of Sciences* 94.3 (1997), pp. 814–819.

[259] Chris McCaig, Rachel Norman, and Carron Shankland. "From individuals to populations: A symbolic process algebra approach to epidemiology". In: *Mathematics in Computer Science* 2.3 (2009), pp. 535–556.

[260] Chris McCaig et al. "A rigorous approach to investigating common assumptions about disease transmission". In: *Theory in Biosciences* 130.1 (2011), pp. 19–29.

[261] Chris McCaig et al. "Using process algebra to develop predator–prey models of within-host parasite dynamics". In: *Journal of theoretical biology* 329 (2013), pp. 74–81.

[262] Kenneth L McMillan. "Symbolic model checking". In: *Symbolic Model Checking*. Springer, 1993, pp. 25–60.

[263] Aaron Meurer et al. "SymPy: symbolic computing in Python". In: *PeerJ Computer Science* 3 (2017), e103.

[264] Marino Miculan and Giorgio Bacci. "Modal logics for brane calculus". In: *International Conference on Computational Methods in Systems Biology*. Springer. 2006, pp. 1–16.

[265] Robin Milner. "A calculus of communicating systems". In: (1980).

[266] Robin Milner. "Bigraphical Reactive Systems". In: *CONCUR 2001 — Concurrency Theory*. Ed. by Kim G. Larsen and Mogens Nielsen. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 16–35.

[267] Robin Milner. *Communicating and mobile systems: the π-calculus*. Cambridge university press, 1999.

[268] Robin Milner. "The polyadic π-calculus: a tutorial". In: *Logic and algebra of specification*. Springer, 1993, pp. 203–246.

[269] Robin Milner, Joachim Parrow, and David Walker. "A calculus of mobile processes, I". In: *Information and Computation* 100.1 (1992), pp. 1–40. URL: http://dx.doi.org/10.1016/0890-5401(92)90008-4.

[270] Robin Milner, Joachim Parrow, and David Walker. "A calculus of mobile processes, II". In: *Information and Computation* 100.1 (1992), pp. 41–77. URL: http://dx.doi.org/10.1016/0890-5401(92)90009-5.

[271] Ramon E Moore, R Baker Kearfott, and Michael J Cloud. *Introduction to interval analysis*. Vol. 110. Siam, 2009.

[272] Masao Nagasaki et al. "Bio-calculus: Its concept and molecular interaction". In: *Genome Informatics* 10 (1999), pp. 133–143.

[273] Dawn H. Nagel and Steve A. Kay. "Complexity in the Wiring and Regulation of Plant Circadian Networks". In: *Current Biology* 22.16 (2012), R648–R657. URL: http://www.sciencedirect.com/science/article/pii/S0960982212008160.

[274] *National Center for Biotechnology Information. PubChem Compound Database; CID=177674, https://pubchem.ncbi.nlm.nih.gov/compound/177674 (accessed Jan. 8, 2018).*

[275] Nedialko S Nedialkov. "Implementing a rigorous ODE solver through literate programming". In: *Modeling, Design, and Simulation of Systems with Uncertainties*. Springer, 2011, pp. 3–19.

[276] Markus Neher, Kenneth R Jackson, and Nedialko S Nedialkov. "On Taylor model based integration of ODEs". In: *SIAM Journal on Numerical Analysis* 45.1 (2007), pp. 236–262.

[277] L. Nenzi et al. "Qualitative and Quantitative Monitoring of Spatio-Temporal Properties with SSTL". In: *Logical Methods in Computer Science* Volume 14, Issue 4 (Oct. 2018). URL: https://lmcs.episciences.org/4913.

[278] Uwe Nestmann. "Welcome to the Jungle: A Subjective Guide to Mobile Process Calculi". In: *CONCUR 2006 – Concurrency Theory*. Ed. by Christel Baier and Holger Hermanns. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 52–63.

[279] Dejan Nickovic and Oded Maler. "AMT: A property-based monitoring tool for analog systems". In: *International Conference on Formal Modeling and Analysis of Timed Systems*. Springer. 2007, pp. 304–319.

[280] Rocco de Nicola et al. "A uniform definition of stochastic process calculi". In: *ACM Computing Surveys (CSUR)* 46.1 (2013), pp. 1–35.

[281] Rachel Norman and Carron Shankland. “Developing the use of process algebra in the derivation and analysis of mathematical models of infectious disease”. In: *International Conference on Computer Aided Systems Theory*. Springer. 2003, pp. 404–414.

[282] Peter Øhrstrøm and Per Hasle. “AN Prior’s rediscovery of tense logic”. In: *Erkenntnis* 39.1 (1993), pp. 23–50.

[283] Catuscia Palamidessi and Frank Valencia. “Recursion vs replication in process calculi: Expressiveness”. In: *Bulletin-European Association for Theoretical Computer Science* 87 (2005), pp. 105–125.

[284] Loïc Paulevé, Morgan Magnin, and Olivier Roux. “Refining dynamics of gene regulatory networks in a stochastic π-calculus framework”. In: *Transactions on computational systems biology xiii*. Springer, 2011, pp. 171–191.

[285] Michael Pedersen, Andrew Phillips, and Gordon D Plotkin. “A high-level language for rule-based modelling”. In: *PloS one* 10.6 (2015), e0114296.

[286] Michael Pedersen and Gordon D Plotkin. “A language for biochemical systems: Design and formal specification”. In: *Transactions on computational systems biology XII*. Springer, 2010, pp. 77–145.

[287] Francisco Penedo, Harold Park, and Calin Belta. “Control synthesis for partial differential equations from spatio-temporal specifications”. In: *2018 IEEE Conference on Decision and Control (CDC)*. IEEE. 2018, pp. 4890–4895.

[288] Daniela Paula Petrinca and Eneia Nicolae Todoran. “Immune System Modeling and Analysis using CARMA”. In: *2018 IEEE 14th International Conference on Intelligent Computer Communication and Processing (ICCP)*. 2018, pp. 409–416.

[289] Anna Philippou, Mauricio Toro, and Margarita Antonaki. “Simulation and Verification in a Process Calculus for Spatially-Explicit Ecological Models.” In: *Scientific Annals of Computer Science* 23.1 (2013).

[290] Andrew Phillips. “An abstract machine for the stochastic bioambient calculus”. In: *Electronic Notes in Theoretical Computer Science* 227 (2009), pp. 143–159.

[291] Andrew Phillips and Luca Cardelli. “A correct abstract machine for the stochastic pi-calculus”. In: *In Bioconcur’04. ENTCS* (2004).

[292] Carla Piazza et al. “Algorithmic algebraic model checking I: Challenges from systems biology”. In: *International Conference on Computer Aided Verification*. Springer. 2005, pp. 5–19.

[293] Andrew M Pitts. “Nominal logic, a first order theory of names and binding”. In: *Information and computation* 186.2 (2003), pp. 165–193.

[294] André Platzer. “A temporal dynamic logic for verifying hybrid system invariants”. In: *International Symposium on Logical Foundations of Computer Science*. Springer. 2007, pp. 457–471.

[295] André Platzer. “Differential dynamic logic for hybrid systems”. In: *Journal of Automated Reasoning* 41.2 (2008), pp. 143–189.

[296] André Platzer. “Quantified differential dynamic logic for distributed hybrid systems”. In: *International Workshop on Computer Science Logic*. Springer. 2010, pp. 469–483.

[297] André Platzer and Edmund M Clarke. “Computing differential invariants of hybrid systems as fixedpoints”. In: *International Conference on Computer Aided Verification*. Springer. 2008, pp. 176–189.

[298] Gordon D Plotkin. “A structural approach to operational semantics”. In: (1981).

[299] Thomas Ploug and Peter Øhrstrøm. “Branching time, indeterminism and tense logic”. In: *Synthese* 188.3 (2012), pp. 367–379.

[300] Amir Pnueli. “The temporal logic of programs”. In: *18th Annual Symposium on Foundations of Computer Science (sfcs 1977)*. IEEE, pp. 46–57.

[301] Alexandra Pokhilko, Paloma Mas, and Andrew J. Millar. “Modelling the widespread effects of TOC1 signalling on the plant circadian clock and its outputs”. In: *BMC Systems Biology* 7.1 (Mar. 2013), p. 23.

[302] Alexandra Pokhilko et al. “Data assimilation constrains new connections and components in a complex, eukaryotic circadian clock model”. In: *Molecular Systems Biology* 6.1 (2010). URL: `http://msb.embopress.org/content/6/1/416`.

[303] C. Priami. “Stochastic π-Calculus”. In: *The Computer Journal* 38.7 (1995), pp. 578–589. URL: `http://comjnl.oxfordjournals.org/content/38/7/578.abstract`.

[304] Corrado Priami and Paola Quaglia. “Beta Binders for Biological Interactions”. In: *Computational Methods in Systems Biology: International Conference CMSB 2004, Paris, France, May 26-28, 2004, Revised Selected Papers*. Ed. by Vincent Danos and Vincent Schachter. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 20–33. URL: http://dx.doi.org/10.1007/978-3-540-25974-9_3.

[305] Corrado Priami et al. “Application of a stochastic name-passing calculus to representation and simulation of molecular processes”. In: *Information Processing Letters* 80.1 (Jan. 2001), pp. 25–31. URL: http://dx.doi.org/10.1016/s0020-0190(01)00214-9.

[306] Arthur N Prior. *Time and Modality*. Greenwood Press, 1955.

[307] David Pym and Chris Tofts. “A calculus and logic of resources and processes”. In: *Formal Aspects of Computing* 18.4 (2006), pp. 495–517.

[308] Jean-Pierre Queille and Joseph Sifakis. “Specification and verification of concurrent systems in CESAR”. In: *International Symposium on programming*. Springer. 1982, pp. 337–351.

[309] David A Randell, Zhan Cui, and Anthony G Cohn. “A spatial logic based on regions and connection.” In: *KR* 92 (1992), pp. 165–176.

[310] Aviv Regev, William Silverman, and Ehud Shapiro. “Representation and simulation of biochemical processes using the π-calculus process algebra”. In: *Pacific symposium on biocomputing*. Vol. 6. 2001, pp. 459–470.

[311] Aviv Regev et al. “BioAmbients: an abstraction for biological compartments”. In: *Theoretical Computer Science* 325.1 (Apr. 2004), pp. 141–167. URL: http://dx.doi.org/10.1016/j.tcs.2004.03.061.

[312] Michael Rettelbach and Markus Siegle. “Compositional minimal semantics for the stochastic process algebra TIPP”. In: *Proc. of 2nd Process Algebra and Performance Modelling Workshop*. 1994, p. 143.

[313] John C Reynolds. “Separation logic: A logic for shared mutable data structures”. In: *Proceedings 17th Annual IEEE Symposium on Logic in Computer Science*. IEEE. 2002, pp. 55–74.

[314] Aurélien Rizk et al. “On a continuous degree of satisfaction of temporal logic formulae with applications to systems biology”. In: *International Conference on Computational Methods in Systems Biology*. Springer. 2008, pp. 251–268.

[315] Hendrik Roehm et al. “STL model checking of continuous and hybrid systems”. In: *International Symposium on Automated Technology for Verification and Analysis*. Springer. 2016, pp. 412–427.

[316] B. J. Ross. “Using multi-objective genetic programming to evolve stochastic logic gate circuits”. In: *2015 IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB)*. Aug. 2015, pp. 1–8.

[317] William C Rounds and Hosung Song. “The φ-calculus-a hybrid extension of the π-calculus to embedded systems”. In: *Eighteenth Workshop on the Mathematical Foundations of Programming Semantics*. 2002.

[318] William C. Rounds. “A Spatial Logic for the Hybrid π-Calculus”. In: *Hybrid Systems: Computation and Control*. Ed. by Rajeev Alur and George J. Pappas. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 508–522.

[319] Andreas Ruscheinski et al. “Pragmatic Logic-Based Spatio-Temporal Pattern Checking in Particle-Based Models”. In: *2020 Winter Simulation Conference (WSC)*. 2020, pp. 2245–2256.

[320] José Domingo Salazar et al. “Prediction of Photoperiodic Regulators from Quantitative Gene Circuit Models”. In: *Cell* 139.6 (2009), pp. 1170–1179.

[321] Faustino Sánchez-Garduño, Pedro Miramontes, and Tatiana T Marquez-Lago. “Role reversal in a predator–prey interaction”. In: *Royal Society open science* 1.2 (2014), p. 140186.

[322] Davide Sangiorgi. “π-calculus, internal mobility, and agent-passing calculi”. In: *Theoretical Computer Science* 167.1-2 (1996), pp. 235–274.

[323] Davide Sangiorgi. “Extensionality and intensionality of the ambient logics”. In: *ACM SIGPLAN Notices* 36.3 (2001), pp. 4–13.

[324] Sriram Sankaranarayanan and Ashish Tiwari. “Relational Abstractions for Continuous and Hybrid Systems”. In: *Computer Aided Verification*. Ed. by Ganesh Gopalakrishnan and Shaz Qadeer. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 686–702.

[325] Moisés Santillán and Michael C Mackey. “Why the lysogenic state of phage λ is so stable: a mathematical modeling approach”. In: *Biophysical journal* 86.1 (2004), pp. 75–84.

[326] Andreas Schäfer. “A calculus for shapes in time and space”. In: *International Colloquium on Theoretical Aspects of Computing*. Springer. 2004, pp. 463–477.

[327] Christoph Schmal, Jean-Christophe Leloup, and Didier Gonze. “Modeling and Simulating the Arabidopsis thaliana Circadian Clock Using XPP-AUTO”. In: *Plant Circadian Networks: Methods and Protocols*. Ed. by Dorothee Staiger. New York, NY: Springer New York, 2014, pp. 337–358. URL: https://doi.org/10.1007/978-1-4939-0700-7_23.

[328] Andrew Schumann and Andy Adamatzky. “Physarum spatial logic”. In: *New Mathematics and Natural Computation* 7.03 (2011), pp. 483–498.

[329] Erin Scott, Andrew Hoyle, and Carron Shankland. “PEPA’d Oysters: Converting Dynamic Energy Budget Models to Bio-PEPA, Illustrated by a Pacific Oyster Case Study”. In: *Electronic Notes in Theoretical Computer Science* 296 (2013). Proceedings the Sixth International Workshop on the Practical Application of Stochastic Modelling (PASM) and the Eleventh International Workshop on Parallel and Distributed Methods in Verification (PDMC)., pp. 211–228. URL: https://www.sciencedirect.com/science/article/pii/S1571066113000455.

[330] Erin Scott et al. “Process Algebra with Layers: Multi-scale Integration Modelling Applied to Cancer Therapy”. In: *International Meeting on Computational Intelligence Methods for Bioinformatics and Biostatistics*. Springer. 2016, pp. 118–133.

[331] Madeline A Shea and Gary K Ackers. “The OR control system of bacteriophage lambda: A physical-chemical model for gene regulation”. In: *Journal of molecular biology* 181.2 (1985), pp. 211–230.

[332] Andrew Sogokon. “Direct methods for deductive verification of temporal properties in continuous dynamical systems”. PhD thesis. University of Edinburgh, 2016.

[333] Andrew Sogokon, Paul B Jackson, and Taylor T Johnson. “Verifying safety and persistence properties of hybrid systems using flowpipes and continuous invariants”. In: *NASA Formal Methods Symposium*. Springer. 2017, pp. 194–211.

[334] Andrew Sogokon et al. “Pegasus: A Framework for Sound Continuous Invariant Generation”. In: *Formal Methods – The Next 30 Years*. Ed. by Maurice H. ter Beek, Annabelle McIver, and José N. Oliveira. Cham: Springer International Publishing, 2019, pp. 138–157.

[335] Anton Stefanek, Maria Vigliotti, and Jeremy T Bradley. “Spatial extension of stochastic pi calculus”. In: *8th Workshop on Process Algebra and Stochastically Timed Activities*. 2009, pp. 109–117.

[336] William Stein and David Joyner. “Sage: System for algebra and geometry experimentation”. In: *Acm Sigsam Bulletin* 39.2 (2005), pp. 61–64.

[337] Steven Strogatz et al. “Nonlinear Dynamics and Chaos: With Applications to Physics, Biology, Chemistry, and Engineering”. In: *Computers in Physics* 8.5 (1994), pp. 532–532.

[338] Thomas Sturm. “A survey of some methods for real quantifier elimination, decision, and satisfiability and their applications”. In: *Mathematics in Computer Science* 11.3 (2017), pp. 483–502.

[339] Haiying Sun et al. “Specifying cyber physical system safety properties with metric temporal spatial logic”. In: *2015 Asia-Pacific Software Engineering Conference (APSEC)*. IEEE. 2015, pp. 254–260.

[340] Stefano Taschini. “Interval arithmetic: python implementation and applications”. In: *Proc 7th Python Sci Conf (ScyPy 2008)*. 2008.

[341] The pandas development team. *pandas-dev/pandas: Pandas*. Version latest. Feb. 2020. URL: https://doi.org/10.5281/zenodo.3509134.

[342] Tianhai Tian and Kevin Burrage. “Bistability and switching in the lysis/lysogeny genetic regulatory network of bacteriophage λ”. In: *Journal of Theoretical Biology* 227.2 (2004), pp. 229–237.

[343] CHOU Ting-Chao and Paul TaLaLay. “Generalized equations for the analysis of inhibitions of Michaelis-Menten and higher-order kinetic systems with two or more mutually exclusive and nonexclusive inhibitors”. In: *European journal of biochemistry* 115.1 (1981), pp. 207–216.

[344] C Tofts. “Using process algebra to describe social insect behaviour”. In: *Transactions of the Society for Computer Simulation* 9.4 (1993), pp. 227–283.

[345] Mauricio Toro. “A general overview of formal languages for individual-based modelling of ecosystems”. In: *Journal of Logical and Algebraic Methods in Programming* 104 (2019), pp. 117–126. URL: https://www.sciencedirect.com/science/article/pii/S2352220818300932.

[346] Mirco Tribastone, Stephen Gilmore, and Jane Hillston. "Scalable differential analysis of process algebra models". In: *IEEE Transactions on Software Engineering* 38.1 (2012), pp. 205–219.

[347] Matej Troják et al. "Executable biochemical space for specification and analysis of biochemical systems". In: *Electronic Notes in Theoretical Computer Science* 350 (2020), pp. 91–116.

[348] Max Tschaikowski. "Over-Approximation of Fluid Models". In: *CoRR* abs/1705.00530 (2017). URL: http://arxiv.org/abs/1705.00530.

[349] Alan Mathison Turing. "The chemical basis of morphogenesis". In: *Philosophical Transactions of the Royal Society of London. Series B, Biological Sciences* 237.641 (1952), pp. 37–72.

[350] John J Tyson, Katherine C Chen, and Bela Novak. "Sniffers, buzzers, toggles and blinkers: dynamics of regulatory and signaling pathways in the cell". In: *Current opinion in cell biology* 15.2 (2003), pp. 221–231.

[351] Jacob VanderPlas et al. "Altair: Interactive Statistical Visualizations for Python". In: *Journal of Open Source Software* 3.32 (2018), p. 1057. URL: https://doi.org/10.21105/joss.01057.

[352] Moshe Y Vardi and Pierre Wolper. "An automata-theoretic approach to automatic program verification". In: *Proceedings of the First Symposium on Logic in Computer Science*. IEEE Computer Society. 1986, pp. 322–331.

[353] Cristian Versari and Roberto Gorrieri. "π@: A π-Based Process Calculus for the Implementation of Compartmentalised Bio-inspired Calculi". In: *Formal Methods for Computational Systems Biology: 8th International School on Formal Methods for the Design of Computer, Communication, and Software Systems, SFM 2008 Bertinoro, Italy, June 2-7, 2008 Advanced Lectures*. Ed. by Marco Bernardo, Pierpaolo Degano, and Gianluigi Zavattaro. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 449–506. URL: https://doi.org/10.1007/978-3-540-68894-5_13.

[354] José M. G. Vilar et al. "Mechanisms of noise-resistance in genetic oscillators". In: *Proceedings of the National Academy of Sciences* 99.9 (2002), pp. 5988–5992.

[355] Ludovica Luisa Vissat et al. "MELA: Modelling in Ecology with Location Attributes". In: *arXiv preprint arXiv:1610.08171* (2016).

[356] Qinsi Wang et al. "SReach: A bounded model checker for stochastic hybrid systems". In: *arXiv preprint arXiv:1404.7206* (2014).

[357] James N Weiss. "The Hill equation revisited: uses and misuses." In: *The FASEB Journal* 11.11 (1997), pp. 835–841.

[358] Thomas Wright. "A Process-Algebraic Approach to Modelling Multiagent Interaction Dynamics in Biological Systems". MA thesis. The University of Edinburgh, 2017.

[359] *Modelling Patterns of Gene Regulation in the bond-calculus*. Vol. 350. 2020, pp. 117–138. URL: https://www.sciencedirect.com/science/article/pii/S1571066120300359.

[360] Thomas Wright and Ian Stark. "Property-Directed Verified Monitoring of Signal Temporal Logic". In: *Runtime Verification*. Ed. by Jyotirmoy Deshmukh and Dejan Ničković. Cham: Springer International Publishing, 2020, pp. 339–358.

[361] Michael Zakharyaschev and F Wolter. "Spatio-temporal representation and reasoning based on RCC-8". In: *Principles of Knowledge Representation and Reasoning: Proceedings of the Seventh International Conference (KR2000), Breckenridge, Colorado, April 12-15 2000*. Morgan Kaufmann Pub. 2000, p. 3.

[362] Melanie N Zeilinger et al. "A novel computational model of the circadian clock in Arabidopsis that incorporates PRR7 and PRR9". In: *Molecular Systems Biology* 2.1 (2006). URL: http://msb.embopress.org/content/2/1/58.

[363] Bingzhuo Zhong, Claudius Jordan, and Julien Provost. "Extending Signal Temporal Logic with Quantitative Semantics by Intervals for Robust Monitoring of Cyber-physical Systems". In: *ACM Transactions on Cyber-Physical Systems* 5.2 (2021), pp. 1–25.

[364] Roberto Zunino et al. "ℓ: An Imperative DSL to Stochastically Simulate Biological Systems". In: *Programming Languages with Applications to Biology and Security: Essays Dedicated to Pierpaolo Degano on the Occasion of His 65th Birthday*. Ed. by Chiara Bodei, Gianluigi Ferrari, and Corrado Priami. Cham: Springer International Publishing, 2015, pp. 354–374. URL: https://doi.org/10.1007/978-3-319-25527-9_23.