Formal Methods for Biological Systems: Languages, Algorithms, and Applications

Qinsi Wang

CMU-CS-16-129

September 2016

School of Computer Science
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA

Thesis Committee
Edmund M. Clarke, Chair
Stephen Brookes
Marta Zofia Kwiatkowska, University of Oxford
Frank Pfenning
Natasa Miskov-Zivanov, University of Pittsburgh

Submitted in partial fulfillment of the requirements for the Degree of Doctor of Philosophy

This research was sponsored by the National Science Foundation under grant numbers CNS-0926181 and CNS-1035813, the Army Research Laboratory under grant numbers FA95501210146 and FA955015C0030, the Defense Advanced Research Projects Agency under grant number FA875012C0204, the Office of Naval Research under grant number N000141310090, the Semiconductor Research Corporation under grant number 2008-TJ-1860, and the Microelectronics Advanced Research Corporation (DARPA) under grant number 2009-DT-2049.

The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of any sponsoring institution, the U.S. government or any other entity.

Keywords: Model checking, Formal specification, Formal Analysis, Boolean networks, Qualitative networks, Rule-based modeling, Multiscale hybrid rule-based modeling, Hybrid systems, Stochastic hybrid systems, Symbolic model checking, Bounded model checking, Statistical model checking, Bounded reachability, Probabilistic bounded reachability, Parameter estimation, Sensitivity analysis, Statistical tests, Pancreatic cancer, Phage-based bacteria killing, Prostate cancer treatment, C. elegans

For My Beloved Mom & Dad

Abstract

As biomedical research advances into more complicated systems, there is an increasing need to model and analyze these systems to better understand them. Formal specification and analyzing methods, such as model checking techniques, hold great promise in helping further discovery and innovation for complicated biochemical systems. Models can be tested and adapted inexpensively in-silico providing new insights. However, development of accurate and efficient modeling methodologies and analysis techniques are still open challenges for biochemical systems. This thesis is focused on designing appropriate modeling formalisms and efficient analyzing algorithms for various biological systems in three different thrusts:

- **Modeling Formalisms**: we have designed a multi-scale hybrid rule-based modeling formalism to depict intra- and intercellular dynamics using discrete and continuous variables respectively. Its hybrid characteristic inherits advantages of logic and kinetic modeling approaches.
- **Formal Analyzing Algorithms**: 1) We have developed a LTL model checking algorithm for Qualitative Networks (QNs). It considers the unique feature of QNs and combines it with over-approximation to compute decreasing sequences of reachability set, resulting in a more scalable method. 2) We have developed a formal analyzing method to handle probabilistic bounded reachability problems for two kinds of stochastic hybrid systems considering uncertainty parameters and probabilistic jumps. It combines a SMT-based model checking technique with statistical tests in a sound manner. Compared to standard simulation-based methods, it supports non-deterministic branching, increases the coverage of simulation, and avoids the zero-crossing problem. 3) We have designed a new framework, where formal methods and machine learning techniques take joint efforts to enhance the understanding of biological and biomedical systems. Within this framework, statistical model checking is used as a (sub)model selection method.

- **Applications:** To check the feasibility of our model language and algorithms, we have 1) constructed Boolean Network models for the signaling network for single pancreatic cancer cell, and used symbolical model checking to analyze these models, 2) built Qualitative Network models describing cellular interactions during skin cells' differentiation, and applied our improved bounded LTL model checking technique, 3) developed a multi-scale hybrid rule-based model for the pancreatic cancer micro-environment, and employed statistical model checking, 4) created a nonlinear hybrid model to depict a bacteria-killing process, and adopted a recently promoted δ-complete decision procedure-based model checking technique, 5) extended hybrid models for atrial fibrillation, prostate cancer treatment, and our bacteria-killing process into stochastic hybrid models, and applied our probabilistic bounded reachability analyzer SReach, and 6) carried out the probabilistic reachability analysis of the tap withdrawal circuit in C. elegans using SReach.

Acknowledgments

I am greatly indebted to my advisor, Edmund Clarke. Ed is the very first person who pointed out the importance of developing appropriate formal methods that can benefit the study of biological systems. Through the years he has guided me and steered my works to the current state. Without the insight, guidance, and encouragement from him, this thesis would not have been possible.

The other members of my committee, Stephen Brookes, Marta Zofia Kwiatkowska, Frank Pfenning, and Natasa Miskov-Zivanov, have greatly helped me with polishing the thesis. Steve and Frank gave me many insightful comments with regard to my thesis. Marta offered suggestions for the further development of probabilistic reachability analysis for general stochastic hybrid systems. Natasa encouraged me with her enthusiasm for the work, and enlightened me with many new points of view from wider contexts.

My work has been helped and improved significantly under the help of many other researchers in the field: James Faeder, Jasmin Fisher, Sicun Gao, Samin Ishtiaq, Md. Ariful Islam, Soonho Kong, Kai-Wen Liang, Bing Liu, Michael T. Lotze, Nir Piterman, Cheryl A. Telmer, Ronald Watro, and Paolo Zuliani are among a long list of people who have contributed to the formation of my work.

Martha Clarke, Charlotte Yano, and Denny Marous have patiently taken care of a lot of details involved in our daily research activities to make the work possible.

I thank my parents, Qing Wang and Yaping Lu, for their constant support for me as well as the serious efforts in understanding my unbalanced work and life. My friends in Pittsburgh and around the globe added much warmth and color to my otherwise dryly symbolic world.

Declaration of Collaboration

The work documented in this thesis has benefited from collaboration with many others. Specifically:

- Edmund Clarke has contributed to all topics in the thesis.
- Haijun Gong has contributed much to results in Chapter 2, as reflected in our published joint papers [100, 101].
- Nir Piterman and Samin Ishtiaq has contributed much to checking results in Chapter 3, as reflected in our published joint paper [61].

- Cheryl A. Telmer and Natasa Miskov-Zivanov have led the experimental design of our phage-based bacteria-killing procedure, which offers insights about the corresponding dynamics and values of important system parameters that are critical to the construction of the hybrid automaton for this procedure in Chapter 4.

- Md. Ariful Islam led the third case study of SReach - analyzing the tap withdrawal circuit in C. elegans, as discussed in Chapter 5.4.3, and reflected in our published joint paper [129].

- Michael T. Lotze and Bing Liu have helped me much through discussions when constructing the microenvironmental model in Chapter 6.

- For the experiment part in Chapter 7, the auto-reading output is offered by Peter Spirtes. I provide the baseline model, a set of system properties as the final selection standard, and heuristics to generate extended models. The statistical model checking is carried out by Kai-Wen Liang. The graphic presentation of the checking results (Figure 7.3) is created by Kai-Wen Liang and Natasa Miskov-Zivanov. This collaboration is reflected in our joint paper [153].

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Systems biology studies systems of biological components, which may be molecules, cells, organisms or entire species. It aims to better understand the properties of individual parts within complex living systems as well as the dynamics of entire systems. To achieve this, given quantitative measurements of the behavior of groups of interacting elements, mathematical and computational models are constructed to reproduce and predict dynamical behaviors. For decades, biologists have been using diagrammatic models to describe and understand the mechanisms and dynamics behind their experimental observations. Although these models are simple to be built and understood, they can only offer a rather static picture of the corresponding biological systems, and scalability is limited. Then, models expressed mathematically (e.g. using differential equations) have occupied the leading position. System biologists simply translate such a model into a computer program simulating that model. As the collaboration between biologists and computer scientists becomes tighter, researchers have realized that biological systems and (distributed) computer systems share a lot of features. That is, similar to (distributed) computer systems, biological systems are consist of various components that communicate with each other and thus influence each other's behavior. This led to an increasing interest for system biologists and computer scientists in borrowing existing formal specification and analysis techniques that were designed for computer systems and in developing biological domain-specific methods, and thus to the success of application of these techniques to biomedical systems.

As shown in Figure 1.1, with formal executable models and well-founded analysis methods for them, it offers an excellent means to present knowledge about biological systems, and to reason about these systems rigorously. Moreover, traditional in-vivo and in-vitro experiments are usually expensive, and need to take an awfully long time. While, the execution of formal models, providing in silico numerical evaluation of hypotheses, only takes comparatively little time and effort. Especially, when considering different experimental configurations, multiple wet-lab experiments need to be carried out repeatedly. Whereas, for formal models, only trivial modifications on the initial assignment of system variables and parameters are required.

In the following part of this chapter, we will first review formal modeling formalisms that have been successfully applied to biological and biomedical systems. We will discuss the main ideas

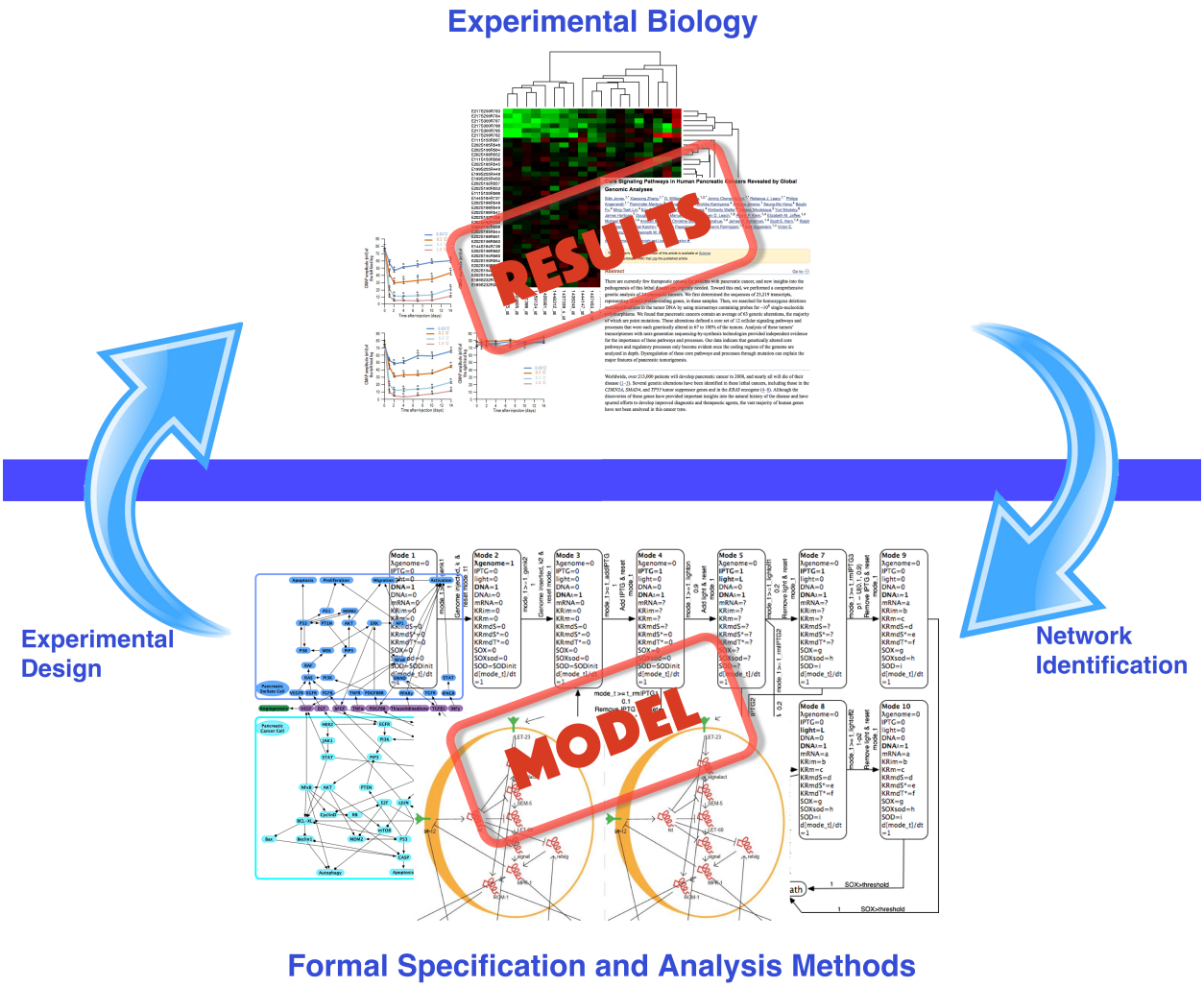


Figure 1.1: Experimental biology itself is an iterative process of hypothesis-driven experimentation of a specific biological system. We can boost this process by using formal executable models, and formal analysis methods, such as model checking, to provide interesting new hypotheses for biological experimental design.

and important features of five specification languages that have fallen on fertile ground in systems biology. (See [25, 88, 155] for reviews of more classes of formal models that have been used in systems biology.) Then, we will briefly go over primary model checking techniques, and discuss its advantages over the mathematical analysis and simulation-based methods.

1.1 Modeling Formalisms for Biological Systems

Boolean Networks

Boolean networks (BNs), as one of the most widely used formal models, were first introduced by Kauffman [142] in 1969, where BNs were used to model gene regulatory networks. A BN is a directed graph containing a set of nodes. Nodes are defined as Boolean variables, whose values represent the dynamic activity and behavior of the involved elements (e.g., genes or proteins). At each time step, the next value of a variable is determined by a Boolean function of its regulators. The values of all variables form a global state to be updated synchronously. In this way, the execution of a BN illuminates the causal and temporal relationships between the involved elements.

The main advantage of this modeling language is that, even with a strongly simplified view of biological networks, it can still capture the network structure and dynamics, and offer biologically meaningful predictions and insights. Also, using such a high abstraction, it is possible to model interactions among large numbers of elements and perform model validation and model-based prediction. Besides being applied to analyze the robustness and stability of genetic regulatory networks [12, 72, 152], BNs have also been used to study cell signaling networks and understand their impacts on distinct cell states [100, 101, 106, 126]. Moreover, BNs can be inferred directly from experimental time-series data [10, 74, 90, 186].

BNs use a coarse approximation where the status of each modeled element as either active (on) or inactive (off) by neglecting intermediate states. In real-word biological systems, some elements may have multiple states. The difference between this binary assumption and biological reality led researchers to suggest extensions of BNs, such as Qualitative Networks (QNs) [190], Gene Regulatory Networks [168], and the logical model considering the time delay mechanism THiMED [165]. In QNs, each variable can have one of a small discrete number of values. Dependencies between variables become algebraic functions instead of Boolean functions. Dynamically, a state of the model corresponds to a valuation of variables and changes in values of variables occur gradually based on these algebraic functions. QNs have been shown to be a suitable formalism to model some biological systems [31, 61, 190]. For the logical model using THiMED, system elements are modeled by multi-valued variables. The timing details that capture relative delays between events are allowed, and implemented by truth tables. Another type of extensions of BNs copes with the inherent noise and the uncertainty in biological processes, such as Boolean networks with noise [11] and Probabilistic Boolean networks [193]. These modeling formalisms allow one to consider the uncertainty in the knowledge of signaling networks as well as stochasticity in biological systems.

Petri Nets

Petri nets [172] were created by Carl Adam Petri in 1962 to describe chemical processes, and then were also intensively employed in computer science to model and analyze concurrent and distributed systems. A Petri net is a graph with two types of nodes - places and transitions, which are connected by directed arcs. Places represent the resources of the system; transitions indicates the events that can change the state of the resources; and directed arcs, connecting places to transitions and transitions to places, describe which places are pre and/or post-conditions for which transitions. The data, in Petri nets, are represented as so-called tokens. The state of the system is represented by places holding tokens. Note that, one place may hold multiple tokens. Given a start configuration of a Petri net, which assigns tokens to each place, transitions change the state of the system by moving tokens along edges. For each transition, tokens are consumed from the input place through the transition and then created in the output place(s). A transition fires whenever it is enabled by the presence of some tokens in one of the places directly connected to it. In a given state of the system, there may be more than one transition that can move a token, so that the execution of a Petri net is non-deterministic.

Petri nets allow for concurrency and nondeterminism, and provide a natural framework in which both qualitative (given by the static structural topology of the Petri nets) and quantitative (given by the time evolution of the token distribution) analysis are tightly integrated. Thus, this modeling language is more general than BNs, and holds a good balance between modeling power and analyzability. Petri nets are well-suited for modeling the concurrent behavior of biochemical networks such as genetic regulatory pathways. In detail, the places in a Petri net can represent genes, protein species and complexes; transitions represent reactions or transfer of a signal; directed arcs represent reaction substrates and products; and a transition firing is execution of a reaction where substrates are consumed and products are created. They have been used to describe the concurrent behavior of biochemical networks, including metabolic pathways and protein synthesis [55, 196].

Another good thing about Petri nets is that, there are several successful extensions forming a very versatile framework providing additional possibilities in modeling and analysis. For instance, in timed Petri nets, transitions can be timed, which allow for modeling the timing of the system as well. They have been used to model and analyze signal transductions in an apoptosis pathway [57]. In colored Petri nets, tokens with different colors denote multiple possible values for each place, and thus allow for distinct activation levels to be assigned to resources. They have been used to analyze metabolic pathways [96]. In stochastic Petri nets, probabilities have been added to the different choices of the transitions to consider the uncertainty of biomedical systems. They have been used to analyze signaling pathways, where the number of molecules of a given type is represented by the color of a place and probabilities represent reaction rates [103, 115].

Rule-based Modeling

The combinatorial explosion, which emerges from the complexity of multi-protein assemblies, poses a major barrier to the development of detailed, mechanistic models of biological systems. Modeling approaches, such as differential equations, that need manually enumerating all potential species and reactions in a network are impractical. To alleviate the problem, rule-based modeling

languages, such as the BioNetGen language (BNGL) [83] and Kappa [71] have been developed. To address the combinatorial complexity in biochemical systems, the key idea of the rule-based languages is to represent interacting molecules as structured objects and to use pattern-based rules to encode their interactions. So that, a rich yet concise description of signaling proteins and their interactions can be provided. In other words, rule-based modeling specifies only those components of a biological macromolecule that are directly involved in a biochemical transformation. Also, the reaction rules are defined as transformations of classes of species, avoiding the need for specifying one reaction per each possible state of a species.

Due to the similarity to the chemical reaction representation widely used in systems biology, rule-based languages have harvested a lot of attention among biologists. It has been applied in the modeling of different cell signaling pathways and networks [26, 27, 35].

Considering that these rule-based languages were designed for describing molecular level dynamics, one growing need is to extend them to span multiple biological levels of organization. ML-Rules [160] is a multi-level rule-based language, which can consider multiple biological levels by allowing objects to be able to contain collections of other objects. This embedding relationship can affect the behavior of both container and contents, and allows users to describe both inter- and intra-cellular processes. Another extension of the BNGL to enable the formal specification of not only the signaling network within a single cell, but also interactions among multiple cells is proposed in [219]. Unlike ML-Rules using continuous rate equations to capture the dynamics of intracellular reactions, this multiscale language models intracellular dynamics using BNs, which reduces the difficulty of estimating the values of hundreds of unknown parameters often involved in large models. This has been used to capture the intra- and inter- cellular dynamics involved in the pancreatic cancer microenvironment [219].

The other increasing need is to take the spatial information into consideration when carrying out the cell biological modeling. SRSim [105], as one spatial extension of the BNGL, integrates the BNGL with a three-dimensional coarse-grained simulation building upon the LAMMPS molecular dynamics simulator [177]. SRSim fills a gap located in between the fine-grained MD simulation models, which do not allow for the formulation of reaction networks, and 2-D or 3-D graph drawing software tools, which do not include any possibility for dynamic simulation. SRSim has been used to model and analyze the human mitotic kinetochore [128]. Another spatial extension of the BNGL is cBNGL [111], in which structures and rules are associated with the concept of compartments and membranes. That is, cBNGL distinguishes between three-dimensional (compartment volume) and two-dimensional (surface) compartments. ML-Space [34], as a spatial variant of ML-Rules, considers compartmental dynamics, mesh-based approaches, and individuals moving in the continuous space. In ML-Space, species can be defined as individual particles that react due to collisions, or as a population of species residing in a small area. It has been used to study the dynamics of lipid rafts and their role in receptor co-localization.

Hybrid Systems

Hybrid systems [14] are formal models that combine continuous and discrete dynamics in a piecewise manner. In detail, the state space of a hybrid system is defined by a finite set of discrete modes. In each mode, the system evolves continuously obeying processes, generally ordinary

differential equations (ODEs) [68]. Transition conditions control the switch from one mode to another, which can be followed by a 'reset' of the involved continuous variables. In general, the temporal dynamics of a hybrid system is piecewise continuous.

By using ODEs, one of the most powerful techniques in modeling system dynamics, hybrid systems aim to bridge the gap between mathematical models and computational models by combining the two. The continuous part of hybrid systems, which are captured by differential equations, bears the closest relationship to the underlying biochemical rate laws, thus can accurately model complex biological systems. While, the discrete part of such models is the executable control mechanism that drives a hybrid system.

Hybrid systems are particularly suitable to model biological systems that exhibit clear switching characteristics over time (that is, the same system variables need to be regulated by different processes in distinct discrete states), such as the cell cycle. They have been successfully used to describe biological systems at distinct levels, including genetic regulatory networks [21], cell signaling pathways [98], the cell cycle control [56], the cardiac cell [228], bacteria-killing procedures [217], and human ventricular action potentials in tissue [49].

Stochastic Hybrid Models

Stochastic hybrid systems (SHSs) are a class of dynamical systems that involve the interaction of discrete, continuous, and stochastic dynamics. Due to that generality, SHSs have been widely used in systems biology, such as modeling subtilin production in bacillus subtilis [125], and personalized prostate cancer treatment [218]. To describe stochastic dynamics, uncertainties have been added to hybrid systems in various ways. A wealth of models has been promoted over the last decade.

One way expresses random initial values and stochastic dynamical coefficients using random variables, resulting in hybrid automata (HAs) with parametric uncertainty [218]. When modeling real-world biological systems using hybrid models, parametric uncertainty arises naturally. Although its cause is multifaceted, two factors are critical. First, probabilistic parameters are needed when the physics controlling the system is known, but some parameters are either not known precisely, are expected to vary because of individual differences, or may change by the end of the system's operational lifetime. Second, system uncertainty may occur when the model is constructed directly from experimental data. Due to imprecise experimental measurements, the values of system parameters may have ranges of variation with some associated likelihood of occurrence.

Another class of models integrates deterministic flows with probabilistic jumps. When state changes forced by continuous dynamics involve discrete random events, we refer to such systems as probabilistic hybrid automata (PHAs) [200]. PHAs extend HAs with discrete probability distributions. More precisely, for discrete transitions in a model, instead of making a purely (non)deterministic choice over the set of currently enabled jumps, a PHA (non)deterministically chooses among the set of recently enabled discrete probability distributions, each of which is defined over a set of transitions. Although randomness only influences the discrete dynamics of the model, PHAs are still very useful and have interesting practical applications [201]. One interesting variation of PHAs [218] allows additional randomness for both transition probabilities and resets of system variables. In other words, in terms of the additional randomness for jump probabilities,

for the probabilities attached to probabilistic jumps from one mode, instead of having a discrete distribution with predefined constant probabilities, they can be expressed by equations involving random variables whose distributions can be either discrete or continuous. This extension is motivated by the fact that some transition probabilities can vary due to factors such as individual and environmental differences in real-world systems. When it comes to the randomness of variable resets, a system variable can be reset to a value obtained according to a known discrete or continuous distribution, instead of being assigned a fixed value. When continuous probabilistic events are also involved, we call them stochastic hybrid automata (SHAs) [92].

Other models replace deterministic flows with stochastic ones, such as stochastic differential equations (SDEs) [18] and stochastic hybrid programs (SHPs) [175], where the random perturbation affects the dynamics continuously. When all such ingredients have been covered, there are models such as the general stochastic hybrid systems (GSHSs) [50, 124]. In the next section, we will show how to construct a stochastic hybrid model of the effect of estrogen at different levels in species' population change in a fresh water ecosystem.

1.2 Model Checking

Model Checking, as a framework consisting of powerful techniques for verifying finite-state systems, was independently developed by Clarke and Emerson [66] and by Queille and Sifakis [180] in the early 1980's. Over the last few decades, it has been successfully applied to numerous theoretical and practical problems [52, 61, 114, 118, 158, 218], such as verification of sequential circuit designs, communication protocols, software device drivers, security algorithms, cyber-physical systems, and biological systems. There are several major factors contributing to its success. Primarily, Model Checking is fully automated. Unlike deductive reasoning using theorem provers, this 'push-button' method neither requires proofs nor experts to check whether a finite-state model satisfies given system specifications. Besides verification of correctness, it permits bug detection as well. If a property does not hold, a model checker can return a diagnostic counterexample denoting an actual execution of the given system model leading to an error state. Such counterexamples can then help detect subtle bugs. Finally, from a practical aspect, Model Checking also works with partial specifications, which allows the separation of system design and development from verification and debugging.

Typically, a model checker has three basic components: a *modeling formalism* adopted to encode a state machine representing the system to be verified, a *specification language* based on Temporal Logics [178], and a *verification algorithm* which employs an exhaustive searching of the entire state space to determine whether the specification holds or not. Because of the exhaustive search, when being applied to complex systems, all model checkers face an unavoidable problem in the worst case. The number of global states of a complex system can be enormous. Given n processes, each having m states, their asynchronous composition may have m^n states which is exponential in both the number of processes and the number of states per process. In Model Checking, we refer to this as the *State Explosion Problem*. Great strides have been made on this

problem over the past 32 years for various types of real-world systems. In the following, we first discuss major breakthroughs that have been made during the development of Model Checking for formal analysis of various types of systems.

Symbolic Model Checking with OBDDs

In the original implementation of the first model checking algorithm [66], the transition system has an explicit representation using the adjacency lists. Such an enumerative representation is feasible for concurrent systems with small numbers of processes and states per process, but not adequate for very large transition systems. In the fall of 1987, McMillan made a fundamental breakthrough. He realized that by reformulating the original model checking procedure in a symbolic way where sets of states and sets of transitions are represented rather than individual states and transitions, Model Checking could be used to verify larger systems with more than 10^{20} states [53]. The new symbolic representation was based on Bryant's ordered binary decision diagrams (OBDDs) [46]. In this symbolic approach, the state graphs, which need to be constructed in the explicit model checking procedure, are described by Boolean formulas represented by OBDDs. Model Checking algorithms can then work directly on these OBDDs. Since OBDD-based algorithms are set-based, they cannot directly implement the depth-first search, and thus the property automaton should also be represented symbolically.

Since then, various refinements of the OBDD-based algorithms [36, 51, 110, 191] have pushed the size of state space count up to more than 10^{120} [51]. The most widely used symbolic model checkers SMV [162], NuSMV [59], and VIS [39] are based on these ideas.

Partial Order Reduction

As mentioned in Section 1, the size of the parallel composition of n processes in a concurrent system may be exponential in n. Verifying a property of such a system requires inspecting all states of the underlying transition system. That is, $n!$ distinct orderings of the interleaved executions of n states need to be considered in the setting where there are no synchronizations between the individual processes. This is even more serious for software verification than for hardware verification, as software tends to be less structured than hardware. One of the most successful techniques for dealing with asynchronous systems is partial order reduction. Since the effect of concurrent actions is often independent of their ordering, this method aims at decreasing the number of possible orderings, and thus reducing the state space of the transition system that needs to be analyzed for checking properties. Intuitively, if executing two events in either order results in the same result, they are independent of each other. In this case, it is possible to avoid exploring certain paths in the state transition system.

Partial order reduction crucially relies on two assumptions. One is that all processes are fully asynchronous. The other is that the property to be checked does not involve the intermediate states. When coping with realistic systems where the processes may communicate and thus depend on one another, this approach attempts to identify path fragments of the full transition system, which only differ in the order of the concurrently executed activities. In this way, the analysis of state space can be restricted to one (or a few) representatives of every possible interleaving.

Godefroid, Peled, and Valmari have developed the concepts of incorporating partial order reduction with Model Checking independently in the early 1990's. Valmari's stubborn sets [209],

Godefroid's persistent sets [99], and Peled's ample sets [171] differ on the actual details but contain many similar ideas. The SPIN model checker, developed by Holzmann [123], uses the ample-set reduction to great advantage.

Bounded Model Checking

Although Symbolic Model Checking (SMC) with OBDDs has successfully improved the scalability and is still widely used, OBDDs have multiple problems which restrict the size of models that can be checked with this method. Since the ordering of variables has to be identical for each path from the root of an OBDD to a leaf node, finding a space-efficient ordering is critical for this technique. Unfortunately, it is quite difficult, sometimes impossible, to find an order resulting in a small OBDD. Consider the formula for the middle output bit of a combinational multiplier for two n-bit numbers. It can be proved that, for all variable orderings, the size of the OBDD for this formula is exponential in n.

To further conquer the state explosion problem, Biere et al. proposed the Bounded Model Checking (BMC) using Boolean satisfiability (SAT) solvers [33]. The basic idea for BMC is quite straightforward. Given a finite-state transition system, a temporal logic property and a bound k (we assume $k \geq 1$), BMC generates a propositional logical formula whose satisfiability implies the existence of a counterexample of length k, and then passes this formula to a SAT solver. This formula encodes the constraints on initial states, the transition relations for k steps, and the negation of the given property. When the formula is unsatisfiable (no counterexample found), we can either increase the bound k until either a counterexample is found, or k reaches the upper bound on how much the transition relation would need to be unwound for the completeness, or stop if resource constraints are exceeded. As an industrial-strength model checking technique, BMC has been observed to surpass SMC with OBDDs in fast detection of counterexamples of minimal length, in saving memory, and by avoiding performing costly dynamic reordering. With a fast SAT solver, BMC can handle designs that are order-of-magnitude larger than those handled by OBDD-based model checkers.

As an efficient way of detecting subtle counterexamples, BMC is quite useful in debugging. In order to prove correctness when no counterexamples are found using BMC, an upper bound on steps to reach all reachable states needs to be determined. It has been shown that the diameter (i.e., the longest shortest path between any two states) of the state-transition system could be used as an upper bound [33]. But, it appears to be computationally difficult to compute the diameter when the state-transition system is given implicitly. Other ways for making BMC complete are based on induction [192], cube enlargement [161], Craig interpolants [163], and circuit co-factoring [93]. This problem remains a topic of active research.

An interesting variation of the original BMC is to adopt a Satisfiability Modulo Theories (SMT) solver instead of a SAT solver [70, 205]. SMT encodings in model checking have several advantages. The SMT encodings offers more powerful specification language. They use (unquantified) first-order formulas instead of Boolean formulas, and use more natural and compact encodings, as there is no need to convert high level constraints into Boolean logic formulas. These SMT encodings also make the BMC work the same for finite and infinite state systems. Above all, high level of automation has not been sacrificed for the above advantages. CBMC is a widely used Bounded

model checker for ANSI-C and C++ programs [149] , having supports for SMT solvers such as Z3 [73], and Yices [79].

Counterexample-Guided Abstraction Refinement

When the model state space is enormous, or even infinite, it is infeasible to conduct an exhaustive search of the entire space. Another method of coping with the state explosion problem is to abstract away irrelevant details, according to the property under consideration, from the concrete state transition system when constructing the model. We call this approach *abstraction*. This simplification incurs information loss. Depending on the method used to control the information loss, abstraction techniques can be distinguished into either over-approximation or under-approximation techniques. The over-approximation methods enrich the behavior of the system by releasing constraints. They establish a relationship between the abstract model and the original system so that the correctness of the former implies the correctness of the latter. The downside is that they admit false negatives, where there are properties which hold in the original system but fail in the abstract model. Therefore, a counterexample found in the abstract system may not be a feasible execution in the original system. These counterexamples are called *spurious*. Conversely, the under-approximation techniques, which admit false positives, obtain the abstraction by removing irrelevant behavior from the system so that a specification violation at the abstract level implies a violation of the original system.

The *counterexample-guided abstraction refinement* (CEGAR) technique [63] integrates an over-approximation technique - existential abstraction [67] - and SMC into a unified, and automatic framework. It starts verification against universal properties with an imprecise abstraction, and iteratively refines it according to the returned spurious counterexamples. When a counterexample is found, its feasibility with regard to the original system needs to be checked first. If the violation is feasible, this counterexample is reported as a witness for a bug. Otherwise, a proof of infeasibility is used to refine the abstraction. The procedure then repeats these steps until either a real counterexample is reported, or there is no new counterexamples returned. When the property holds on the abstract model, by the Property Preservation Theorem [67], it is guaranteed for the property to be correct in the concrete systems. CEGAR is used in many software model checkers including the SLAM project [22] at Microsoft.

Model Checking for Stochastic Hybrid Systems

The popularity of SHSs in real-world applications plays an important role as the motivation for putting a significant research effort into the foundations, analysis and control methods for this class of systems. Among various problems, one of the elementary questions for the quantitative analysis of SHSs is the probabilistic reachability problem. There are two main reasons why it catches researchers' attention. Primarily, it is motivated by the fact that most temporal properties can be reduced to reachability problems due to the very expressive hybrid modeling framework. Moreover, probabilistic state reachability is a hard and challenging problem which is undecidable in general. Intuitively, this class of problems is to compute the probability of reaching a certain set of states. The set may represent a set of certain unsafe states which should be avoided or visited only with some small probability, or dually, a set of good states which should be visited frequently.

Over the last decade, research efforts concerning SHSs are rapidly increasing. At the same

time, Model Checking methods and tools for probabilistic systems, such as PRISM [150], MRMC [141], and Ymer [230], have been proposed and designed. Results related to the analysis and verification of SHSs are still limited. For instance, analysis approaches for GSHSs are often based on Monte-Carlo simulation [37, 182]. Considering the hardness of dealing with the general class, efforts have been mainly placed on different subclasses [7, 8, 9, 91, 92, 108, 175, 200, 218, 232, 234].

For a decidable subclass which is called probabilistic initialized rectangular automata (PIRAs), Sproston offered a model checking procedure against the probabilistic branching time logic (PBTL) [200]. The procedure first translates PIRA to a probabilistic timed automaton (PTA), then constructs a finite-state probabilistic region graph for the PTA, and employs existing PBTL Model Checking techniques. For probabilistic rectangular automata (PRAs) which are less restricted than PIRAs, Sproston proposed a semi-decidable model checking procedure via using a forward search through the reachable state space [201].

For a more expressive class of models - probabilistic hybrid automata (PHAs), Zhang et al. abstracted the original PHA to a probabilistic automaton (PA), and then used the established Model Checking methods for the abstracting model [232]. Hahn et al. also discussed an abstraction-based method where the given PHA was translated into a n-player stochastic game using two different abstraction techniques [108]. All abstractions obtained by these methods are over-approximations, which means that the estimated maximum probability for a safety property on the abstracted model is no less than the one on the original model. Another method proposed is a SMT-based bounded Model Checking procedure [91].

A similar class of models, which is widely used in the control theory, is called discrete-time stochastic hybrid systems (DTSHSs) [15]. Akin to PHAs, DTSHSs comprise nondeterministic as well as discrete probabilistic choices of state transitions. Unlike PHAs, DTSHSs are sampled at discrete time points, use control inputs to model nondeterminism, do not have an explicit notion of symbolic transition guards, and support a more general concept of randomness which can describe discretized stochastic differential equations. With regard to the system analysis, the control problem concerned can be understood as to find an optimal control policy that minimizes the probability of reaching unsafe states. A backward recursive procedure, which uses dynamic programming, was then proposed to solve the problem [7, 15]. Another approach to a very similar problem as above, where a DTSHS model doesn't have nondeterministic control inputs, was presented in [8]. Compared to former method, the latter approach exploits the grid to construct a discrete-time Markov chain (DTMC), and then employs standard model checking procedures for it. This approach then had been used in [9] as an analysis procedure for the probabilistic reachability problems in the product of a DTSHS and a Büchi automaton representing a linear temporal property. Zuliani et al. also mentioned a simulation-based method for model checking DTSHSs against bounded temporal properties [234]. We refer to this method as Statistical Model Checking (StatMC). The main idea of StatMC is to generate enough simulations of the system, record the checking result returned from a trace checker from each simulation, and then use statistical testing and estimation methods to determine, with a predefined degree of confidence, whether the system satisfies the property. Although this statistical model checking procedure does not belong

to the class of exhaustive state-space exploration methods, it usually returns results faster than the exhaustive search with a predefined arbitrarily small error bound on the estimated probability.

In [92], as an extension of PHAs, stochastic hybrid automata (SHAs) allow continuous probability distributions in the discrete state transitions. With respect to the verification procedure, a given SHA is firstly over-approximated by a PHA via discretizing continuous distributions into discrete ones with the help of additional uncountable nondeterminism. As mentioned, this over-approximation preserves safety properties. For the second step, the verification procedure introduced in [232] is exploited to model check the over-approximating PHA.

Another interesting work is about stochastic hybrid programs (SHPs) introduced in [175]. This formalism is quite expressive with regard to randomness: it takes stochastic differential equations, discrete probabilistic branching, and random assignments to real-valued variables into account. To specify system properties, Platzer proposed a logic called stochastic differential dynamic logic, and then suggested a proof calculus to verify logical properties of SHPs.

1.3 Overview of Contributions

In this thesis, we have been focusing on designing appropriate modeling formalisms and efficient analyzing algorithms for various biological systems in three different thrusts:

- **Modeling Formalisms**: We design a multi-scale hybrid rule-based modeling formalism, extended from the traditional rule-based language - BioNetGen. This new language is able to describe the intracellular reactions and intercellular interactions simultaneously. Furthermore, to depict intracellular reactions, its hybrid characteristic asks for less information about model parameters, such as reaction rates, than traditional rule-based languages. In a nutshell, our language can describe both discrete and continuous models using a unified rule-based representation. This results in a modeling framework that combines the advantages of logic and kinetic modeling approaches. Details can be found in Chapter 6.

- **Formal Analysis Algorithms**:

  - We develop a model checking algorithm for Qualitative Networks (QNs), a formalism for modeling signal transduction networks in biology. One of the unique features of qualitative networks, due to their lacking initial states, is that of "reducing reachability sets". Our method considers this unique features of QNs and combines it with over-approximation to compute decreasing sequences of reachability set for QN models, which results in a more scalable model checking algorithm for QNs. Details can be found in Chapter 3.

  - We propose a formal analyzing method to handle probabilistic bounded reachability problems for two kinds of stochastic hybrid systems - general hybrid systems with parametric uncertainty and probabilistic hybrid automata with additional randomness.

Standard approaches to reachability problems for linear hybrid systems require numerical solutions for large optimization problems, and become infeasible for systems involving both nonlinear dynamics over the reals and stochasticity. Our approach combines a SMT-based model checking technique with statistical tests in a sound manner. Compared to standard simulation-based methods, it supports non-deterministic branching, increases the coverage of simulation, and avoids the zero-crossing problem. Details can be found in Chapter 5.

- We design a framework, where formal methods and machine learning techniques take joint efforts to automate the model construction, analysis, and refinement of biological and biomedical systems. The creation of models most often relies on intense human effort. That is, model developers have to read hundreds of published papers and conduct numerous discussions with experts to understand the behavior of the system and to construct the model. This laborious process results in slow development of models, let alone validating the model and extending it with thousands of other possible components that already exist in published literature. Meanwhile, research results are published at a high rate, and the published literature is voluminous, but often fragmented, and sometimes even inconsistent. Our framework offers the automation of information extraction from literature, smart assembly into models, and model analysis, to enable researchers to re-use and reason about previously published work, in a comprehensive and timely manner. Details can be found in Chapter 7.

- **Modeling and Applications:**

  - To investigate whether and how formal modeling and analysis methods can contribute to the study of biological systems, we construct a Boolean Network model for the signaling network for a single pancreatic cancer cell. Important system dynamics with respect to cell fate, cell cycle, and oscillating behaviors are formulated into CTL formulas. Then, we used an existing symbolic model checker NuSMV to check against these CTL properties, and confirmed experimental observations and thus validated our model. Details can be found in Chapter 2.

  - To show the speedup offered by our improved bounded LTL model checking technique for QNs, we build several QN models describing the cellular interactions during the development of the skin differentiation. By comparing our method with an existing model checking technique for QNs, we showed that our method offered a significant acceleration especially when analyzing large and complex models. Details can be found in Chapter 3.

  - We create a nonlinear hybrid model to depict a light-aided bacteria-killing process. Then, by using a recently promoted δ-complete decision procedure-based model checking technique, we found that 1) the earlier we turn on the light after adding IPTG, the quicker bacteria cells can be killed; 2) in order to kill bacteria cells, the light has to be turned on for at least 4 time units; 3) the time difference between removing the light and removing IPTG has insignificant impact on the cell killing outcome; and 4) the

range of the necessary concentration of SOX to kill bacteria cells can be broader than the range indicated by our collaborating biologist. Details can be found in Chapter 4.

- We extend hybrid models for atrial fibrillation, prostate cancer treatment, and our bacteria-killing process into stochastic hybrid models. We, then, apply our probabilistic bounded reachability analyzer SReach to demonstrate its feasibility in model falsification, parameter estimation, and sensitivity analysis. We also use SReach to perform the bounded-time reachability analysis on the Tap Withdrawal circuit model of C. elegans to estimate the probability of various TW responses related to parameter uncertainty, and thus to derive population percentages that exhibit various behaviors in response to tap stimuli. It shows that SReach can handle large-scale systems for which traditional reachability analysis may not scale. Details can be found in Chapter 5.4.
- To study the mechanism underlying the tumor micro-environment during the development of the pancreatic cancer, we develop a multi-scale hybrid rule-based model for the pancreatic cancer micro-environment, and employ statistical model checking to analyze it. The formal analysis results showed that our model could reproduce existing experimental findings with regard to the mutual promotion between pancreatic cancer and stellate cells. The results also explained how treatments latching onto different targets resulted in distinct outcomes. We then used our model to predict possible targets for drug discovery. Details can be found in Chapter 6.

Chapter 2

Pancreatic Cancer Single Cell Model as Boolean Network and Symbolic Model Checking

Signal transduction is a process for cellular communication where the cell receives (and responds to) external stimuli from other cells and from the environment. It affects most of the basic cell control mechanisms such as differentiation and apoptosis. The transduction process begins with the binding of an extracellular signaling molecule to a cell-surface receptor. The signal is then propagated and amplified inside the cell through signaling cascades that involve a series of trigger reactions such as protein phosphorylation. The output of these cascades is connected to gene regulation in order to control cell function. Signal transduction pathways are able to crosstalk, forming complex signaling networks.

In this chapter, we have investigated the functionality of six signaling pathways that have been shown to be 100% genetically mutated during the progression of pancreatic cancer [133], within a pancreatic cancer cell, and constructed a in-silico Boolean network model considering the crosstalk among them [100, 101].

Pancreatic cancer (PC) is a highly aggressive malignancy and the 4th leading cause of cancer-related death in the United States [5]. It arises from intraepithelial neoplasia (PanIN), a progression of lesions that occur in the pancreatic ducts. It is characterized by a propensity for early local and distant invasion - rapid growth, early metastasis - and an unresponsiveness to most conventional treatments - it is highly resistant to chemotherapy and radiation. Vogelstein et al. [133] global genomic analysis identified 12 cellular signaling pathways that are genetically altered in over 67% of pancreatic cancers. The study also found that PC contains an average of 63 genetic alterations, and that the KRAS, apoptosis, TGFβ, Hedgehog, and Wnt/Notch signaling pathways, and the regulation of G1/S phase transition have genetic alterations in 100% of tumors. A number of molecular and pathological analyses of evolving pancreatic adenocarcinoma revealed progressive genetic mutations of KRAS, CDKN2A, TP53, SMAD4, corresponding to the mutations of the KRAS, INK4a, ARF, P53, and SMAD4 proteins in the above mentioned pathways. Mutations of

oncoprotein and tumor suppressor proteins result in uncontrolled cell proliferation and evasion of apoptosis (programmed cell death), eventually leading to cancer. In addition, PC over-expresses a number of growth factors (GF) and their respective receptors, including the epidermal growth factor (EGF), sonic hedgehog (SHH), WNT, transforming growth factor (TGFβ), and Insulin-like growth factor (IGF1) or Insulin. These growth factors can stimulate pancreatic cancer cell growth via autocrine and/or paracrine feedback loops. In our model, we have considered three important cell functions - proliferation, apoptosis, and cell cycle arrest. Given this model, we are interested in verifying that sequences of signal activation will drive the network to a pre-specified state within a pre-specified time. Thus, we have applied symbolic model checking (SMC) to it, and shown that its behaviors are qualitatively consistent with experiments. We have demonstrated that SMC offers a powerful approach for studying logical models of relevant biological processes.

2.1 Pancreatic Cancer Cell Model

Genomic analyses [133] have identified six cellular signaling pathways that are genetically altered in 100% of pancreatic cancers: the KRAS, Hedgehog, Wnt/Notch, Apoptosis, TGFβ, and regulation of G1/S phase transition signaling pathways. Also, many in vitro and in vivo experiments with pancreatic cancer cells have found that several growth factors and cytokines including IGF/Insulin, EGF, Hedgehog, WNT, Notch ligands, HMGB1, TGFβ, and oncoprotein including RAS, NFκB, and SMAD7 are overexpressed [19]. We performed an extensive literature search and constructed a signaling network model composed by the EGF-PI3K-P53, Insulin/IGF-KRAS-ERK, SHH-GLI, HMGB1-NFκB, RB - E2F, WNTβ - Catenin, Notch, TGFβ - SMAD, and Apoptosis pathway. Our aim is to study the interplay between tumor growth, cell cycle arrest, and apoptosis in the pancreatic cancer cell. In Figure 2.1, we depict the crosstalk model of different signaling pathways in the pancreatic cancer cell. Here, we will first iterate these pathways, and focus on their association with apoptosis, cell cycle arrest and tumor proliferation. In the following, the symbol $\rightarrow$ means activation (or overexpression), while the symbol $\dashv$ denotes inhibition (or deactivation).

Insulin/IGF-KRAS-ERK pathway
Insulin/IGF $\rightarrow$ IR $\rightarrow$ KRAS $\rightarrow$ RAF $\rightarrow$ MEK $\rightarrow$ ERK $\rightarrow$ AP1,MYC. The overexpressed growth factors, including Insulin-like growth factor (IGF) and Insulin, could activate the KRAS protein, resulting in the phosphorylation of its downstream proteins RAF, MEK, and ERK [75]. These can phosphorylate or activate the transcription factors AP1 and MYC to activate the expression of the cell cycle regulatory protein Cyclin D, enabling progression of the cell cycle through the G1 phase. KRAS is mutated in over 90% of pancreatic cancers [23]. This pathway could also upregulate the expression level of GLI in the sonic hedgehog pathway [133].

EGF-PI3K-P53 pathway
There are two important downstreaming pathways: PI3K $\rightarrow$ PIP3 $\rightarrow$ AKT $\rightarrow$ MDM2 $\dashv$ P53 $\rightarrow$ P21,BAX, and P53 $\rightarrow$ PTEN $\dashv$ PIP3 $\rightarrow$ AKT $\rightarrow$ MDM2 $\dashv$ P53. PC overexpresses a number of mitogenic growth factors and receptor tyrosine kinase (RTK), including EGF(R), IGF(R), which can activate the PI3K pathway to promote the growth of pancreatic cancer cells. The activation

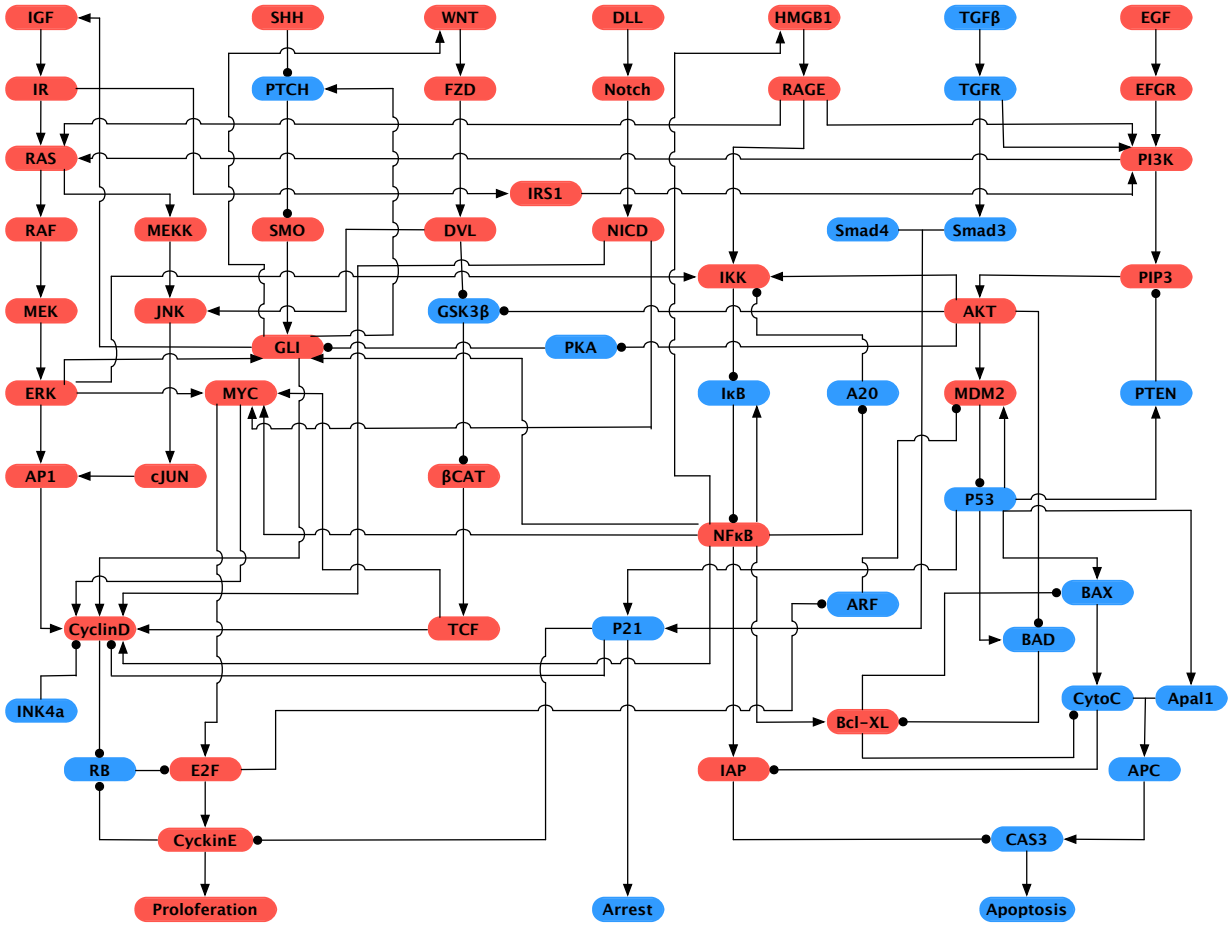


Figure 2.1: Schematic view of signal transduction in the pancreatic cancer model. Blue nodes represent tumor-suppressor proteins, red nodes represent oncoproteins/lipids. Arrow represents protein activation, circle-headed arrow represents deactivation.

of PI3K initiates a cascade of reactions including the phosphorylation of PIP2, AKT and MDM2, leading to the inhibition of P53s transcription activity in the nucleus [113]. The tumor-suppressor protein P53, expressed in the later stage of PanIN, is mutated in more than 50% of pancreatic adenocarcinomas [23]. Also, P53 is a transcription factor for many tumor-suppressor proteins including PTEN and P21, which can negatively regulate the AKT pathway, and induce cell cycle arrest, respectively.

RB-E2F pathway

CyclinD $\dashv$ RB $\dashv$ E2F $\rightarrow$ CyclinE. This pathway regulates the cell cycle progression from phase G1 to phase S, induced by the Cyclin E and CDK2 complex. In the normal cell, the unphosphorylated RB, a tumor suppressor protein, binds to E2F and inhibits its transcription activity. E2F will be activated when its inhibitor RB is phosphorylated and inhibited by CyclinD, promoting the transcription of CyclinE [227]. The germline mutations of CDKN2A in this pathway, which encodes the tumor suppressors INK4a (inhibitor of CyclinD-CDK4/6) and ARF (inhibit MDM2s activity to stabilize P53), were found in up to 90% of pancreatic cancers [23].

SHH-GLI pathway

It is composed of two main parts: 1) SHH $\dashv$ PTCH $\dashv$ SMO $\rightarrow$ GLI $\rightarrow$ IGF,WNT,CyclinD,PTCH; and 2) AKT $\dashv$ PKA $\dashv$ GLI. The Sonic hedgehog (SHH) protein and its receptor Smoothened (SMO) are activated and overexpressed in later-stage pancreatic carcinomas, and it occurs in over 70% of PCs [203]. In the quiescent cell without SHH, SMO is bound and inhibited by the tumor suppressor protein patched (PTCH). Once SHH binds to PTCH, SMO is released to activate the glioma-associated oncogene homologue (GLI1/2/3), leading to an active form of transcription factor. In the absence of SHH, the protein kinase A (PKA) and CKI (only PKA is shown in Fig. 2.1) transform GLI into a repressor form which can inhibit GLIs transcriptional activity. The activation of the SHH-GLI pathway is associated with tumor proliferation and pancreatic cancer-associated fibroblasts [216]. The expression of GLI could also be up-regulated by the PI3K-AKT and KRAS-ERK pathways, independently from SHH activation. In particular, SHH signaling alone is sufficient to drive pancreatic neoplasia, but does not form pancreatic adenocarcinomas [203].

WNT pathway

WNT $\rightarrow$ FZD $\rightarrow$ DVL $\dashv$ GSK3β $\dashv$ β-Catenin $\rightarrow$ TCF $\rightarrow$ CyclinD. WNT pathway activation and the overexpression of several pathway components were observed in 65% of pancreatic adenocarcinomas [231]. When the WNT protein is absent, β-catenin is localized in the cytoplasm, bound to and inhibited by the complexes composed of Axin, APC, and GSK3β [225]. The canonical WNT pathway is activated by the interaction of WNT and Frizzled (FZD) proteins, which can destabilize the Axin-APC-GSK3 complex and translocate β-catenin to the nucleus, where it activates the TCF-LEF transcription factors [212].

Notch pathway

DLL $\rightarrow$ Notch $\rightarrow$ NICD $\rightarrow$ CyclinD. The Notch pathway is activated after binding of transmembrane ligands, including DLL (Delta-like 1, 3, 4) and Jagged 1- 2 with Notch proteins. After that, Notch will be cleaved and a Notch intracellular domain (NICD) will be released, which will translocate to the nucleus to induce the expression of several target genes, including the cell regulatory protein CyclinD. Recent findings indicate that the Notch pathway is involved in the devel-

opment of pancreatic cancer [48].

HMGB1-NFκB pathway
signaling $\rightarrow$ IKK $\dashv$ IκB $\dashv$ NFκB $\rightarrow$ A20,IκB,BclXL,GLI. A recent study [136] has found that the overexpression of HMGB1 could promote the growth of pancreatic cancer cells by activating the RAGE pathway. In the resting cell, NFκB is located in the cytoplasm, bound to and inhibited by IκB. Once activated by HMGB1, the IκB kinase (IKK) will phosphorylate and deactivate Iκ B, leading to the translocation of NFκB into the nucleus to promote the transcription of a number of genes, including CyclinD, the anti-apoptotic protein Bcl-XL, its inhibitors A20 and IκB [122, 210], and HMGB1 [137].

TGFβ-SMAD pathway
it has two main parts: 1) TGFβ $\rightarrow$ TGFR $\rightarrow$ SMAD2/3/4 $\rightarrow$ P21; and 2) TGFβ $\rightarrow$ TGFR $\rightarrow$ PI3K-RAS-pathway. The TGFβ-SMAD signaling pathway can inhibit the growth of normal human epithelial cells. When the TGFβ ligand binds to type II TGFβ recep tors (TGFR), Type I receptors will be activated, leading to the phosphorylation of the cytoplasmic SMAD2/3 proteins. The proteins SMAD2/3 form a complex with SMAD4, and translocate into the nucleus to activate several transcription factors, upregulating the expression of cyclin-dependent kinase (CDK) inhibitors, including P21 [76, 80]. SMAD4 was found to be either mutated or deleted in over 50% of pancreatic cancers which occurred in the later-stage PanINs [131]. In addition to the Smad-dependent signaling pathway, TGF also activates the PI3K-RAS pathway, leading to the crosstalk with the WNT and EGF pathways. Impairment of the TGFβ-SMAD pathway promotes cell proliferation and contributes to carcinogenesis.

Apoptosis pathway
P53 $\rightarrow$ BAD, BAX , Apaf1 $\rightarrow$ cytochrome-C $\rightarrow$ Cas3. The apoptosis pathway is regulated by both the anti-apoptotic (BclX) and the pro-apoptotic Bcl-2 families of proteins [187]. The activation of P53 will induce or upregulate the transcription of several pro-apoptotic pro- teins including BAX, BAD, and Apaf1 (Apoptotic protease activating factor 1). After receiving pro-apoptotic signals from P53, BAD will inhibit Bcl-XLs pro-apoptotic effects, while this process is inhibited by the pro-survival signals from AKT. BAX is a protein of the Bcl-2 family which can activate the apoptosis process by promoting the release of cytochrome C (Cyto-C) from the mitochondrion. This, in turn, promotes the formation of the apoptosome complex (APC) [183] which contains Cyto-C and Apaf1. Cas3 is an apoptosis effector caspase (cysteine-dependent aspartate specific proteases) which can cleave proteins in the execution phase of cell apoptosis [199]. The activation of Cas3 is promoted by APC and inhibited by the inhibitors of apoptosis (IAP). It has been found that Cas3 is mutated in many cancer types [199].

2.2 Boolean Network

In this Section, we translate the above signaling pathways into a Boolean network model. The input signals of the model are different growth factors including SHH, EGF, TGF. The output signals are Apoptosis, (Cell) Proliferation, and (Cell Cycle) Arrest.

In the Boolean network model of the pancreatic cancer cell, each node represents a protein/lipid in the signaling pathway. At any specific time, each node can be in either the ON(1) or OFF(0) state. The state evolution of a node from time t to $t+1$ is described by a Boolean transfer function. This function will in general depend on the state of the neighbor nodes. In this paper we use several forms of transfer function. In one form, we assume that a node is activated (inhibited) if its incoming neighbor is active (inhibited). This form is used, for example, for receptor nodes such as EGFR, which are expressed only if their upstream ligand is present. A dual form assumes that a node is activated (inhibited) when its incoming neighbor is inhibited (activated). This form is used, e.g., for SMO, which is bound and inhibited by PTCH (see the description of the SHH-GLI pathway above).

In another form, we assume that neighboring nodes are classified as activators or inhibitors. Activators node can change the state of a node n if and only if no inhibitor acting on node n is in the ON state. Our assumption is motivated by the fact that many tumor-suppressor proteins including P53, PTEN, SMAD4, INK4a, and ARF, are either lost or mutated in the early or late stages of PDAC, while oncoproteins such as KRAS, NFκB, and GLI, are continuously activated or overexpressed. The transfer function for node n can be written as

$$n(t+1) = \{n(t) \vee \bigvee_{a \in A(n)} a(t)\} \wedge \neg(\bigvee_{i \in I(n)} i(t)), \tag{2.1}$$

where $A(n)$ and $I(n)$ are the activators and inhibitors of node n, respectively.

In our model, we assume synchronous state update for all the nodes in the network. That is, at any time step the state of each node in the model is updated according to its transfer function. In the future we plan to study asynchronous models, to take into account the observation that biological processes may evolve at different speeds. We remark that our verification approach would still work, since Model Checking can cope with asynchronous systems.

The Boolean network in Figure 2.1 comprises 61 nodes, including 7 control (input) nodes, and 3 output nodes. We emphasize that the structure depicted in Figure 2.1 is not a state transition graph. Rather, it represents the wiring diagram of our model. Since each node is a Boolean variable, the state space of the model has cardinality 2^{61}. Is it a correct model to describe the proliferation and apoptosis of pancreatic cancer cell? To answer this question we use Symbolic Model Checking of Computational Tree Logic (CTL) properties, which we will introduce next.

2.3 Symbolic Model Checking

Given our pancreatic single cell model, we express its intended behavior as Computation Tree Logic (CTL) [65] formulas. Then, we apply symbolic model checking against it. Here, we give a brief introduction to CTL and symbolic model checking.

Kripke structure

A finite state system can be described as a tuple:

$$\mathcal{M} =< \mathcal{S}, \mathcal{I}, \mathcal{R}, \mathcal{L} >$$

where $\mathcal{S}$ is a finite set of states, $\mathcal{I} \subseteq \mathcal{S}$ is the set of initial states, and $\mathcal{R} \subseteq \mathcal{S} \times \mathcal{S}$ is the transition relation, specifying the possible transitions from state to state. $\mathcal{L}$ is a function that labels states with the atomic propositions from a given language. Such a tuple is called state transition system or Kripke structure [148].

Computational Tree Logic

Temporal logics are used to predicate over the behavior defined by Kripke structures. A behavior in a Kripke structure is obtained starting from a state $s \in \mathcal{I}$, and then repeatedly appending states reachable through $\mathcal{R}$. We require that the transition relation $\mathcal{R}$ be total (that is, a transition relation $\mathcal{R} \subseteq S \times S$ is total if and only if for each state $s \in \mathcal{S}$ there exists a state $s' \in \mathcal{S}$ such that $(s, s') \in \mathcal{R}$). As a consequence all the behaviors of the system are infinite. Since a state can have more than one successor, the structure can be thought of as unwinding into an infinite tree, representing all the possible executions of the system starting from the initial states.

Two useful temporal logics are Computation Tree Logic (CTL) and Linear Temporal Logic (LTL). They differ in how they handle branching in the underlying computation tree. In CTL temporal operators it is possible to quantify over the paths departing from a given state. In LTL operators are intended to describe properties of all possible computation paths. For our pancreatic cancer cell model, we use CTL to describe the corresponding system properties.

The syntax of CTL formulas is given by the following rules:

- any atomic proposition is a CTL formula;
- if α and β are CTL formulas, then $\alpha \bullet \beta$ and $\neg\alpha$ are CTL formulas, where $\bullet$ is any boolean connective ($\wedge, \vee, \ldots$); and
- if α and β are CTL formulas, then $\mathbf{EX}\alpha$, $\mathbf{EG}\alpha$, $\mathbf{E}[\alpha \mathbf{U} \beta]$ are CTL formulas.

The intuitive meaning of CTL formulas:

- $\mathbf{EX}\alpha$ means that there exists (E) a path starting from a state $s_0 \in \mathcal{S}$ in which in the next (X) state α holds.
- $\mathbf{EG}\alpha$ means that there exists a path starting from a state s_0 in which globally (G) α holds.
- $\mathbf{E}[\alpha \mathbf{U} \beta]$ there exists a path starting from a state s_0 in which α holds until (U) β holds.

The other CTL operators (e.g., $\mathbf{AF}\alpha$, meaning for all paths eventually α) can be derived from these three as follows.

- $\mathbf{AX}\alpha = \neg\mathbf{EX}(\neg\alpha)$ means that, for all paths, in the next state α.

- $\mathbf{EF}\alpha = \mathbf{E}[\top \mathbf{U} \alpha]$ means that there exists a path in which eventually α.
- $\mathbf{AG}\alpha = \neg\mathbf{EF}(\neg\alpha)$ means invariantly α.
- $\mathbf{A}[\alpha \mathbf{U} \beta] = \neg\mathbf{E}[\neg\beta \mathbf{U} \neg\alpha \wedge \neg\beta] \wedge \neg\mathbf{EG}\neg\beta$ means that, for all paths, α until β.
- $\mathbf{AF}\alpha = \mathbf{A}[\top \mathbf{U} \alpha]$ means that, for all paths, eventually α.

Symbolic CTL Model Checking

The Model Checking algorithm applied to a CTL formula ϕ works by recursively labeling the state graph with the sub-formulas of ϕ, and then parses the graph to compute, for each sub-formula, its truth value in a state according to the CTL operators and the truth values of its subformulas. In the original Model Checking algorithm, the state transitions were represented explicitly: this can lead to state explosion. The main idea behind symbolic model checking is to represent and manipulate a finite state-transition system symbolically as a Boolean function, so as to alleviate the state explosion problem. In particular, Ordered binary decision diagrams (OBDDs) [46] are a canonical form for Boolean formulas. OBDDs are often substantially more compact than traditional normal forms. Moreover, they can be manipulated very efficiently.

We consider Boolean formulas over n variables $x_1, \cdots, x_n$. A binary decision diagram (BDD) is a rooted directed acyclic graph with two types of vertices, terminal vertices and nonterminal vertices. Each nonterminal vertex v is labeled by a variable $var(v)$ and has two successors, $low(v)$ and $high(v)$. Each terminal vertex v is labeled by either 0 or 1 via a Boolean function $value(v)$. A BDD with root v determines a Boolean function $f_v(x_1, \cdots, x_n)$ in the following manner.

- If v is a terminal vertex then $f_v(x_1, \cdots, x_n) = value(v)$.
- If v is a nonterminal vertex with $var(v) = x_i$ then $f_v(x_1, \cdots, x_n)$ is given by

$$(\neg x_i \wedge f_{low(v)}(x_1, \cdots x_n)) \vee (x_i \wedge f_{high(v)}(x_1, \cdots x_n))$$

In an OBDD there is a strict total ordering of the variables $x_1, \cdots, x_n$ when traversing the diagram from the root to the terminals. Given an assignment to the variables $x_1, \cdots, x_n$, the value of the formula can be decided by traversing the OBDD from the root to the terminals. At each node, branching is decided by the value assigned to the variable that labels the node.

There exist efficient algorithms for operating on OBDDs. All sixteen two-argument logical operations can be implemented efficiently on Boolean functions that are represented as OBDDs. In particular, the complexity of these operations is linear in the product of the size of the argument OBDDs. The key idea for efficient implementation of these operations is Shannon expansion: $f = (\neg x \wedge f|_{x=0}) \vee (x \wedge f|_{x=1})$. In [46], Bryant gave a uniform procedure for computing all 16 logical operations.

McMillan developed the symbolic CTL model checking algorithm using BDDs [162]. This algorithm can handle much larger concurrent systems than the explicit-state model checking [54].

State transition systems can be represented with BDDs as follows. First, we must represent the states in terms of n Boolean state variables $v = \{v_1, v_2, \cdots, v_n\}$. Then, we express the transition relation R as a Boolean formula in terms of the state variables:

$$f_R(v_1, v_2, \cdots, v_n, v_1', v_2', \cdots, v_n') = 1 \quad iff \quad R(v_1, v_2, \cdots, v_n, v_1', v_2', \cdots, v_n')$$

where $v_1, v_2, \cdots, v_n$ represent the current state and $v_1', v_2', \cdots, v_n'$ represent the next state. Finally, we convert f_R to a BDD.

2.4 Results and Discussion

We used NuSMV [60], a Symbolic Model Checker to determine whether our in silico pancreatic cancer cell model satisfies certain properties written in a temporal logic. In our model, we set the initial values of ARF, INK4α, and SMAD4 to be OFF (0), while Cyclin D is set to be ON (1). These choices are motivated by the following observations. According to the genetic progression model of pancreatic adenocarcinoma, the malignant transformation from normal duct to pancreatic adenocarcinomas requires multiple genetic alterations in the progression of neoplastic growth, represented by Pancreatic intraepithelial neoplasia (PanINs)1A/B, PanIN-2, PanIN-3 [23]. The loss of the functions of CDKN2A, which encodes two tumor suppressors INK4A and ARF, occurs in 80 - 95% of sporadic pancreatic adenocarcinomas [185]. SMAD4 is a key component in the TGFβ pathway which can inhibit most normal epithelial cellular growth by blocking the G1-S phase transition in the cell cycle; and it is frequently lost or mutated in pancreatic adenocarcinoma [224]. Furthermore, it has been shown that the loss of SMAD4 can predict decreased survival in pancreatic adenocarcinoma [116]. Besides the loss of many tumor suppressors, the oncoprotein Cyclin D is frequently overexpressed in many human pancreatic endocrine tumors [58]. As shown in Table 2.1, we divide the properties that have been considered into three categories, according to their relationship with Cell Fate, Cell Cycle, and Oscillations.

Cell Fate

The first properties we verify concern the pancreatic cancer cell's fate, i.e., survival or death. In our model, the following two CTL properties are false,

$$\mathbf{AF}\ Apoptosis, \qquad \mathbf{AF}\ Arrest$$

which mean that the cell does not necessarily have to undergo apoptosis, and that the cell cycle does not necessarily stop. On the other hand, the property

$$\mathbf{AF}\ Proliferate$$

is true, indicating that the cancer cell will necessarily proliferate. Furthermore, since the following "steady state" property is true,

$$\mathbf{AF\ AG}\ Proliferate$$

property	verification result	discussion
Cell Fate		
$\mathbf{AF}\ Apoptosis \vee \mathbf{AF}\ Arrest$	False	the cell does not necessarily have to undergo apoptosis, and the cell cycle does not necessarily stop
$\mathbf{AF}\ Proliferate$	True	the cancer cell will necessarily proliferate
$\mathbf{AF}\ \mathbf{AG}\ Proliferate$	True	proliferation is eventually both unavoidable and permanent
$\mathbf{AF}\ !Apoptosis \wedge \mathbf{AF}\ !Arrest$	True	it is always possible for the cancer cell to reach states in which Apoptosis and Arrest are OFF, thereby making cell proliferation possible
$\mathbf{AF}\ (!Apoptosis \wedge !Arrest \wedge Proliferate)$	False	the model cannot always eventually reach a state in which apoptosis and cell cycle arrest are not inhibited and cell proliferation is active
$\mathbf{AF}\ \mathbf{AG}\ !Apoptosis \vee \mathbf{AF}\ \mathbf{AG}\ !Arrest$	False	inhibition of apoptosis and cell cycle arrest are not unavoidable and permanent
Cell Cycle		
$\mathbf{A}\ (!Proliferate\ \mathbf{U}\ CyclinD)$	True	it is always the case that cell proliferation does not occur until Cyclin D is expressed (or activated)
$\mathbf{AF}\ \mathbf{AG}\ CyclinD$	False	in our model the activation of Cyclin D is not a steady state
$\mathbf{!E}\ (!P53\ \mathbf{U}\ Apoptosis)$	False	apoptosis can be activated even when P53 is not
Oscillations		
$TGF\beta \rightarrow \mathbf{AG}\ ((!NF\kappa B \rightarrow \mathbf{AF}\ NF\kappa B) \wedge (NF\kappa B \rightarrow \mathbf{AF}\ !NF\kappa B)$	True	an initial overexpression of TGFβ always leads to oscillations in NFκB's expression level
$PIP3 \rightarrow \mathbf{AG}\ ((!NF\kappa B \rightarrow \mathbf{AF}\ NF\kappa B) \wedge (NF\kappa B \rightarrow \mathbf{AF}\ !NF\kappa B))$	True	PIP3 has the similar impact on NFκB's expression level
$\mathbf{AG}\ ((P53 \rightarrow \mathbf{AF}\ MDM2) \wedge (MDM2 \rightarrow \mathbf{AF}\ !P53))$	True	overexpression of P53 will always activate MDM2, which will in turn inhibit P53

Table 2.1: Model checking results.

we know that proliferation is eventually both unavoidable and permanent. We now ask whether it is always possible for the cancer cell to reach states in which Apoptosis and Arrest are OFF,

thereby making cell proliferation possible. The following two properties are true.

$$\mathbf{AF}\ !Apoptosis, \qquad \mathbf{AF}\ !Arrest$$

However, the property

$$\mathbf{AF}\ (!Apoptosis \wedge !Arrest\ \wedge\ Proliferate)$$

is false, which means that the model cannot always eventually reach a state in which apoptosis and cell cycle arrest are not inhibited and cell proliferation is active. We also report that the two properties

$$\mathbf{AF\ AG}\ !Apoptosis, \qquad \mathbf{AF\ AG}\ !Arrest$$

are false, so that inhibition of apoptosis and cell cycle arrest are not unavoidable and permanent.

Cell Cycle

We study properties involving the cell cycle, in which the protein Cyclin D is a key player. The next property is true,

$$\mathbf{A}\ (!Proliferate\ \mathbf{U}\ CyclinD)$$

which means that it is always the case that cell proliferation does not occur until Cyclin D is expressed (or activated). This property agrees with the experimental finding that Cyclin D is frequently overexpressed in pancreatic tumors [58]. This indicates that Cyclin D is potentially good target for pancreatic cancer treatments. However, in our model the activation of Cyclin D is not a steady state, since the following property is false.

$$\mathbf{AF\ AG}\ CyclinD$$

Next, we study the role of P53 in apoptosis. It is known that P53 can induce apoptosis through several signaling pathways [112]. Here, we ask whether in our model it is never the case that P53 is not activated until Apoptosis is activated. This question can be encoded in the following CTL formula, which is verified to be false.

$$!\mathbf{E}\ (!P53\ \mathbf{U}\ Apoptosis)$$

Thus, Apoptosis can be activated even when P53 is not.

Oscillations

There have been several experimental demonstrations of oscillations of NFκB signaling [122, 169]. We therefore ask whether our in silico model features oscillations as well. A CTL formula for encoding oscillations in NFκB is the following,

$$\mathbf{AG}\ ((!NF\kappa B\ \rightarrow\ \mathbf{AF}\ NF\kappa B)\ \wedge\ (NF\kappa B\ \rightarrow\ \mathbf{AF}\ !NF\kappa B))$$

which turns out to be false. Next, we check whether overexpression of TGFβ can instead induce NFκB's oscillations. The formula

$$TGF\beta\ \rightarrow\ \mathbf{AG}\ ((!NF\kappa B \rightarrow \mathbf{AF}\ NF\kappa B)\ \wedge\ (NF\kappa B \rightarrow \mathbf{AF}\ !NF\kappa B))$$

is in fact true, which means that an initial overexpression of TGFβ always leads to oscillations in NFκB's expression level. A similar property holds true for PIP3.

$$PIP3 \ \rightarrow \mathbf{AG}\ ((!NF\kappa B \rightarrow \mathbf{AF}\ NF\kappa B)\ \wedge\ (NF\kappa B \rightarrow \mathbf{AF}\ !NF\kappa B))$$

This property is actually an invariant of the model, since the following formula is also true.

$$\mathbf{AG}\ (PIP3 \ \rightarrow \mathbf{AG}\ ((!NF\kappa B \rightarrow \mathbf{AF}\ NF\kappa B)\ \wedge\ (NF\kappa B \rightarrow \mathbf{AF}\ !NF\kappa B)))$$

It would be interesting to test experimentally the properties regarding TGFβ and PIP3. Finally, oscillations have also been detected in the expression level of P53 and MDM2. In [97], oscillations of P53 lasted more than 72 hours after cell damage induced by γ radiation. The next property is true,

$$\mathbf{AG}\ ((P53 \ \rightarrow\ \mathbf{AF}\ MDM2)\ \wedge\ (MDM2 \ \rightarrow\ \mathbf{AF}\ !P53))$$

which means that overexpression of P53 will always activate MDM2, which will in turn inhibit P53.

Chapter 3

Biological Signaling Networks as Qualitative Networks and Improved Bounded Model Checking

The usage of Boolean networks has been one successful approach to the usage of abstraction in biology. Boolean networks call for abstracting the status of each modeled substance as either active (on) or inactive (off). Although a very high level abstraction, it has been found useful to gain better understanding of certain biological systems [188, 193]. The appeal of this discrete approach along with the shortcomings of the very aggressive abstraction, led researchers to suggest various formalisms, such as Qualitative Networks [190] and Gene Regulatory Networks [168] that allow to refine models when compared to the Boolean approach. In these formalisms, every substance can have one of a small discrete number of levels. Dependencies between substances become algebraic functions instead of Boolean functions. Dynamically, a state of the model corresponds to a valuation of each of the substances and changes in values of substances occur gradually based on these algebraic functions. Qualitative networks and similar formalisms (e.g., genetic regulatory networks [204]) have proven to be a suitable formalism to model some biological systems [31, 188, 190, 204].

In this chapter, we consider model checking of qualitative networks. One of the unique features of qualitative networks is that they have no initial states. That is, the set of initial states is the set of all states. Obviously, when searching for specific executions or when trying to prove a certain property we may want to restrict attention to certain initial states. However, the general lack of initial states suggests a unique approach towards model checking. It follows that if a state that is not visited after i steps will not be visited after i' steps for every $i' > i$. These "decreasing" sets of reachable states allow to create a more efficient symbolic representation of all the paths of a certain length.

However, this observation alone is not enough to create an efficient model checking procedure. Indeed, accurately representing the set of reachable states at a certain time amounts to the original problem of model checking (for reachability), which does not scale. In order to address this we use

an over-approximation of the set of states that are reachable by exactly n steps. We represent the over-approximation as a Cartesian product of the set of values that are reachable for each variable at every time point. The computation of this over-approximation never requires us to consider more than two adjacent states of the system. Thus, it can be computed quite efficiently. Then, using this over-approximation we create a much smaller encoding of the set of possible paths in the system.

We test our method on many of the biological models developed using Qualitative Networks. Properties expressed by Linear Temporal Logic (LTL) [178] formulas are translated to an additional set of constraints on the set of paths. Our encoding is based on temporal testers [179]. The experimental results show that there is significant acceleration when considering the decreasing reachability property of qualitative networks. In many examples, in particular larger and more complicated biological models, this technique leads to considerable speedups. The technique scales well with increase of size of models and with increase in length of paths sought for. In particular, for an existing model of Leukemia, our approach works at least 5 times faster than the standard approach and up to 100 times faster in some cases. These results are especially encouraging given the methodology biologists have been using when employing our tools [29]. Typically, models are constructed and then compared with experimental results. The process of model development is a highly iterative process involving trial and error where the biologist compares a current approach with experimental results and refines the model until it matches current experimental knowledge. In this iterative process it is important to give fast answers to queries of the biologist. We hope that with the speed ups afforded by this new technique, model checking could be incorporated into the routine methodology of experimental biologists using our tools.

3.1 Qualitative Networks Example

We start with an example giving some introduction to Qualitative Networks and the usage of LTL model checking in this context.

Figure 3.1 shows a model representing aspects of cell-fate determination during C. elegans vulval development [89]. The part shown in the figure includes three cells. Each cell in the model represents a vulval precursor cell and the elements inside it represent proteins whose level of activity influences the decision of the cell as to which part of the vulva the descendants of the cell should form. All cells execute the same program and it is the communication between the cells themselves as well as communication between the cells and additional parts of the model (i.e., external signals) that determine a different fate for each of the cells. Understanding cell-fate determination is crucial for our understanding of normal development processes as well as occasions where these go wrong such as disease and cancer. The pictorial view gives rise to a formal model expressed as a qualitative network [190]. Formal definitions are in the next section.

Each of the cells in the model includes executing components, for example LET-60, that correspond to a single variable. Each variable v holds a value, which is a number in $0, 1, \cdots, N_v$, where N_v is the granularity of the variable. Specifically, in Figure 1 all variables range over 0,1,2. A target function, T_v, defined over the values of variables affecting v (i.e., having incoming arrows

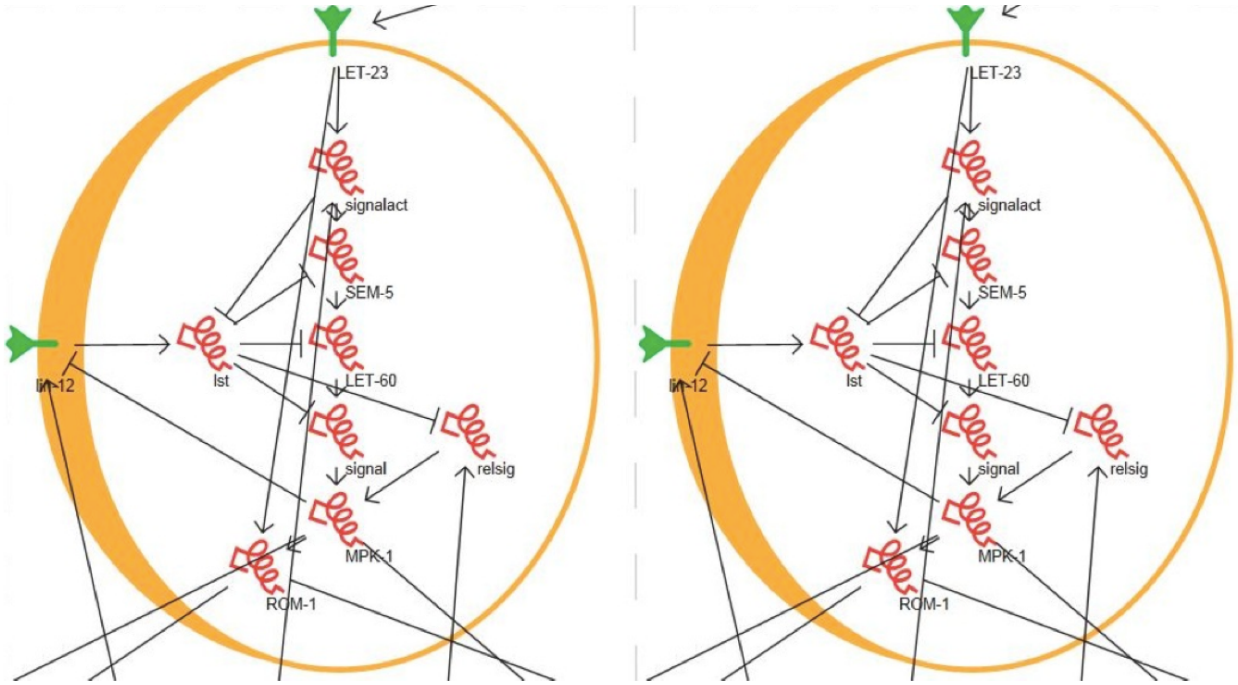


Figure 3.1: A pictorial view of part of a model describing aspects of cell-fate determination during *C. elegan's* vulval development [89]. The image shows two cells having the "same program". Neighboring cells and connections between cells are not shown.

into v),determines how v is updated: if $v < T_v$ and $v < N_v$ then $v' = v + 1$, if $v > T_v$ and $v > 0$ then $v' = v - 1$, else v does not change. In a qualitative network all variables are updated synchronously in parallel.

Intuitively, the update function of each variable is designed such that the value of the variable follows its target function, which depends on other variables. In the biological setting, the typical target of a variable, v, combines the positive influence of variables $w_1, w_2, \cdots, w_s$ with the negative influence of variables $w_{s+1}, w_{s+2}, \cdots, w_{s+r}$:

$$T_v(w_1, w_2, \cdots, w_{s+r}) \;=\; max(0,\; \left\lfloor \frac{1}{s}\sum_{k=1}^{s} w_k \;-\; \frac{1}{r}\sum_{k=1}^{r} w_{s+k} \;+\; \frac{1}{2} \right\rfloor)$$

Graphically, this is often represented as an influence graph with $\rightarrow$ edges between each of $w_1, w_2, \cdots, w_s$ and v and $\dashv$ edges between each of $w_{s+1}, w_{s+2}, \cdots, w_{s+r}$ and v. More complicated target functions can be defined using algebraic expressions over $w_1, \cdots, w_{s+r}$. We refer the reader to [29, 69] for further details about other modeling options.

Specifically, in the model above, the target of 1st is:

$$T_{1st} \;=\; min(2 \;-\; \mathbf{signalact},\; 1) \;*\; \mathbf{lin-12}$$

This models activation by lin-12 and inhibition by signalact. However, inhibition occurs only when signalact is at its maximal level (2). When inhibition is not maximal the target follows the value of

lin-12. The target of SEM-5 is:

$$T_{SEM-5} = max(0,\ 2 - ((2 - \mathbf{signalact}) * (max(\mathbf{1st} - 1,\ 0) + 1)))$$

This function means that lst inhibits SEM-5 and signalact activates it. However, activation takes precedence: inhibition takes effect only in case that activation is not at its maximum value (2), and only when inhibition is at its maximum value (2). Otherwise, the target follows the value of its activator (signalact).

Models are analyzed to ensure that they reproduce behavior that is observed in experiments. A mismatch between the model and experimental observations signifies that something is wrong with our understanding of the system. In such a case, further analysis is required in order to understand whether and how the model needs to be changed. Models are usually analyzed by simulating them and following the behavior of components. A special property of interest in these types of models is that of stability: there is a unique state that has a self loop and all executions lead to that state [69, 190]. When a model does stabilize it is interesting to check the value of variables in the stabilization point. In addition, regardless of whether the model is stabilizing or not, model checking is used to prove properties of the model or to search for interesting executions. For the model in Figure 3.1 the following properties, e.g., are of interest.

- Do there exist executions leading to adjacent primary fates in which increase of LS happens after down-regulation of lin-12? This property is translated to an LTL formula of the following format:

  $$\theta \wedge \mathbf{FG}\ f_{i,j} \wedge (\neg d_i\ \mathbf{U}\ l_i) \wedge (\neg d_j\ \mathbf{U}\ l_i)$$

  where θ is some condition on initial states, $f_{i,j}$ is the property characterizing the states in which VPCs i and j are both in primary fate, d_i is the property that lin-12 is low in VPC i, l_i is the property that d_i is high in VPC i, and d_j and l_j are similar for VPC j. This property is run in positive mode, i.e., we are searching for execution that satisfies this property.

- Is it true that for runs starting from a given set of states the sequence of occurrences leading to fate execution follows the pattern: MPK-1 increases to high level then lin-12 is down-regulated, and then LS is activated. This property is translated to an LTL formula of the following format:

  $$\theta \Rightarrow \mathbf{F}\ (m_i \wedge \mathbf{XF}\ (l_i \wedge \mathbf{XF}\ d_i))$$

  where θ is some condition on initial states, m_i is the property characterizing states in which VPC i has a high level of MPK-1, l_i is the property characterizing states in which VPC i has a low level of lin-12, and d_i is the property characterizing states in which VPC i has a high level of LS. This property is run in negative (model checking) mode, i.e., we are searching for executions falsifying this property and expecting the search to fail.

3.2 Qualitative Networks

In this section, we formally introduce the Qualitative Networks (QN) framework and recall the definition of linear temporal logic (LTL).

A qualitative network (QN), $Q(V,\ T,\ N)$, of granularity $N+1$ consists of variables: $V = (v_1, v_2, \cdots, v_n)$. (Note that, for simplicity, we assume that all variables have the same range $\{0, \cdots, N\}$. The extension to individual ranges is not complicated. Our implementation supports individual ranges for variables.) A state of the system is a finite map $s:\ V \to \{0, 1, \cdots N\}$. Each variable $vi \in V$ has a target function $T_i \in T$ associated with it: $T_i\ :\ \{0, 1, \cdots, N\}^n \to \{0, 1, \cdots, N\}$. Qualitative networks update the variables using synchronous parallelism.

Target functions in qualitative networks direct the execution of the network: from state $s = (d_1, d_2, \cdots, d_n)$, the next state $s' = (d'_1, d'_2, \cdots, d'_n)$ is computed by:

$$d'_i = \begin{cases} d_i + 1 & d_i < T_i(s) \text{ and } d_i < N, \\ d_i - 1 & d_i > T_i(s) \text{ and } d_i > 0, \\ d_i & \text{otherwise} \end{cases} \qquad (1)$$

A target function of a variable v is typically a simple algebraic function, such as sum, over several other variables $w_1, w_2, \cdots, w_m$. We often say that v depends on $w_1, w_2, \cdots, w_m$ or that $w_1, w_2, \cdots, w_m$ are inputs of v. In the following, we use the term *network* to refer to a qualitative network.

A QN $Q(V,\ T,\ N)$ defines a state space $\Sigma = \{s:\ V \to \{0, 1, \cdots, N\}\}$ and a transition function $f:\ \Sigma \to \Sigma$, where $f(s) = s'$ such that for every $v \in V$ we have $s'(v)$ depends on $T_{v(s)}$ as in Equation (1). For a state $s \in \Sigma$ we denote by $s(v)$ also by s_v. In particular, $f_v(s) = f(s)(v)$ is the value of v in $f(s)$. We say that a state s is *recurring* if it is possible to get back to s after a finite number of applications of f. That is, if for some $i > 0$ we have $f^i(s) = s$. As the state space of a qualitative network is finite, the set of recurring states is never empty. We say that a network is stabilizing if there exists a unique recurring state s. That is, there is a unique state s such that $f(s) = s$, and for every other state s' and every $i > 0$ we have $fi(s') \neq s'$. Intuitively, this means that starting from an arbitrary state, we always end up in a fixpoint and always the same one. A run of a QN $Q(V,\ T,\ N)$ is an infinite sequence $r = s_0, s_1, \cdots$ such that for every $i \geq 0$ we have $s_i \in \Sigma$ and $s_{i+1} = f(s_i)$.

We now define LTL over runs of qualitative networks as follows. For every variable $v \in V$ and every value $n \in 0, 1, \cdots, N$, we define an atomic proposition $v \sim n$, where $\sim \in >, \geq, \leq, <$. Let AP denote the set of all atomic propositions (for a network Q). The set of LTL formulas is:

$$\varphi ::= \text{AP} \mid \varphi \vee \varphi \mid \neg\varphi \mid \mathbf{X}\varphi \mid \varphi \mathbf{U} \varphi$$

As usual, we introduce $\wedge$, $\rightarrow$, $\mathbf{F}$, and $\mathbf{G}$ as syntactic sugar.

An LTL formula φ is satisfied over a run $r = s_0, s_1, \cdots$ in location i, denoted $r, i \models \varphi$ according to the following:

- For $\varphi = v \sim n \in \text{AP}$ we have $r, i \models \varphi$ if $s_i(v) \sim n$.
- For $\varphi = \neg\psi$ we have $r, i \models \varphi$ if it is not the case that $r, i \models \psi$.
- For $\varphi = \psi_1 \vee \psi_2$ we have $r, i \models \varphi$ if either $r, i \models \psi_1$ or $r, i \models \psi_2$.
- For $\varphi = \mathbf{X}\psi$ we have $r, i \models \varphi$ if $r, i+1 \models \psi$.
- For $\varphi = \psi_1 \mathbf{U} \psi_2$ we have $r, i \models \varphi$ if there is $j \geq i$ such that $r, j \models \psi_2$ and for every $i \leq k < j$ we have $r, k \models \psi_1$.

We say that a run r satisfies an LTL formula φ, denoted $r \models \varphi$ if $r, 0 \models \varphi$. Given a Qualitative Network Q, we say that Q satisfies an LTL formula φ, denoted $Q \models \varphi$, if for every run r of Q we have $r \models \varphi$. In case that $Q \not\models \varphi$ a *counterexample* is a run r such that $r \not\models \varphi$.

We use bounded model checking [64] for checking whether a qualitative network satisfies a given LTL formula φ. Intuitively, we search for a run of a certain structure (and length) that does not satisfy the formula by constructing a Boolean formula whose satisfiability corresponds to such a run. Searching for a counterexample of length l means that we (1) create Boolean variables that represent the state of the system in l different time points, (2) add constraints that enforce that the transition of the qualitative network holds between every two consecutive time points, (3) add constraints that enforce that the transition of the qualitative network holds between the state at time $l-1$ (last state) and some previous state (i.e., that the sequence of states ends in a loop), and (4) add Boolean variables and constraints that enforce satisfaction of the (negation of) the temporal property.

In order to create a Boolean encoding of the LTL formula we use a variant of the temporal testers approach in [179]. Specifically, for every temporal subformula (and every time point in the trace) we add a Boolean variable that tracks the truth value of the subformula at that time. The truth value of these variables are connected to the truth values of propositions (encoded through the state of the model) and truth values of other subformulas. In addition, we add constraints that enforce satisfaction of eventualities in the loop. In order to search for a trace that satisfies a certain LTL formula we add the encoding of the formula to the trace. Satisfiability then provides a run satisfying the formula. To prove that all runs up of a certain length satisfy a formula, we add the encoding of the negation of the formula to the trace. Unsatisfiability then provides a proof that no run (of the given length) satisfies the formula.

3.3 Decreasing Reachability Sets

A notable difference between QNs and "normal" transition systems is that QNs do not specify initial states. For example, for the classical stability analysis all states are considered as initial states. It follows that if a state s of a QN is not reachable after i steps, it is not reachable after i' steps for every $i' > i$. Thus, there is a decreasing sequence of sets $\Sigma_0 \supseteq \Sigma_1 \supseteq \cdots \supseteq \Sigma_l$ such that searching for runs of the network can be restricted to the set of runs of the form $\Sigma_0,\ \Sigma_1,\ \cdots,\ (\Sigma_l)^\omega$.

Here we show how to take advantage of this fact in constructing a more scalable model checking algorithm for qualitative networks.

Consider a Qualitative Network $Q(V, T, N)$ with set of states $\Sigma : V \to \{0, \cdots, N\}$. We say that a state $s \in \Sigma$ is reachable by exactly i steps if there is some run $r = s_0, s_1, \cdots$ such that $s = s_i$. Dually, we say that s is not reachable by exactly i steps if for every run $r = s_0, s_1, \cdots$ we have $s_i \neq s$.

Lemma 3.1. If a state s is not reachable by exactly i steps then it is not reachable by exactly i' steps for every $i' > i$.

The algorithm 1 computes a decreasing sequence $\Sigma_0 \supset \Sigma_1 \supset \cdots \supset \Sigma_{j-1}$ such that all states that are reachable by exactly i steps are in Σ_i if $i < j$ and in Σ_{j-1} if $i \geq j$. We note that the definition of Σ_{j+1} in line 5 is equivalent to the standard $\Sigma_{j+1} = f(\Sigma_j)$, where function $f(\cdot)$ is used to compute the next reachable set. However, we choose to write it as in the algorithm below in order to stress that only states in Σ_j are candidates for inclusion in Σ_{j+1}. Given the sets $\Sigma_0, \cdots, \Sigma_{j-1}$, every run $r = s_0, s_1, \cdots$ of Q satisfies $s_i \in \Sigma_i$ for $i < j$ and $s_i \in \Sigma_{j-1}$ for $i \geq j$. In particular, if $Q \not\models \varphi$ for some LTL formula φ, then the run witnessing the unsatisfaction of φ can be searched for in this smaller space of runs. Unfortunately, the algorithm 1 is not feasible. Indeed, it amounts to computing the exact reachability sets of the QN Q, which does not scale well [69].

Algorithm 1 Concrete Decreasing Reachability

1: $\Sigma_0 = \Sigma$;
2: $\Sigma_{-1} = \emptyset$;
3: $j = 0$;
4: **while** $\Sigma_{j-1} \neq \Sigma_j$ **do**
5: $\quad \Sigma_{j+1} = \Sigma_j \setminus \{s' \in \Sigma | \forall s \in \Sigma \cdot s' \neq f(s)\}$;
6: $\quad j++$;
7: **end while**
8: **return** $\Sigma_0, \cdots, \Sigma_{j-1}$

In order to effectively use Lemma 1 we combine it with over-approximation, which leads to a scalable algorithm. Specifically, instead of considering the set Σ_k of states reachable at step k, we identify for every variable $v_i \in V$ the domain $D_{i,k}$ of the set of values possible at time k for variable v_i. Just like the general set of states, when we consider the possible values of variable v_i we get that $D_{i,0} \supseteq D_{i,1} \supseteq \cdots \supseteq D_{i,l}$. The advantage is that the sets $D_{i,k}$ for all $v_i \in V$ and $k > 0$ can be constructed by induction by considering only the knowledge on previous ranges and the target function of one variable.

Consider the algorithm 2. For each variable, it initializes the set of possible values at time 0 as the set of all values. Then, based on the possible values at time j, it computes the possible values at time $j + 1$. The actual check can be either implemented explicitly if the number of inputs of all target functions is small (as in most cases) or symbolically. Considering only variables (and values) that are required to decide the possible values of variable v_i at time j makes the problem much simpler than the general reachability problem. Notice that, again, only values that are possible at

time j need be considered at time $j+1$. That is, $D_{i,j+1}$ starts as empty (line 6) and only values from $D_{i,j}$ are added to it (lines 7 - 10). As before, $D_{i,j+1}$ is the projection of $f(D_{1,j} \times \cdots \times D_{m,j})$ on v_i. The notation used in the algorithm above stresses that only states in $D_{i,j}$ are candidates for inclusion in $D_{i,j+1}$.

Algorithm 2 Abstract Decreasing Reachability

1: $\forall v_i \in V \cdot D_{i,0} = \{0, 1, \cdots, N\}$;
2: $\forall v_i \in V \cdot D_{i,-1} = \emptyset$;
3: $j = 0$;
4: **while** $\exists v_i \in V \cdot D_{i,j} \neq D_{i,j-1}$ **do**
5: **for each** $v_i \in V$ **do**
6: $D_{i,j+1} = \emptyset$;
7: **for each** $d \in D_{i,j}$ **do**
8: **if** $\exists (d_1, \cdots, d_m) \in D_{1,j} \times \cdots \times D_{m,j} \cdot f_v(d_1, \cdots, d_m) = d$ **then**
9: $D_{i,j+1} = D_{i,j+1} \cup \{d\}$;
10: **end if**
11: **end for**
12: **end for**
13: **end while**
14: $j++$;
15: **return** $\forall v_i \in V, \forall j' \leq j \cdot D_{i,j'}$

The algorithm produces very compact information that enables to follow with a search for runs of the QN. Namely, for every variable v_i and for every time point $0 \leq k < j$ we have a decreasing sequence of domains

$$D_{i,0} \supseteq D_{i,1} \supseteq \cdots \supseteq D_{i,k}.$$

Consider a Qualitative Network $Q(V, T, N)$, where $V = \{v_1, \cdots, v_n\}$ and a run $r = s_0, s_1, \cdots$. As before, every run $r = s_0, s_1, \cdots$ satisfies that for every i and for every t we have $s_t(v_i) \in D_{i,t}$ for $t < j$ and $s_t(v_i) \in D_{i,j-1}$ for $t \geq j$.

We look for paths that are in the form of a lasso, as we explain below. We say that r is *a loop of length l* if for some $0 < k \leq l$ and for all $m \geq 0$ we have $s_{l+m} = s_{l+m-k}$. That is, the run r is obtained by considering a prefix of length $l-k$ of states and then a loop of k states that repeats forever. A search for a loop of length l that satisfies an LTL formula φ can be encoded as a bounded model checking query as follows. We encode the existence of l states $s_0, \cdots, s_{l-1}$. We use the decreasing reachability sets $D_{i,t}$ to force state s_t to be in $D_{0,t} \times \cdots \times D_{n,t}$. This leads to a smaller encoding of the states $s_0, \cdots, s_{l-1}$ and to smaller search space. We add constraints that enforce that for every $0 \leq t < l-1$ we have $s_{t+1} = f(s_t)$. Furthermore, we encode the existence of a time $l-k$ such that $s_{l-k} = f(s_{l-1})$. We then search for a loop of length l that satisfies φ . It is well known that if there is a run of Q that satisfies φ then there is some l and a loop of length l that satisfies φ . We note that sometimes there is a mismatch between the length of loop sought for and length of sequence of sets (j) produced by the algorithm 2. Suppose that the algorithm

returns the sets $D_{i,t}$ for $v_i \in V$ and $0 \leq t < j$. If $l > j$, we use the sets $D_{i,j-1}$ to "pad" the sequence. Thus, states $s_j, \cdots, s_{l-1}$ will also be sought in $\prod_i D_{i,j-1}$. If $l < j$, we use the sets $D_{i,0}, \cdots, D_{i,l-2}, D_{i,j-1}$ for $v_i \in V$. Thus, only the last state s_{l-1} is ensured to be in our "best" approximation $\prod_i D_{i,j-1}$. A detailed explanation of how we encode the decreasing reachability sets as a Boolean satisfiability problem is given in [62].

3.4 Results for Various Biological Models

We implemented this technique to work on models defined through our tool BMA [29]. Here, we present experimental results of running our implementation on a set of different biological models, including a total of 22 benchmark problems from various sources (skin cells differentiation models by ourselves, diabetes models from [31], models of cell fate determination during *C. elegans* vulval development, a Drosophila embryo development model from [188], Leukemia models constructed by ourselves, and a few additional examples constructed by ourselves). The number of variables in the models and the maximal range of variables is reported in Table 3.1.

Model name	#Vars	Range	Model name	#Vars	Range
2var_unstable	2	0..1	Bcr-Abl	57	0..2
Bcr-AblNoFeedbacks	54	0..2	BooleanLoop	2	0..1
NoLoopFound	5	0..4	Skin1D_TF_0	75	0..4
Skin1D_TF_1	75	0..4	Skin1D	75	0..4
Skin2D_3X2_0	90	0..4	Skin2D_3X2_1	90	0..4
Skin2D_3X2_2	90	0..4	Skin2D_5X2_TF	198	0..4
Skin2D_5X2	198	0..4	SmallTestCase	3	0..4
SSkin1D_TF_0	30	0..4	SSkin1D_TF_1	31	0..4
SSkin1D	30	0..4	SSkin2D_3X2	40	0..4
VerySmallTest	2	0..4	VPC_lin15ko	85	0..2
VPC_Non_stable	33	0..2	VPC_stable	43	0..2

Table 3.1: Number of variables in models and their ranges.

Our experiments compare two encodings. One encoding is explained in algorithm 2, referred to as "opt" (for optimized). the other considers l states $s_0, \cdots, s_l$ where $s_t(v_i) \in \{0, \cdots, N\}$ for every t and every i. That is, for every variable v_i and every time point $0 \leq t \leq l$ we consider the set $D_{i,t} = 0, \cdots, N$. This encoding is referred to as "naïve". In both cases we use the same encoding to a Boolean satisfiability problem. Further details about the exact encoding can be found in [62].

We perform two kinds of experiments. First, we search for loops of length 10, 20, $\cdots$, 50 on all the models for the optimized and naïve encodings. Second, we search for loops that satisfy a certain LTL property (either as a counterexample to model checking or as an example run satisfying a given property). Again, this is performed for both the optimized and the naïve encodings. LTL properties are considered only for four biological models. The properties were suggested by our collaborators as interesting properties to check for these models. For both experiments, we report

separately on the global time and the time spent in the SAT solver. All experiments were run on an Intel Xeon machine with CPU X7560@2.27GHz running Windows Server 2008 R2 Enterprise.

In Tables 3.2 and 3.3 we include experimental results for the search for loops. We compare the global run time of the optimized search vs the naïve search. The global run time for the optimized search includes the time it takes to compute the sequence of decreasing reachability sets. Accordingly, in some of the models, especially the smaller ones, the overhead of computing this additional information makes the optimized computation slower than the naïve one. For information we include also the net runtime spent in the SAT solver.

In Table 3.4 we include experimental results for the model checking experiment. As before, we include the results of running the search for counterexamples of lengths 10, 20, 30, 40, and 50. We include the total runtime of the optimized vs the naïve approaches as well as the time spent in the SAT solver. As before, the global runtime for the optimized search includes the computation of the decreasing reachability sets. The properties in the table are of the following form. Let I, $a \cdots d$ denote formulas that are Boolean combinations of propositions.

- $I \rightarrow (\neg a)\ \mathbf{U}\ b$: we check that the sequence of events when starting from the given initial states (I) satisfies the order that b happens before a.
- $I \wedge \mathbf{FG}\ a \wedge \mathbf{F}\ (b \wedge \mathbf{XF}\ c)$: we check that the model gets from some states (I) to a loop that satisfies the condition a and the path leading to the loop satisfies that b happens first and then c.
- $I \wedge \mathbf{FG}\ a \wedge \mathbf{F}\ (b \wedge \mathbf{XF}\ (c \wedge \mathbf{XF}\ d))$: we extend the previous property by checking the sequence a then b then c and then d.
- $I \wedge \mathbf{FG}\ a \wedge (\neg b)\ \mathbf{U}\ c$: we check that the model gets from some states (I) to a loop that satisfies the condition a and the path leading to the loop satisfies that b cannot happen before c.
- $\mathbf{GF}\ a \wedge \mathbf{GF}\ b$: we check for the existence of loops that exhibit a form of instability by having states that satisfy both a and b.

When considering the path search, on many of the smaller models the new technique does not offer a significant advantage. However, on larger models, and in particular the two dimensional skin model (Skin2D_5X2 from [190]) and the Leukemia model (Bcr_Abl) the new technique is an order of magnitude faster. Furthermore, when increasing the length of the path it scales a lot better than the naïve approach. When model checking is considered, the combination of the decreasing reachability sets accelerates model checking considerably. While the naïve search increases considerably to the order of tens of minutes, the optimized search remains within the order of 10s, which affords a "real-time" response to users.

Length of loop	10				20				30			
	Global Time (s)		Sat Time (s)		Global Time (s)		Sat Time (s)		Global Time (s)		Sat Time (s)	
Model name	Naïve	Opt	Naïve	Opt	Naïve	Opt	Naïve	Opt	Naïve	Opt	Naïve	Opt
2var_unstable	6.92	0.78	0.21	0	0.46	0.54	0	0	0.51	0.57	0	0
Bcr-Abl	67.76	9.32	28.92	1.46	196.68	9.49	142.41	1.31	281.27	10.29	108.14	1.85
Bcr-AblNoFeedbacks	66.52	6.77	29.58	0.71	201.59	6.71	101.69	0.56	307.60	6.60	219.72	0.62
BooleanLoop	0.49	0.51	0	0	0.48	0.57	0.01	0	0.53	0.59	0.01	0.01
NoLoopFound	0.78	0.74	0.06	0.01	1.14	0.93	0.09	0.03	1.45	1.04	0.10	0.06
Skin1D_TF_0	136.21	140.78	122.85	127.47	218.52	80.33	191.06	55.23	127.28	96.49	86.05	60.06
Skin1D_TF_1	167.32	173.03	154.00	159.55	698.47	445.32	670.77	419.24	883.35	572.03	842.06	536.04
Skin1D	90.92	68.82	77.63	54.54	45.67	23.21	17.55	8.77	133.72	23.46	92.36	8.13
Skin2D_3X2_0	567.31	640.71	545.49	618.44	238.28	205.15	192.28	162.14	164.79	218.77	93.45	153.11
Skin2D_3X2_1	910.08	553.27	891.70	535.02	82.04	117.48	44.70	82.79	122.77	219.04	64.96	167.65
Skin2D_3X2_2	315.20	169.92	293.45	151.64	121.12	36.58	74.49	18.74	188.78	39.36	114.81	20.15
Skin2D_5X2_TF	511.31	223.93	459.38	182.65	1466.90	391.96	1378.80	353.06	1275.30	73.77	1135.25	35.83
Skin2D_5X2	343.96	85.64	300.03	56.71	721.58	57.20	630.92	28.46	965.24	48.26	828.12	16.83
SmallTestCase	0.53	0.54	0.01	0	0.54	0.73	0.01	0	0.60	0.54	0.01	0
SSkin1D_TF_0	70.71	69.00	63.71	61.93	21.35	20.71	5.87	5.93	33.07	32.74	12.52	12.34
SSkin1D_TF_1	9.77	10.05	2.88	2.93	22.85	26.02	8.23	9.04	35.61	35.16	15.12	14.96
SSkin1D	145.28	146.74	138.61	139.76	32.00	33.38	18.29	18.51	33.89	33.80	13.57	13.49
SSkin2D_3X2	301.33	158.62	286.80	148.08	63.46	50.12	35.44	36.14	86.26	32.41	44.30	14.91
VerySmallTest	0.37	0.42	0	0	0.39	0.43	0.01	0	0.40	0.43	0.01	9
VPC_lin15ko	8.31	6.81	3.35	0.32	14.87	6.74	5.13	0.26	21.99	6.76	7.42	0.20
VPC_Non_stable	3.43	3.40	0.85	0.26	6.02	3.95	1.23	0.29	9.35	4.87	2.10	0.62
VPC_stable	3.31	4.79	0.74	0.14	5.84	4.79	0.99	0.18	9.10	4.67	1.92	0.14

Table 3.2: Searching for loops (10, 20, 30).

Length of loop	40				50			
	Global Time (s)		Sat Time (s)		Global Time (s)		Sat Time (s)	
Model name	Naïve	Opt	Naïve	Opt	Naïve	Opt	Naïve	Opt
2var_unstable	0.54	0.60	0.01	0	1.05	0.64	0.01	0.01
Bcr-Abl	667.22	11.54	552.90	2.74	1019.68	11.94	869.56	2.76
Bcr-AblNoFeedbacks	574.61	6.79	316.07	0.64	857.17	6.90	719.21	0.69
BooleanLoop	0.54	0.60	0.01	0.01	0.59	0.66	0.01	0.01
NoLoopFound	1.90	1.15	0.23	0.04	2.23	1.34	0.22	0.05
Skin1D_TF_0	126.13	153.85	68.31	104.11	224.38	247.93	149.55	182.58
Skin1D_TF_1	108.84	160.72	52.01	112.33	167.86	290.97	91.13	228.46
Skin1D	122.73	29.39	64.99	12.84	259.75	34.04	182.50	16.09
Skin2D_3X2_0	391.08	325.43	293.83	237.89	470.89	663.87	341.24	545.49
Skin2D_3X2_1	196.99	271.98	118.22	202.01	476.94	557.09	366.61	464.88
Skin2D_3X2_2	413.13	44.06	314.95	23.75	445.78	47.51	308.71	25.18
Skin2D_5X2_TF	3067.08	93.15	2649.01	48.12	5135.87	82.38	3956.13	34.25
Skin2D_5X2	2403.53	47.69	2149.43	14.87	4025.83	56.86	3254.90	18.18
SmallTestCase	0.96	0.57	0.02	0	0.77	0.58	0.02	0
SSkin1D_TF_0	44.81	42.03	13.52	13.37	58.09	57.45	22.64	21.91
SSkin1D_TF_1	43.97	46.26	15.88	16.13	60.46	60.52	22.77	24.35
SSkin1D	41.13	41.49	12.48	12.59	60.77	61.34	22.82	22.87
SSkin2D_3X2	117.64	42.86	50.82	20.36	157.07	51.19	80.54	22.95
VerySmallTestCase	0.48	0.44	0	0	0.81	0.67	0.01	0
VPC_lin15ko	27.04	6.94	7.34	0.20	45.70	7.14	20.78	0.23
VPC_Non_stable	14.58	5.64	2.36	0.65	16.21	6.50	4.10	1.07
VPC_stable	13.13	6.66	3.44	0.12	17.07	4.99	5.07	0.20

Table 3.3: Searching for loops (40, 50).

Model name	Global Time (s)		Sat Time (s)		Ratio		
	Naïve	Opt	Naïve	Opt	Global	Sat	
Bcr-Abl1	69.30	9.04	26.67	0.90	7.66	29.61	sat
Bcr-Abl1	188.13	12.21	87.70	1.42	15.40	61.47	sat
Bcr-Abl1	380.24	13.12	292.21	2.01	28.96	145.02	sat
Bcr-Abl1	648.02	12.37	349.70	2.30	52.38	151.87	sat
Bcr-Abl1	1005.37	11.52	588.34	2.17	87.19	270.93	sat
Bcr-Abl2	47.04	10.97	9.94	0.72	4.28	13.76	Unsat
Bcr-Abl2	136.48	8.62	41.04	0.75	15.82	54.66	Unsat
Bcr-Abl2	285.28	11.28	112.35	0.77	25.28	144.58	Unsat
Bcr-Abl2	561.65	9.29	443.91	0.80	60.41	553.83	Unsat
Bcr-Abl2	781.64	12.03	408.55	0.87	64.96	465.55	Unsat
Bcr-Abl3	48.64	8.47	9.54	0.83	5.74	11.45	Unsat
Bcr-Abl3	133.83	9.10	38.68	1.11	14.69	34.81	Unsat
Bcr-Abl3	283.73	9.45	106.61	1.16	30.01	91.28	Unsat
Bcr-Abl3	596.50	9.50	466.01	1.18	62.78	394.48	Unsat
Bcr-Abl3	853.53	10.05	480.77	1.36	84.89	351.99	Unsat
Bcr-Abl4	75.27	9.19	44.50	0.80	8.18	55.31	sat
Bcr-Abl4	202.06	9.95	143.49	1.53	20.30	93.50	sat
Bcr-Abl4	296.02	11.35	116.24	2.54	26.07	45.75	sat
Bcr-Abl4	740.39	11.00	116.24	2.54	26.07	45.74	sat
Bcr-Abl4	975.97	10.42	823.53	1.10	93.63	747.14	sat
Bcr-AblNoFeedbacks1	42.98	6.25	7.94	0.40	6.87	19.51	Unsat
Bcr-AblNoFeedbacks1	163.33	8.18	95.43	0.77	19.95	123.90	Unsat
Bcr-AblNoFeedbacks1	302.17	6.41	122.25	0.46	47.07	260.90	Unsat
Bcr-AblNoFeedbacks1	493.28	6.41	314.24	0.45	76.92	686.28	Unsat
Bcr-AblNoFeedbacks1	809.97	6.45	680.70	0.46	125.51	1461.69	Unsat
Bcr-AblNoFeedbacks2	44.88	6.39	6.59	0.40	7.01	16.27	Unsat
Bcr-AblNoFeedbacks2	117.96	6.34	20.98	0.39	18.58	53.61	Unsat
Bcr-AblNoFeedbacks2	312.73	7.59	231.87	0.46	41.18	500.00	Unsat
Bcr-AblNoFeedbacks2	527.40	6.31	423.61	0.39	83.46	1084.74	Unsat
Bcr-AblNoFeedbacks2	751.45	6.83	362.09	0.44	109.87	806.35	Unsat
Bcr-AblNoFeedbacks3	60.99	6.95	20.45	0.64	8.77	31.64	sat
Bcr-AblNoFeedbacks3	204.66	7.06	144.58	0.61	28.97	233.95	sat
Bcr-AblNoFeedbacks3	356.33	8.81	267.48	0.49	40.42	539.32	sat
Bcr-AblNoFeedbacks3	Time out	7.06	Time out	0.42	N/A	N/A	sat
VPC_non_stable1	30.14	10.83	4.83	0.69	2.78	6.93	Unsat
VPC_non_stable2	17.42	9.85	3.59	1.11	1.76	3.24	sat
VPC_non_stable3	52.01	11.91	26.69	1.48	4.36	17.93	Unsat
VPC_non_stable4	19.53	8.31	7.08	0.60	2.34	11.77	Unsat
VPC_stable1	3.75	5.11	0.31	0.07	0.73	3.99	Unsat
VPC_stable2	5.53	5.32	0.86	0.11	1.04	7.41	sat

Table 3.4: Model checking results.

Chapter 4

Phage-based Bacteria Killing as A Nonlinear Hybrid Automaton and δ-complete Decision-based Bounded Model Checking

Due to the widespread misuse and overuse of antibiotics, drug resistant bacteria now pose significant risks to health, agriculture and the environment. Therefore, we were interested in an alternative to conventional antibiotics, a phage therapy. Phages, or bacteriophages, are viruses that infect bacteria and have evolved to manipulate the bacterial cells and genome, making resistance to bacteriophages difficult to achieve. However, many phages are temperate, meaning that they can enter a lysogenic phase and therefore not lyse and kill the host bacteria. The addition of a phototoxic protein - KillerRed [176] - to the system offers a second method of killing those bacteria targeted by a lysogenic phage. In this chapter, we constructed a hybrid model of a bacteria killing procedure that mimics the stages through which bacteria change when phage therapy is adopted. Our model was designed according to an experimental procedure to engineer a temperate phage, Lambda (λ), and then kill bacteria via light-activated production of superoxide. We applied δ-complete decision based bounded model checking [95] to our model and the results show that such an approach can speed up evaluation of the system, which would be impractical or possibly not even feasible to study in a wet lab.

4.1 The KillerRed Model

The discovery of antibiotics has been quickly followed by the development of antibiotic resistance. New medicines are becoming increasingly scarce in tackling this issue. The document released by CDC (Centers for Disease Control and Prevention), "Antibiotic Resistance Threats in the United States, 2013" [4], intends to raise public awareness of the problems associated with overuse and

misuse of antibiotics and to outline the threats to society caused by these organisms. The organisms have been categorized by hazard level as urgent, serious and concerning. Over 2 million illnesses and $23,000$ deaths per year are a direct result of antibiotic resistance.

There are multiple mechanisms of antibiotic resistance. First, altered permeability of the antimicrobial agent is suggested to be due to the inability of the agent to enter the bacterial cell, or alternatively, due to the active export of the agent from the cell. Second, resistance is often the result of the production of an enzyme that is capable of inactivating the antimicrobial agent. Next, resistance can arise due to alteration of the target site for the antimicrobial agent. Finally, resistance can result from the acquisition of a new enzyme to replace the sensitive one, thus replacing the pathway that was originally sensitive to antibiotic to another pathway.

The CDC outlines four core actions that will help fight deadly infections [4]: (a) preventing infections and the spread of resistance; (b) tracking resistant bacteria; (c) improving the use of today's antibiotics; and (d) promoting the development of new antibiotics and developing new diagnostic tests for resistant bacteria. Recently, we have addressed this problem by designing a new system that relies on phage-based therapy. Phages, or bacteriophages, are viruses that infect bacteria and have evolved to manipulate the bacterial cells and genome, making resistance to bacteriophages difficult to achieve. Bacteriophages are complex and utilize many host pathways such that they cannot be inactivated or bypassed. Bacteriophages infect only specific hosts and can kill the host by cytolysis. However, many phages are temperate, meaning that they can enter a lysogenic phase and therefore not lyse and kill the host bacteria. The addition of a phototoxic protein to the system offers a second method of killing those bacteria targeted by a lysogenic phage. Thus, our system, shown in Figure 4.1, explores the possibility that temperate phages can also be used for phage therapy and bacteria killing applications. We incorporated several proteins (KillerRed [176], SuperNova [202]), that have been shown to be phototoxic and that provide another level of controlled bacteria killing.

We have modeled synthesis and action of KillerRed that occurs over three main phases of a typical photobleaching experiment: induction at 37°C, storage at 4°C to allow for protein maturation, and photobleaching at room temperature. Within these phases, we identify several stages of interest in KillerRed synthesis and activity as follows.
- mRNA synthesis and degradation
- KillerRed synthesis, maturation, and degradation
- KillerRed states: singlet (S), singlet excited (S^*), triplet excited (T^*), and deactivated (Da)
- Superoxide production (by KillerRed)
- Superoxide elimination (by superoxide dismutase)

We implemented these system stages with distinct model states, and outlined them in Figure 4.3, together with state variables (values are included if variables are fixed within a state), transitions between states, and events that trigger state transitions. In Table 4.1 we list the model states that are used to describe the stages of the system. In the following, we detail our implementation of system stages within the model. We also list equations that we derived for each stage.

Cell exposure to light
In [206], the authors describe a method for determining the rate coefficient of activation from

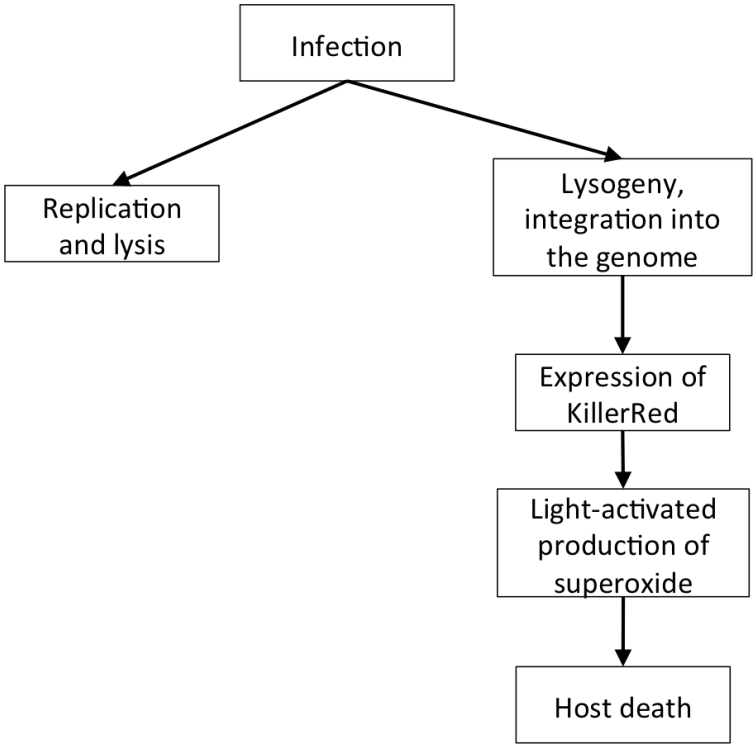


Figure 4.1: Interactions between phage and bacteria used in our model

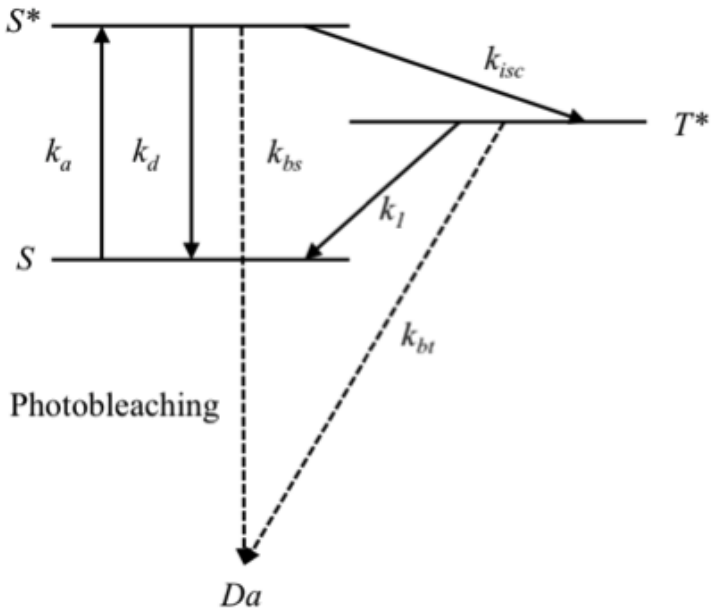


Figure 4.2: Energy diagram for a generic fluorochrome [198]

the ground state k_a: $k_a = \sigma I$. In detail, σ is the optical cross-section per molecule and I is the excitation intensity in photons per unit area. The lamp used for photobleaching gave $I = 1 \times 1027\ photons/cm^2 s$ (about $1W$). σ is given by $\sigma = \varepsilon(1000cm^3/L)(ln10)NA$, where ε is the extinction coefficient and NA is Avogadro's number. We calculated $k_a = 1.72 \times 1011 s^{-1}$ for KillerRed for our photobleaching experiments. The rate constant for returning to the ground state is $k_f = ln2/\tau$, where τ is the half-life for KillerRed in the excited state. τ for KillerRed is assumed to be similar to τ for dsRed (about $3.0ns$ [38]), since their chromophores are identical. Thus, by assuming that KillerRed is always in the excited state (if it has not been deactivated) during photobleaching, we have that $k_f = 2.3 \times 108 s^{-1}$ and $F = k_a/(k_a + k_f) = 0.9987$.

Production of superoxide

Production of the superoxide radical is governed by several reactions. Fluorescein is used as a model chromophore. S, S^*, T^*, and Da are the singlet, excited singlet, excited triplet, and deactivated states, respectively, of the chromophore. Figure 4.2 outlines transitions between different forms of the chromophore. In detail, fluorochrome molecules absorb photon energy at a rate k_a and go from the ground singlet state S up to the excited singlet state S^*. Then they may return to the ground state by radiative (fluorescence) or non-radiative (internal conversion) pathway at a combined rate k_d. They may also undergo non-radiative intersystem crossing, at a rate k_{isc}, to T^*, where they may return to the ground state at a rate k_1. Photobleaching may take place from both S^* and T^* at rates k_{bs} and k_{bt}, respectively. Those photobleached molecules can no longer participate in the excitation-emission cycle.

Superoxide dismutase

Superoxide dismutase is $E.\ coli's$ main defense against superoxide. Its action was incorporated using Michaelis-Menten kinetics:
$-\frac{d[O_2^{\cdot -}]}{dt} = \frac{V_{max}[O_2^{\cdot -}]}{K_m + [O_2^{\cdot -}]}$,
where V_{max} estimated using k_cat from [138], and K_m was estimated using k_m and k_{cat}/k_m from [84, 138].

Cell without λ-phage genome

The first system stage that we model is a bacteria cell that does not have phage genome injected, and gene transcription is not induced. Thus, all of the model elements are at their initial level, assumed to be 0. In the model, we assume that λ-phage genome is injected into bacteria cell with rate k_1, or t_1 time units after the start of time counting. When analyzing individual cells this does not have an effect, but is important to take into account when analyzing cell population.

Cell with injected λ-phage genome

After the injection of phage genome into the cell, the genome will be inserted into the bacterial DNA with rate k_2. Or, in terms of counting time units, it will take t_2 time units to integrate the phage genome into bacterial plasmid once it is inside the cell. However, since IPTG is still not added to the cell, we assume that gene transcription is not induced yet. Therefore, similar to previous two states, initial state and the state of phage genome injected, this state is assumed to be static.

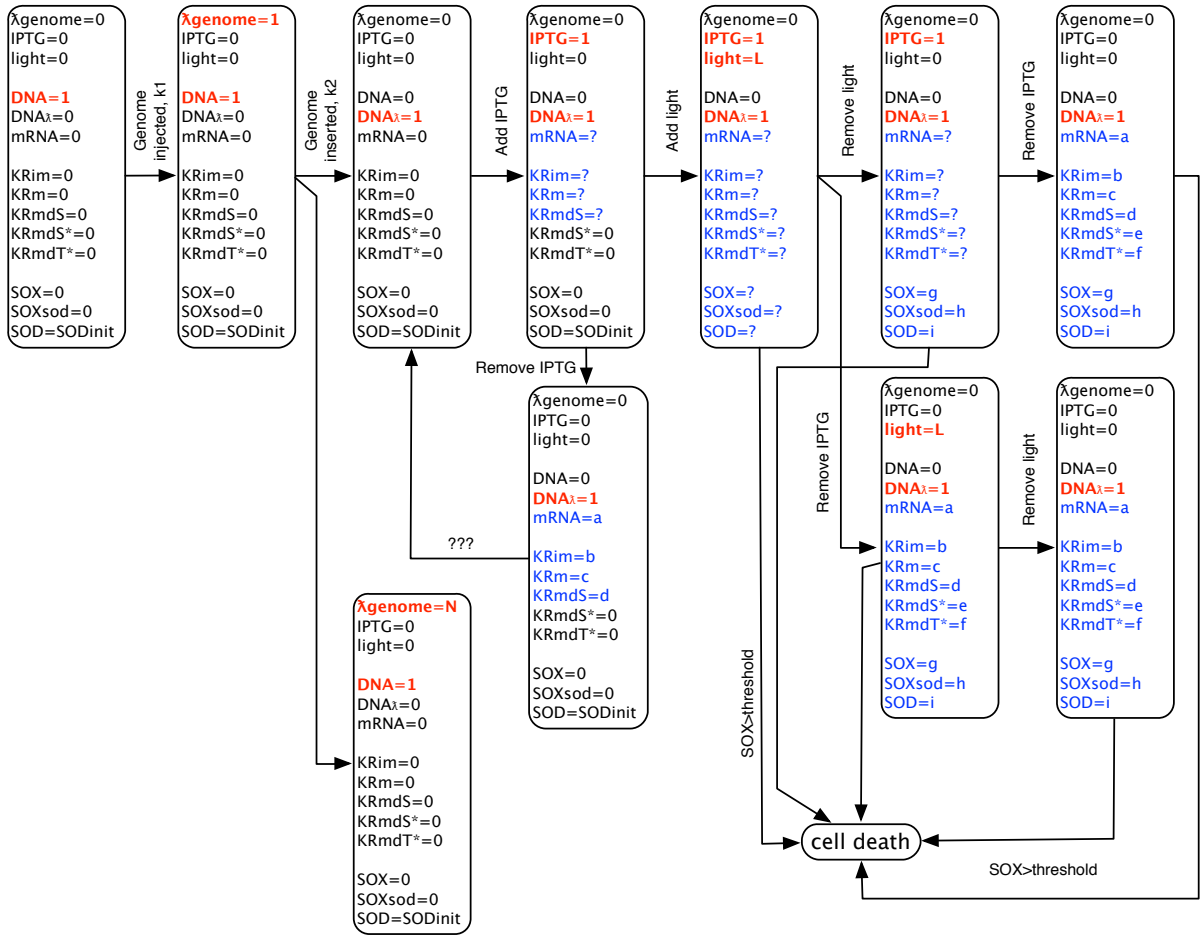


Figure 4.3: Hybrid automaton for our KillerRed model

Addition of IPTG

When IPTG is added to the system, the transcription starts. Measure of transcriptional efficiency is the rate of mRNA synthesis, k_{RNAsyn}. Our construct uses a wild-type lac promoter, so we assume that its transcription rates are similar to the lac operon. Next, mRNA transforms into immature KillerRed molecules with translational efficiency, k_{KRsyn}. The maximum translation rate in the model is three orders of magnitude lower to reflect the presents of several rare codons. This adjustment is suggested by comparisons of our fluorescence data for KillerRed and mRFP, which have nearly identical brightness.

Immature synthesized KillerRed (KR_{im}) requires additional time to become mature KillerRed form (KR_m), and to fold and create a dimer (KR_{md}). These two events together occur with the overall rate of k_{KR_m}. This folded and dimerized form of KillerRed that can be activated by light is called singlet form (KR_{mdS}). Degradations of synthesized mRNA and KillerRed in both modeled forms are included in equations with rates $k_{mRNAdeg}$ (characteristic half-life), $k_{KR_{im}deg}$, and k_{mdSdeg}, respectively. In this model state, resulting from addition of IPTG and ending with

either removal of IPTG or addition of light, we use the following ordinary differential equations (ODEs) to describe the continuous dynamics.

$$\begin{aligned}
\frac{\mathrm{d}[mRNA]}{\mathrm{d}t} &= k_{RNAsyn} \cdot [DNA] - k_{RNAdeg} \cdot [mRNA] \\
\frac{\mathrm{d}[KR_{im}]}{\mathrm{d}t} &= k_{KR_{im}syn} \cdot [mRNA] - (k_{KR_m} + k_{KR_{im}deg}) \\
&\quad \cdot [KR_{im}] \\
\frac{\mathrm{d}[KR_{mdS}]}{\mathrm{d}t} &= k_{KR_m} \cdot [KR_{im}] - k_{KR_{mdS}deg} \cdot [KR_{mdS}]
\end{aligned}$$

State	State description	Input	Next state(s)
S_0	Initial system state, bacteria cell, without phage	n/a	S_1 (ex.)
S_1	Phage genome injected	λ-phage genome	S_2 (in.), S_3 (in.)
S_2	Phage genome replication (lytic cycle)	Genome replication	n/a
S_3	Phage genome within bacterial DNA (lysogenic cycle)	Genome insertion	S_4 (ex.)
S_4	Gene transcription, translation	Addition of IPTG	S_5 (ex.), S_6 (ex.)
S_5	Gene transcription decrease	Removal of IPTG	S_3 (in.)
S_6	Activation of KillerRed	Light turned ON	S_7 (ex.), S_8 (ex.), S_{11} (in.)
S_7	Mixture of KillerRed forms, no activation	Light turned OFF	S_9 (ex.), S_{11} (in.)
S_8	Mixture of KillerRed forms, transcription decrease	Removal of IPTG	S_{10} (ex.), S_{11} (in.)
S_9	Mixture of KillerRed forms, no activation, transcription decrease	Removal of IPTG	S_{11} (in.)
S_{10}	Mixture of KillerRed forms, transcription decrease, no activation	Light turned OFF	S_{11} (in.)
S_{11}	Cell death	SOX>threshold	n/a

Table 4.1: List of modeled system states, their description, inputs and next state(s) with indication whether transition was triggered by external input (ex.) or by internal variable (in.) reaching some specified value.

Addition of light

Addition of light results in moving from the state with KillerRed synthesis into the state of activating KillerRed, S. In the state that assumes system's exposure to light, other forms of KillerRed are

present, including excited singlet state S^*, (KR_{mdS^*}) and triplet state T^*, (KR_{mdT^*}). Transitions between different forms of KillerRed can occur and therefore, in this state, we include the above model equations and modified equation, and add equations for other forms of KillerRed (KR_{mdS^*}, KR_{mdT^*}), as well as equations for produced superoxide (SOX) and for the effect of superoxide dismutase (SOX_{sod}).

$$\begin{aligned}
\frac{\mathrm{d}[KR_{mdS}]}{\mathrm{d}t} &= k_{KR_m} \cdot [KR_{im}] + k_{KR_f} \cdot [KR_{mdS^*}] \\
&\quad + k_{KR_{ic}} \cdot [KR_{mdS^*}] + k_{KR_{nrd}} \cdot [KR_{mdT^*}] \\
&\quad + k_{KR_{SOXd1}} \cdot [KR_{mdT^*}] - k_{KR_{ex}} \cdot [KR_{mdS}] \\
&\quad - k_{KR_{mdS}deg} \cdot [KR_{mdS}] \\
\\
\frac{\mathrm{d}[KR_{mdS^*}]}{\mathrm{d}t} &= k_{KR_{ex}} \cdot [KR_{mdS}] - k_{KR_f} \cdot [KR_{mdS^*}] \\
&\quad - k_{KR_{ic}} \cdot [KR_{mdS^*}] - k_{KR_{isc}} \cdot [KR_{mdS^*}] \\
&\quad - k_{KR_{mdS^*deg}} \cdot [KR_{mdS^*}] \\
\frac{\mathrm{d}[KR_{mdT^*}]}{\mathrm{d}t} &= k_{KR_{isc}} \cdot [KR_{mdS^*}] - k_{KR_{nrd}} \cdot [KR_{mdT^*}] \\
&\quad - k_{KR_{SOXd1}} \cdot [KR_{mdT^*}] \\
&\quad - k_{KR_{SOXd2}} \cdot [KR_{mdT^*}] \\
&\quad - k_{KR_{mdT^*deg}} \cdot [KR_{mdT^*}] \\
\frac{\mathrm{d}[SOX]}{\mathrm{d}t} &= k_{KR_{SOXd1}} \cdot [KR_{mdT^*}] + k_{KR_{SOXd2}} \\
&\quad \cdot [KR_{mdT^*}] - \frac{\mathrm{d}[SOX_{sod}]}{\mathrm{d}t} \\
\frac{\mathrm{d}[SOX_{sod}]}{\mathrm{d}t} &= k_{SOD} \cdot V_{maxSOD} \cdot \frac{[SOX]}{K_m + [SOX]}
\end{aligned}$$

Rates of KillerRed transitioning from state S^* to state S, through fluorescence or internal conversion are denoted with k_{KR_f} and $k_{KR_{ic}}$, respectively. Rates of KillerRed transitioning from state T^* to state S, through non-radiative deactivation or by production of SOX with deactivation are denoted with $k_{KR_{nrd}}$ and $k_{KR_{SOXd_1}}$, respectively. The excited form of KillerRed, S^*, is formed at rate $k_{KR_{ex}}$, and is reduced in several ways: (a) by fluorescence with rate k_{KR_f}, (b) by internal conversion with rate $k_{KR_{ic}}$, (c) by inter-system crossing $k_{KR_{isc}}$, and (d) by degradation with rate $k_{KR_{mdS^*deg}}$. The triplet form, T^*, is formed through intersystem crossing with rate $k_{KR_{isc}}$, and is reduced in several ways, by non-radiative deactivation with rate $k_{KR_{nrd}}$, by superoxide (ROS) production with deactivation to state S with rate $k_{KR_{SOXd_1}}$, by superoxide (ROS) production with photobleaching with rate $k_{KR_{SOXd_2}}$, and by degradation with rate $k_{KR_{mdS^*deg}}$. In addition, $k_{KR_{SOXd_1}}$ and $k_{KR_{SOXd_2}}$ can be computed taking into account relative propensity for KillerRed to generate superoxide without becoming deactivated (c), photo-bleaching rate obtained from experiments ($k_{KR_{pb}}$), and quantum yield (Φ) as follows.

$k_{KR_{SOXd_1}} = c \cdot \frac{k_{KR_{pb}}}{\Phi}$ $\qquad$ $k_{KR_{SOXd_2}} = \frac{k_{KR_{pb}}}{\Phi}$

4.2 δ-Decisions for Hybrid Models

To validate the correctness, estimate parameters, and conduct sensitivity analysis of our model, we constructed a hybrid model for the system, and used delta-complete decision procedures [94, 95] to find solutions to these formulae. Before going over the delta-complete decision procedures, we first give the formal definition of hybrid automata.

Definition 4.2.1 (Hybrid Automaton) *A hybrid automaton H consists of the following components.*

- **Variables**. *A finite set $X = \{x_1, \cdots, x_n\}$ of real-numbered variables, where n is the dimension of H. We write $\dot{X}$ for the set $\{\dot{x_1}, \cdots, \dot{x_n}\}$ to represent first derivatives of variables during the continuous change, and write X' for the set $\{x'_1, \cdots, x'_n\}$ to denote values of variables at the conclusion of the discrete change.*
- **Control graph**. *A finite directed multigraph (V, E). The vertices in V are called control modes, and edges in E are control switches.*
- **Initial, invariant, and flow conditions**. *As vertex labeling functions over each control mode $v \in V$, the initial condition $init(v)$ is predicate whose free variables are from V, the invariant condition $inv(v)$ is a predicate whose free variables are from X, and the flow condition $flow(v)$ is a predicate whose free variables are from $X \cup \dot{X}$.*
- **Jump conditions**. *An edge labeling function $jump$ that assigns to each control switch $e \in E$ a predicate whose free variables are from $X \cup X'$.*
- **Events**. *A finite set Σ of events, and an edge labeling function $event : E \to \Sigma$ that assigns to each control switch an event.*

Formal verification of hybrid systems is crucially very important and challenging. Systems combining nonlinear dynamics and nontrivial discrete control can hardly be handled. In order to overcome the undecidability of reasoning about hybrid systems, Gao *et al.* recently defined the concept of δ-satisfiability over the reals, and presented a corresponding δ-complete decision procedure [94, 95]. The main idea is to decide correctly whether slightly *relaxed* sentences over the reals are satisfiable or not. The following definitions are from [95].

Definition 4.2.2 (Bounded Quantifier) *A bounded quantifier is one of the following:*

$$\exists^{[a,b]} x = \exists x : (a \leq x \wedge x \leq b)$$
$$\forall^{[a,b]} x = \forall x : (a \leq x \wedge x \leq b)$$

Definition 4.2.3 (Bounded Σ_1 Sentence) *A bounded Σ_1 sentence is an expression of the form:*

$$\exists^{I_1} x_1, ..., \exists^{I_1} x_n : \psi(x_1, ..., x_n)$$

where $I_i = [a_i, b_i]$ are intervals, $\psi(x_1, ..., x_n)$ is a Boolean combination of atomic formulas of the form $g(x_1, ..., x_n) \circ p\, 0$, where g is a composition of Type 2-computable functions and $\circ p \in \{<, \leq, >, \geq, =, \neq\}$.

Note that any bounded Σ_1 sentence is equivalent to a Σ_1 sentence in which all the atoms are of the form $f(x_1, ..., x_n) = 0$ (*i.e.*, the only $\circ$p needed is '='). Essentially, Type 2-computable functions can be approximated arbitrarily well by finite computations of a special kind of Turing machines (Type 2 machines); most 'useful' functions over the reals are Type 2-computable. The notion of δ-weakening of a bounded sentence is central to δ-satisfiability.

Definition 4.2.4 (δ-*Weakening*) *Let $\delta \in \mathbb{Q}^+ \cup \{0\}$ be a constant and ϕ a bounded Σ_1-sentence in the standard form*

$$\phi = \exists^{I_1} x_1, ..., \exists^{I_n} x_n : \bigwedge_{i=1}^{m} (\bigvee_{j=1}^{k_i} f_{ij}(x_1, ..., x_n) = 0) \tag{4.1}$$

*where $f_{ij}(x_1, ..., x_n) = 0$ are atomic formulas. The δ-*weakening *of ϕ is the formula:*

$$\phi^\delta = \exists^{I_1} x_1, ..., \exists^{I_n} x_n : \bigwedge_{i=1}^{m} (\bigvee_{j=1}^{k_i} |f_{ij}(x_1, ..., x_n)| \leq \delta) \tag{4.2}$$

Note that ϕ implies ϕ^δ, while the converse is obviously not true. The bounded δ-satisfiability problem asks for the following: given a sentence of the form (4.1) and $\delta \in \mathbb{Q}^+$, correctly decide whether **unsat** (ϕ is false), or δ**-sat** (ϕ^δ is true). If the two cases overlap either decision can be returned: such a scenario reveals that the formula is *fragile* - a small perturbation (*i.e.*, a small δ) can change the formula's truth value.

A qualitative property of hybrid systems that can be checked is bounded δ-reachability. It asks whether the system reaches the unsafe region after $k \in \mathbb{N}$ discrete transitions.

Definition 4.2.5 (Bounded k-Step δ-Reachability) *Bounded k step δ-reachability in hybrid systems can be encoded as a bounded Σ_1-sentence*

$$\begin{aligned} \exists \boldsymbol{x}^0_{0,q_0}, \exists \boldsymbol{x}^t_{0,q_0}, ..., \exists \boldsymbol{x}^0_{0,q_m}, \exists \boldsymbol{x}^t_{0,q_m}, ..., \exists \boldsymbol{x}^0_{k,q_m}, \exists \boldsymbol{x}^t_{k,q_m} : \\ (\bigvee_{q \in Q} (\mathit{init}_q(\boldsymbol{x}^0_{0,q}) \wedge \mathit{flow}_q(\boldsymbol{x}^0_{0,q}, \boldsymbol{x}^t_{0,q}))) \\ \wedge (\bigwedge_{i=0}^{k-1} (\bigvee_{q,q' \in Q} (\mathit{jump}_{q \to q'}(\boldsymbol{x}^t_{i,q}, \boldsymbol{x}^0_{i+1,q'}) \\ \wedge (\mathit{flow}_{q'}(\boldsymbol{x}^0_{i+1,q'}, \boldsymbol{x}^t_{i+1,q'}))) \wedge (\bigvee_{q \in Q} \mathit{unsafe}_q(\boldsymbol{x}^t_{k,q})))) \end{aligned} \tag{4.3}$$

where $\boldsymbol{x}^0_{i,q}$ and $\boldsymbol{x}_{i,q}$ represent the continuous state in the mode q at the depth i, and q' is a successor mode.

Intuitively, the formula above can be understood as follows: the first conjunction is asking for a set of continuous variables which satisfy the initial condition in one of the modes and the flow in that mode; the second conjunction is looking for a set of vectors which satisfy any k discrete jumps and flows in each successor mode defined by the jumps; the third conjunction is verifying whether the state of the system (the mode and the set of continuous variables in the mode after k jumps) belongs to the unsafe region. Note that the previous definition asks for reachability in *exactly* k steps. One can build a disjunction of formula (4.3) for all values from 1 to k, thereby obtaining reachability *within* k steps.

The δ-reachability problem can be solved using the described δ-complete decision procedure, which will correctly return one of the following answers:
- **unsat**: the system never reaches the bad region U,
- δ**-sat**: the δ-perturbation of (4.3) is true, and a witness, *i.e.*, an assignment for all the variables, is returned.

We now show that this δ-decisions technique for hybrid models can be used to handle problems such as model falsification, parameter estimation, and parametric sensitivity analysis.

Model Falsification. The model falsification problem with existing experimental observations is basically a bounded reachability question: Expressing each experimental observation as a goal region, is there any number of steps k in which the model reaches the goal region? If none exists, the model is incorrect regarding the given observation. If, for each observation, a witness is returned, we can conclude that the model is correct with regard to a given set of experimental results. This is a bounded Model Checking problem, where all experimental observations can be expressed as reachability properties.

Parameter Estimation. The parameter estimation problem can also be encoded as a k-step reachability problem: Does it exist a parameter combination for which the model reaches the given goal region in k steps? Considering an assignment of a certain set of system parameters, if a witness is returned, this assignment is potentially a good estimation for those parameters. The goal here is to find an assignment with which all the given goal regions can be reached in bounded steps.

Parametric Sensitivity Analysis. The sensitivity analysis can be conducted by a set of bounded reachability queries as well. For different possible values of a certain system parameter, are the results of reachability analysis the same? If so, the model is insensitive to this parameter with regard to the given experimental observations.

4.3 Results and Discussion

Effect of delay in turning light ON

First, we have studied the relation between the time to turn ON the light after adding IPTG that is a molecular biology reagent used to induce protein expression ($t_{lightON}$), and the total time needed until the bacteria cells being killed (t_{total}). We fixed the values of several other parameters as follows.

- SOX_{thres} = 5e-4m - threshold for the concentration level of SOX which is sufficient to kill the bacteria cells
- $t_{lightOFF_1}$ = 2 hours (hrs) - time to turn the light OFF after turning it ON
- $t_{lightOFF_2}$ = 2 hrs - time to turn the light OFF after removing IPTG
- t_1 = 1 hr - time to inject genome
- t_2 = 1 hr - time to insert genome into DNA after injecting it into bacteria cell
- $t_{addIPTG_3}$ = 1 hr - time to add IPTG after inserting phage genome into bacteria DNA

As shown in the first two rows of Table 4.2, the earlier we turn on the light after adding IPTG, the quicker the bacteria cells will be killed.

Lower bound for the duration of exposure to light

The δ-decisions technique has also been adopted to analyze the impact of the time duration that the cells are exposed to light ($t_{lightOFF_1}$) on the system, and estimate an appropriate range for $t_{lightOFF_1}$ which leads to the successful killing of bacteria cells by KillerRed. By setting SOX_{thres}, $t_{lightOFF_2}$, t_1, t_2, and $t_{addIPTG_3}$ with the same values in Section 4.3, and assigning 2 hr to $t_{lightON}$ (time to turn the light OFF after turning it ON), we have found that, in order to kill bacteria cells, the system has to keep the light ON for at least 4 hours (see row 3-4 of Table 4.2).In addition, we have also found that the bacteria cells can be killed within 100 hours when light is ON for 4 hours.

$t_{lightON}$ (hr)	1	2	3	4	5	6	7	8	9	10
t_{total} (hr)	16	17.2	18.5	20	21.3	22.7	23.5	24.1	25	30
$t_{lightOFF_1}$ (hr)	1	2	3	4	5	6	7	8	9	10
killed bacteria cells	failed	failed	failed	succ	succ	succ	succ	succ	succ	succ
t_{rmIPTG_3} (hr)	1	2	3	4	5	6	7	8	9	10
killed bacteria cells	succ	succ	succ	succ	succ	succ	succ	succ	succ	succ
SOX_{thres} (M)	1e-4	2e-4	3e-4	4e-4	5e-4	6e-4	7e-4	8e-4	9e-4	1e-3
t_{total} (hr)	5.1	5.2	5.4	17	19	48	61	71	36	42

Table 4.2: Formal analysis results for our KillerRed hybrid model

Time to remove IPTG as an insensitive role

The sensitivity of the time difference between removing the light and removing IPTG (t_{rmIPTG_3}) with regard to the successful killing of bacteria cells has also been studied. We have noticed that t_{rmIPTG_3} has insignificant impacts on the cell killing outcome (see row 5-6 of Table 4.2). This is in accordance with our understanding of this system, since any additional KillerRed that will be synthesized will not be activated in the absence of light. Note that, for other involved system parameters, we used the same values for SOX_{thres}, $t_{lightON}$, $t_{lightOFF_2}$, t_1, t_2, and $t_{addIPTG_3}$ as in Section 4.3, and set $t_{lightOFF_1}$ as 4 hours.

Necessary level of superoxide

Finally, we have used the δ-decisions to discuss the correctness of our hybrid model by considering various values of SOX_{thres} within the suggested range - [100uM, 1mM]. We have used the same values for variables SOX_{thres}, $t_{lightON}$, $t_{lightOFF_1}$, $t_{lightOFF_2}$, t_1, t_2, and $t_{addIPTG_3}$ as in Section 4.3. As we can see from row 7-8 of Table 4.2, the bacteria cells can be killed in reasonable time for all 10 point values of SOX_{thres}, which was uniformly chosen from [100uM, 1mM]. Furthermore, we have also found a broader range for SOX_{thres} up to 0.6667M, with which bacteria cells can be killed by KillerRed.

Chapter 5

Biological Systems as Stochastic Hybrid Models and *SReach*

As mentioned in the introduction chapter, stochastic hybrid systems (SHSs) are formal models that tightly combine discrete, continuous, and stochastic components. One important question for the quantitative analysis of SHSs is the probabilistic reachability problem, considering that many verification problems can be reduced to reachability problems. It is to compute the probability of reaching a certain set of states. This problem is no longer a decision problem, as it generalizes that by asking what is the probability that the system reaches the target region. For SHSs with both stochastic and non-deterministic behavior, the problem results in general in a range of probabilities, thereby becoming an optimization problem.

In this chapter, we describe our tool *SReach* which supports probabilistic bounded δ-reachability analysis for two model classes: hybrid automata (HAs) [117] with parametric uncertainty, and probabilistic hybrid automata (PHAs) [200] with additional randomness. (Note that, in the following, we use notations - HA_p and PHA_r - for these two model classes respectively.) Our method combines the recently proposed δ-complete bounded reachability analysis technique [147] with statistical testing techniques. *SReach* saves the virtues of the Satisfiability Modulo Theories (SMT) based Bounded Model Checking (BMC) for HAs [70, 205], namely the fully symbolic treatment of hybrid state spaces, while advancing the reasoning power to probabilistic models. Furthermore, by utilizing the δ-complete analysis method, the full non-determinism of models will be considered. The coverage of simulation will be increased, as the δ-complete analysis method results in an over-approximation of the reachable set, whereas simulation is only an under-approximation of it. The zero-crossing problem can be avoided as, if a zero-crossing point exists, it will always return an interval containing it. By using statistical tests, *SReach* can place controllable error bounds on the estimated probabilities. We discuss three biological models - an atrial fibrillation model, a prostate cancer treatment model, and our synthesized Killerred biological model - to show that *SReach* can answer questions including model validation/falsification, parameter synthesis, and sensitivity analysis.

5.1 Stochastic Hybrid Models

Before introducing the algorithm implemented by *SReach* and the problems that it can handle, we first define two model classes that *SReach* considers formally. For HA$_p$s, we follow the definition of HAs in [117], and extend it to consider probabilistic parameters in the following way.

Definition 5.1.1 (HA$_p$) *A hybrid automaton with parametric uncertainty is a tuple* $H_p = \langle (Q, E), V, RV, \mathsf{Init}, \mathsf{Flow}, \mathsf{Inv}, \mathsf{Jump}, \Sigma \rangle$*, where*

- *The vertices* $Q = \{q_1, \cdots, q_m\}$ *is a finite set of discrete modes, and edges in* E *are control switches.*
- $V = \{v_1, \cdots, v_n\}$ *denotes a finite set of real-valued system variables. We write* $\dot{V}$ *to represent the first derivatives of variables during the continuous change, and write* V' *to denote values of variables at the conclusion of the discrete change.*
- $RV = \{w_1, \cdots, w_k\}$ *is a finite set of independent random variables, where the distribution of* w_i *is denoted by* P_i*.*
- Init*,* Flow*, and* Inv *are labeling functions over* Q*. For each mode* $q \in Q$*, the initial condition* $Init(q)$ *and invariant condition* $Inv(q)$ *are predicates whose free variables are from* $V \cup RV$*, and the flow condition* $Flow(q)$ *is a predicate whose free variables are from* $V \cup \dot{V} \cup RV$*.*
- Jump *is a transition labeling function that assigns to each transition* $e \in E$ *a predicate whose free variables are from* $V \cup V' \cup RV$*.*
- Σ *is a finite set of events, and an edge labeling function* $event : E \to \Sigma$ *assigns to each control switch an event.*

Another class is PHA$_r$s, which extend HAs with discrete probability transitions and additional randomness for transition probabilities and variable resets. In detail, for discrete transitions, instead of making a purely (non)deterministic choice over the set of currently enabled jumps, a PHA$_r$ (non)deterministically chooses among the set of recently enabled discrete probability distributions, each of which is defined over a set of transitions whose probabilities can be uncertain.

Definition 5.1.2 (PHA$_r$) *A probabilistic hybrid automaton with additional randomness* H_r *consists of* Q, E, V, RV, Init, Flow, Inv, Σ *as in Definition 5.1.1, and* $Cmds$*, which is a finite set of probabilistic guarded commands of the form:*
$g \rightarrow p_1 : u_1 + \cdots + p_m : u_m$*,*
where g *is a predicate representing a transition guard with free variables from* V*,* p_i *is the transition probability for the* i*th probabilistic choice which can be expressed by an equation involving random variable(s) in* RV *and the* p_i*'s satisfy* $\sum_{i=1}^{m} p_i = 1$*, and* u_i *is the corresponding transition updating function for the* i*th probabilistic choice, whose free variables are from* $V \cup V' \cup RV$*.*

To illustrate the additional randomness allowed for transition probabilities and variable resets, an example probabilistic guarded command is $x \geq 5 \ \rightarrow \ p_1 : (x' = sin(x)) + (1 - p_1) : (x' = p_x)$, where x is a system variable, p_1 has a Uniform distribution $U(0.2, 0.9)$, and p_x has a Bernoulli distribution $B(0.85)$. This means that, the probability to choose the first transition is not a fixed value, but a random one having a Uniform distribution. Also, after taking the second transition, x can be assigned to either 1 with probability 0.85, or 0 with 0.15. In general, for an individual probabilistic guarded command, the transition probabilities can be expressed by equations of one or more new random variables, as long as values of all transition probabilities are within $[0, 1]$, and their sum is 1. Currently, all four primary arithmetic operations are supported. Note that, to preserve the Markov property, only unused random variables can be used, so that no dependence between the current probabilistic jump and previous transitions will be introduced.

5.2 The *SReach* Algorithm

A recently proposed δ-complete decision procedure [94] relaxes the reachability problem for HAs in a sound manner: it verifies a conservative approximation of the system behavior, so that bugs will always be detected. The over-approximation can be tight (tunable by an arbitrarily small rational parameter δ), and a false alarm with a small δ may indicate that the system is fragile, thereby providing valuable information to the system designer. (See Chapter 4.2 for more details about the δ-complete decision procedure.) We now define the probabilistic bounded δ-reachability problem based on the bounded δ-reachability problem defined in [147] .

Definition 5.2.1 *The probabilistic bounded k step δ-reachability for a HA$_p$ H_p is to compute the probability that H_p reaches the target region T in k steps. Given the set of independent random variables* $\mathbf{r}$*, $Pr(\mathbf{r})$ a probability measure over* $\mathbf{r}$*, and Ω the sample space of* $\mathbf{r}$*, the reachability probability is $\int_\Omega I_T(\mathbf{r}) dPr(\mathbf{r})$, where $I_T(\mathbf{r})$ is the indicator function which is 1 if H_p with* $\mathbf{r}$ *reaches T in k steps.*

Definition 5.2.2 *For a PHA$_r$ H_r, the probabilistic bounded k step δ-reachability estimated by SReach is the maximal probability that H_r reaches the target region T in k steps: $max_{\sigma \in E} Pr^k_{H_r,\sigma,T}(i)$, where E is the set of possible executions of H starting from the initial state i, and σ is an execution in the set E.*

As shown in Figure 5.1, given a stochastic hybrid system, after encoding uncertainties using random variables, *SReach* samples them according to the given distributions. For each sample, a corresponding intermediate HA is generated by replacing random variables with their assigned values. Then, the δ-complete analyzer *dReach* is utilized to analyze each intermediate HA M_i, together with the desired precision δ and unfolding depth k. The analyzer returns either unsat or δ-sat for M_i. This information is then used by a chosen statistical testing procedure to decide whether to stop or to repeat the procedure, and to return the estimated probability. The full procedure is illustrated in Algorithm 3, where MP is a given stochastic model, and ST indicates which

Algorithm 3 SReach

function SREACH(MP, ST, δ, k)
  if MP is a HA$_p$ **then**
    $MP \leftarrow EncRM_1(MP)$ ▷ encode uncertain system parameters
  else ▷ otherwise a PHA$_r$
    $MP \leftarrow EncRM_2(MP)$ ▷ encode probabilistic jumps and extra randomness
  end if
  $Succ, N \leftarrow 0$ ▷ number of δ-sat samples and total samples
  $Assgn \leftarrow \emptyset$ ▷ record unique sampling assignments and dReach results
  $RV \leftarrow \text{ExtractRV}(MP)$ ▷ get the RVs from the probabilistic model
  repeat in parallel
    $S_i \leftarrow \text{Sim}(RV)$ ▷ sample the parameters
    if $S_i \in Assgn.sample$ **then**
      $Res \leftarrow Assgn(S_i).res$ ▷ no need to call dReach
    else
      $M_i \leftarrow \text{Gen}(MP, S_i)$ ▷ generate a dReach model
      $Res \leftarrow \text{dReach}(M_i, \delta, k)$ ▷ call dReach to solve k-step δ-reachability
    end if
    if Res = δ-sat **then** $Succ \leftarrow Succ + 1$
    end if
    $N \leftarrow N + 1$
  until $ST.done(Succ, N)$ ▷ perform statistical test
  return $ST.output$
end function

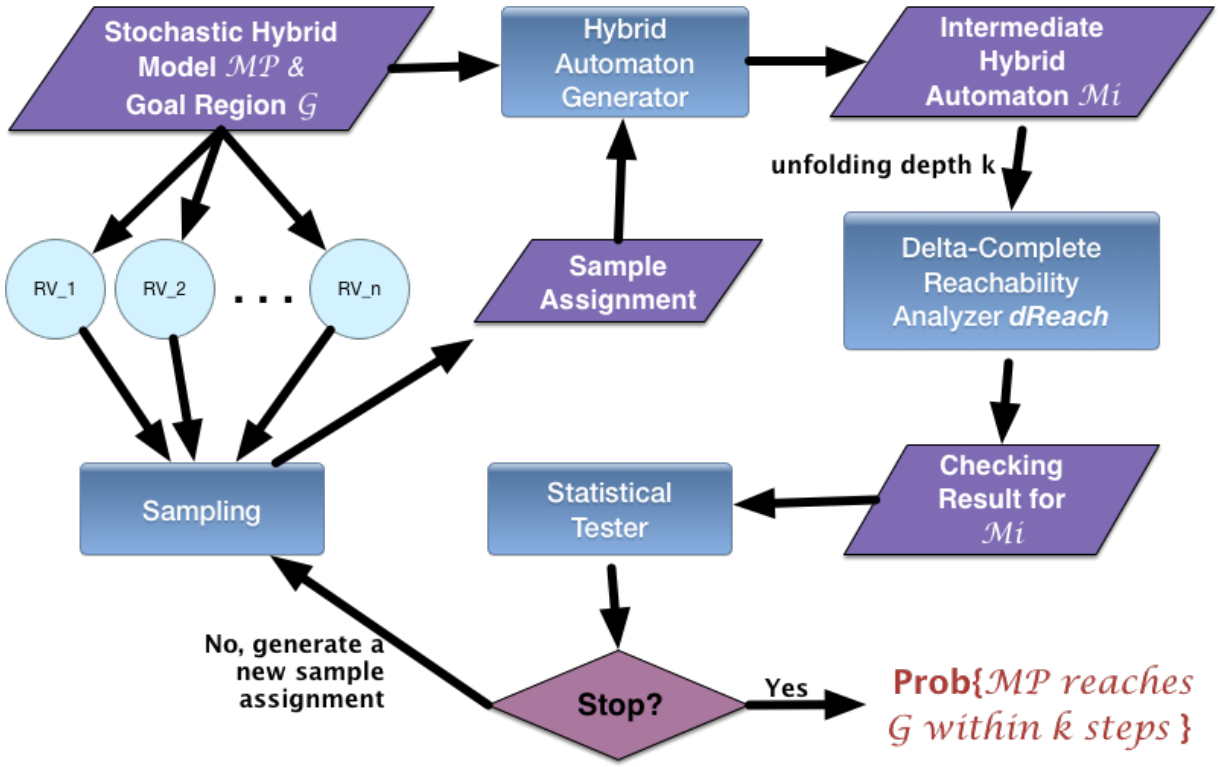


Figure 5.1: The framework of SReach algorithm

statistical testing method will be used. $Succ$ and N are used to record the number of δ-sat instances and total samples generated so far respectively, and are then the inputs of ST. Note that, for a PHA$_r$, sampling and fixing the choices of all the probabilistic transitions in advance results in an over-approximation of the original PHA$_r$, where safety properties are preserved. To promise a tight over-approximation and correctness of estimated probabilities, *SReach* supports PHA$_r$s with no or subtle non-determinism. That is, in order to offer a reasonable estimation, for PHA$_r$s, *SReach* is supposed to be used on models with no or few non-deterministic transitions, or where dynamic interleaving between non-deterministic and probabilistic choices are not important.

To improve the performance of *SReach*, each sampled assignment and its corresponding *dReach* result are recorded for avoiding redundant calls to *dReach*. This significantly reduces the total calls for PHA$_r$s, as the size of the sample space involving random variables describing probabilistic jumps is comparatively small. For the example PHA (as shown in Figure 5.2), with this heuristic, the total checking time has been decreased from 11291.31s for 658 samples (17.16s per sample) to 3295.82s (5.01s per sample). Furthermore, a parallel version of *SReach* has been implemented using OpenMP, where multiple samples and corresponding HAs are generated, and passed to *dReach* simultaneously. Using this parallel *SReach* on a 4-core machine, the running time for the example PHA has been further decreased to $2119.55s$ for 660 samples (3.33s per sample).

Currently, *SReach* supports a number of hypothesis testing methods - Lai's test [151], Bayes factor test [139], Bayes factor test with indifference region [229], and Sequential probability ratio test (SPRT)[215], and statistical estimation techniques - Chernoff-Hoeffding bound [121], Bayesian Interval Estimation with Beta prior[234], and Direct Sampling. All methods, as listed in the following, produce answers that are correct up to a precision that can be set arbitrarily by the user.

Lai's test [151]. As a simple class of sequential tests, it tests the one-sided composite hypotheses $H_0 : \ \theta \leq \theta_0$ versus $H_1 : \ \theta \geq \theta_1$ for the natural parameter θ of an exponential family of distributions under the $0-1$ loss and cost c per observation. [151] shows that these tests have nearly optimal frequentist properties and also provide approximate Bayes solutions with respect to a large class of priors.

Bayes factor test [139]. The use of Bayes factors is a Bayesian alternative to classical hypothesis testing. It is based on the Bayes theorem. Hypothesis testing with Bayes factors is more robust than frequentist hypothesis testing, as the Bayesian form avoids model selection bias, evaluates evidence in favor of the null hypothesis, includes model uncertainty, and allows non-nested models to be compared. Also, frequentist significance tests become biased in favor of rejecting the null hypothesis with sufficiently large sample size.

Bayes factor test with indifference region. A hypothesis test has ideal performance if the probability of the Type-I error (respectively, Type-II error) is exactly α (respectively, β). However, these requirements make it impossible to ensure a low probability for both types of errors simultaneously (see [229] for details). A solution is to use an indifference region. The indifference region indicates the distance between two hypotheses, which is set to separate the two hypotheses.

Sequential probability ratio test (SPRT) [215]. The SPRT considers a simple hypothesis H_0 :

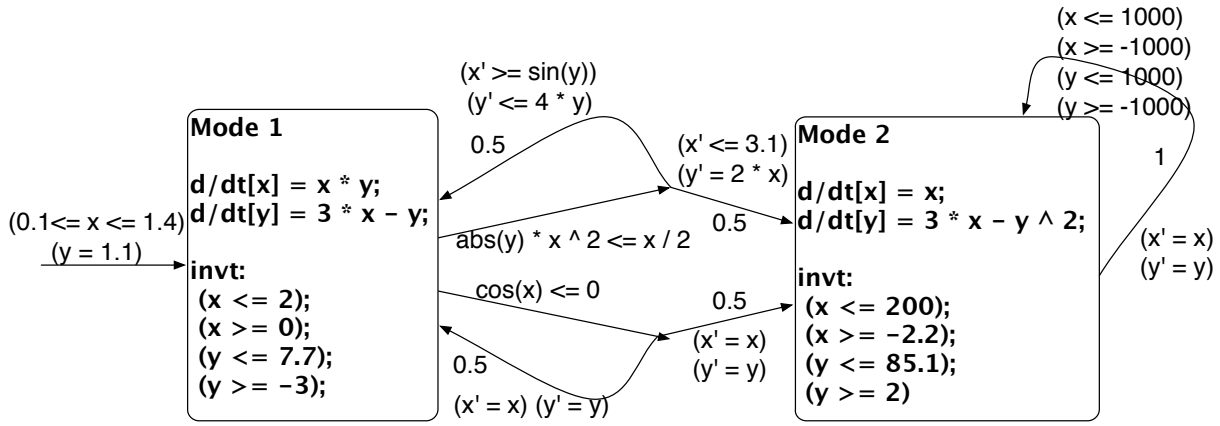


Figure 5.2: An example probabilistic hybrid automaton

$\theta = \theta_0$ against a simple alternative $H_1 : \ \theta = \theta_1$. With the critical region Λ_n and two thresholds A, and B, SPRT decides that H_0 is true and stops when $\Lambda_n < A$. It decides that H_1 is true and terminates if $\Lambda_n > B$. If $A \ < \Lambda_n < B$, it will collect another observation to obtain a new critical region Λ_{n+1}. The SPRT is optimal, among all sequential tests, in the sense that it minimizes the average sample size.

Chernoff-Hoeffding bound [121]. To estimate the mean p of a (bounded) random variable, given a precision δ' and coverage probability α, the Chernoff-Hoeffding bound computes a value p' such that $|p' \ - \ p| \leq \delta'$ with probability at least α.

Bayesian Interval Estimation with Beta prior [234]. This method estimates p, the unknown probability that a random sampled model satisfies a specified reachability property. The estimate will be in the form of a confidence interval, containing p with an arbitrary high probability. [234] assumes that the unknown p is given by a random variable, whose density is called the prior density, and focuses on Beta priors.

Direct sampling. Given N as the number of samples to be sampled, the direct sampling method estimates the mean of p of a (bounded) random variable. According to the central limit theorem [78], the error ϵ with a confidence c between the real probability p and the estimated $\hat{p}$ is bounded: $\epsilon = \phi^{-1}\left(\frac{c+1}{2}\right)\sqrt{\frac{p(1-p)}{N}}$
where $\phi(x) = \frac{1}{\sqrt{2\pi}}\int_{-x}^{x} e^{-t^2/2}dt$. That is, as N goes to ∞, the estimated probability approaches to the real one.

With these hypothesis testing methods, *SReach* can answer qualitative questions, such as "Does the model satisfy a given reachability property in k steps with probability greater than a certain threshold?" With the above statistical estimation techniques, *SReach* can offer answers to quantitative problems. For instance, "What is the probability that the model satisfies a given reachability property in k steps?" *SReach* can also handle additional types of interesting problems by encoding them as probabilistic bounded reachability problems. The **model validation/falsification** problem with prior knowledge can be encoded as a probabilistic bounded reachability question. After ex-

pressing prior knowledge about the given model as reachability properties, is there any number of steps k in which the model satisfies a given property with a desirable probability? If none exists, the model is incorrect regarding the given prior knowledge. The **parameter synthesis** problem can also be encoded as a probabilistic k-step reachability problem. Does there exist a parameter combination for which the model reaches the given goal region in k steps with a desirable probability? If so, this parameter combination is potentially a good estimation for the system parameters. The goal here is to find a combination with which all the given goal regions can be reached in a bounded number of steps. Moreover, **sensitivity analysis** can be conducted by a set of probabilistic bounded reachability queries as well: Are the results of reachability analysis the same for different possible values of a certain system parameter? If so, the model is insensitive to this parameter with regard to the given prior knowledge.

5.3 The *SReach* Tool

Input Format

The inputs to our *SReach* tool are descriptions of (probabilistic) hybrid automata with random variables (representing the probabilistic system parameters, and probabilistic jumps), and the reachability property to be checked. Following roughly the same format as the above definition of (probabilistic) hybrid automata, and adding the declarations of random variables, the description of an automaton is as follows.

Preprocessor. We can use the C language syntax to define constants and macros.

Variable declaration. For a random variable, the declaration specifies its distribution and name. Variables that are not random variables are required to be declared within bounds.

(Probabilistic) Hybrid automaton. A (probabilistic) hybrid automaton is represented by a set of modes. Within each mode declaration, we can specify statements for the mode invariant(s), flow function(s), and (probabilistic) jump condition(s). For a mode invariant, we can give any logic formula of the variables. A flow function is expressed by an ODE. As for a nonprobabilistic jump condition, it is written as

```
<logic_formula1>  ==>  @<target_mode>  <logic_formula2>,
```

where the first logic formula is given as the guard of the jump, and the second one specifies the reset condition after the jump. While for a probabilistic jump condition, we need an extra constraint to express the stochastic choice, which is of the following form

```
(and <logic_formula1> <stochastic choice>)  ==>
                              @<target_mode>  <logic_formula2>,
```

where the stochastic choice is a formula indicating which probabilistic transition will be chosen for this jump.

Initial conditions and Goals. Following the declaration of modes, we can declare one initial mode with corresponding conditions, and the reachability properties in the end.

Example 5.1. The following is an example input file for a hybrid automaton with parametric uncertainty. Currently, users can specify random variables (representing certain system parameters) with Bernoulli distribution (B), Uniform distribution (U), Gaussian distribution (N), Exponential distribution (E), and general Discrete distribution with given possible values and corresponding probabilities (DD).

```
#define pi 3.1416
N(1,0.1) mu1;
U(10,15) thro;
E(0.49) theta1;
B(0.75) xinit;
DD(0:0.7, 1:0.3) mu2;
[0,5] x;
[0,3] time;
{ mode 1;
  invt:
         (x<=1.5);
         (x>=0);
  flow:
         d/dt[x]=thro*(1/(theta1*sqrt(2*pi)))
                  *exp(0-((x-mu1+mu2)^2)/(2*theta1^2));
  jump:
        (x>=(thre1+5))==>@2(x'=x);
}
init:
@1      (x=xinit);
goal:
@4      (x>=50);
```

Example 5.2. This example demonstrates the format of the input file for a probabilistic hybrid automaton with additional randomness for transition probabilities. Note that, unlike the notations of declarations of random variables representing system parameters and probabilistic transitions, declarations of random variables used to express the additional randomness for jump probabilities start with a prefix j.

```
jU(0.7, 0.9) pjumprv;
DD(1:pjumprv, 2:(1 - pjumprv)) pjump1;
DD(1:0.3, 2:0.7) pjump2;
[-1000, 1000] x;
[-1000, 1000] y;
[0, 3] time;

{ mode 1;

```

```
10    invt:
11          (x <= 2);
12          (x >= 0);
13          (y <= 7.7);
14          (y >= -3);
15    flow:
16          d/dt[x] = x * y;
17          d/dt[y] = 3 * x - y;
18    jump:
19          (and (abs(y) * x ^ 2 <= x / 2) (pjump1 = 1)) ==> @1 (and (x' >=
                sin(y)) (y' <= 4 * y));
20          (and (abs(y) * x ^ 2 <= x / 2) (pjump1 = 2)) ==> @2 (and (x' <=
                3.1) (y' = 2 * x));
21          (and (cos(x) <= 0) (pjump2 = 1)) ==> @2 (and (x' = x) (y' = y))
               ;
22          (and (cos(x) <= 0) (pjump2 = 2)) ==> @1 (and (x' = x) (y' = y))
               ;
23  }
24
25  {
26    mode 2;
27    invt:
28          (x <= 200);
29          (x >= -2.2);
30          (y <= 85.1);
31          (y >= 2);
32    flow:
33          d/dt[x] = x;
34          d/dt[y] = 3 * x - y ^ 2;
35    jump:
36          (and (x <= 1000) (x >= -1000) (y <= 1000) (y >= -1000)) ==> @2
               (and (x' = x) (y' = y));
37  }
38  init:
39  @1      (and (x >= 0.1) (x <= 1.4) (y = 1.1));
40
41  goal:
42  @2      (and (x >= -10) (y >= -10));
```

Command Line

SReach offers two choices. It can be run **sequentially** by typing

```
sreach_sq <statistical_testing_option> <filename>
                                  <dReach> <k> <delta>,
```

or **in parallel** by

```
sreach_para <statistical_testing_option> <filename>
                                  <dReach> <k> <delta>,
```

where:

- `statistical_testing_option` is a text file containing a sequence of test specifications. We will introduce the usages of statistical testing options in the following part;
- `filename` is a .pdrh file describing the model of a hybrid system with probabilistic system parameters. It is of the input format described in last sub-section;
- `dReach` is a tool for bounded reachability analysis of hybrid systems based on dReal;
- `k` is the number of steps of the model that the tool will explore; and
- `delta` is the precision for the δ-decision problem.

Statistical Testing Options

SReach can be used with different statistical testing methods through the following specifications.

Lai's test: `Lai <theta> <cost_per_sample>`, where `theta` indicates the probability threshold.

Bayes factor test: `BFT <theta> <T> <alpha> <beta>`, where `theta` is a probability threshold satisfying `0 < theta < 1`, `T` is a ratio threshold satisfying `T > 1`, and `alpha`, and `beta` are beta prior parameters.

BFT with indifference region: `BFTI <theta> <T> <alpha> <beta> <delta>`, where, besides the parameters used in the above Bayes factor test, `delta` is given to create the indifference region - [p_0, p_1], where p_0 = `theta` - `delta` and p_1 = `theta` + `delta`. Now, it tests $H_0 : \ p \geq p_0$ against $H_1 : \ p \leq p_1$.

Sequential probability ratio test (SPRT): `SPRT <theta> <T> <delta>`.

Chernoff-Hoeffding bound: `CHB <delta1> <coverage_probability>`, where `delta1` is the given precision, and `coverage_probability` indicates the confidence.

Bayesian Interval Estimation with Beta prior:
`BEST <delta1> <coverage_probability> <alpha> <beta>`.

Direct/Naïve Sampling: `NSAM <num_of_samples>`.

Both sequential and parallel versions of *SReach* are available on `https://github.com/dreal/SReach`.

5.4 Case Studies

5.4.1 Atrial Fibrillation

The heart rhythm is enabled by the electrical activity of cardiac muscle cells, which make up the atria and ventricles. The electrical dynamics of cardiac cells is governed by the organized opening and closing of ion channel gates on the cell membrane. Improper functioning of the cardiac cell ionic channels can cause the cells to lose excitability, which disorders electric wave propagation and leads to cardiac abnormalities such as ventricular *tachycardia* or *fibrillation*. Mathematical modeling the dynamics of cardiac cells is important in understanding the mechanisms of cardiac disorders. [49] has developed an extremely versatile electrical model for cardiac cells, referred as minimum resistor model (MRM), which reproduces experimentally measured characteristics of human ventricular cell dynamics.

MRM (see Figure 5.3) contains 4 state variables and 26 parameters. An action potential (AP) is a change in the cells transmembrane potential u, as a response to an external stimulus (current) ϵ. The flow of total currents is controlled by a fast channel gate v and two slow gates w and s. In Mode 1, gates v and w are open and gate s is closed. The transmembrane potassium current causes the decay of u. The cell is resting and waiting for stimulation. We assume external stimulus ϵ equals to 1 and lasts for 1 millisecond. The stimulation causes u increase which may trigger $\mathsf{jump}_{1\to 2} : u \geq \theta_o$. In Mode 2, v starts closing. The decay rate of u changes. The systems will jump to Mode 3 if $u \geq \theta_w$. In Mode 3, w is also closing. u is governed by the potassium current and the calcium current. When $u \geq \theta_v$, Mode 4 can be reached which means a successful AP initiation. In Mode 4, u reaches its peak due to the fast opening of sodium channel. The cardiac muscle contracts and u starts decreasing.

MRM reduces the complexity of existing models by representing channel gates of different ions with one fast channel and two slow gates. Identifying the parameter ranges for which the MRM accurately reproduces cardiac abnormalities will benefit the development of the treatment of cardiac disorders. However, due to the simplification, for most model parameters, it becomes impossible to obtain their values through measurements. After adding parametric uncertainty into the original hybrid model, we show that *SReach* can be adapted to synthesize parameters for this stochastic model, i.e., identifying appropriate ranges and distributions for model parameters. We chose two system parameters - $\mathit{EPI\_TO}1$ and $\mathit{EPI\_TO}2$, and varied their distributions to see which ones allow the model to present the desired patterns. As in Table 5.1, when $\mathit{EPI\_TO}1$ is either close to 400, or between 0.0061 and 0.007, and $\mathit{EPI\_TO}2$ is close to 6, the model can satisfy the given bounded reachability property with a probability very close to 1. The analysis for this model was conducted on a server with 2* AMD Opteron(tm) Processor 6172 and 32GB RAM (12 cores were used), running on Ubuntu 14.04.1 LTS. In our experiments, we used 0.001 as the precision for the δ-decision problem, and Bayesian sequential estimation with 0.01 as the estimation error bound, coverage probability 0.99, and a uniform prior ($\alpha = \beta = 1$).

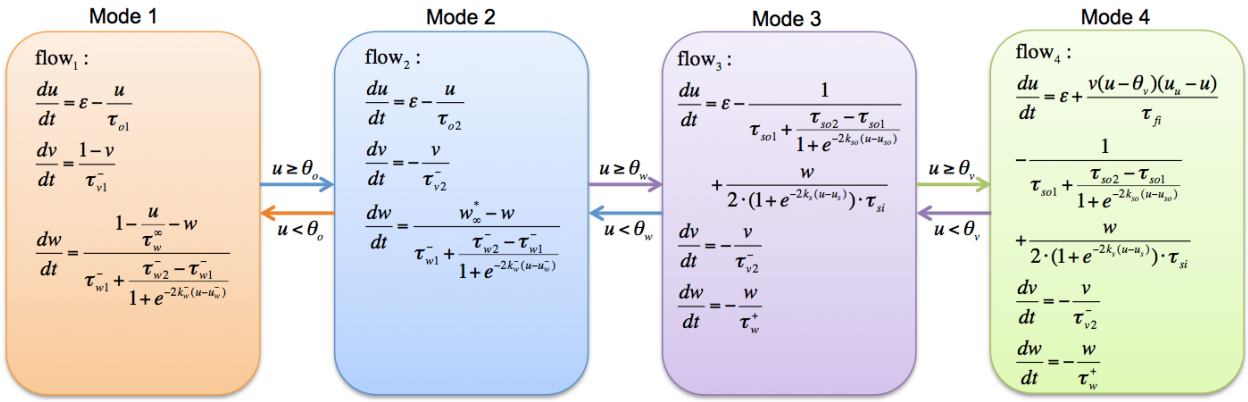


Figure 5.3: The minimal resistor model of cardiac cells

Model	#RVs	EPI_TO1	EPI_TO2	#S_S	#T_S	Est_P	A_T(s)	T_T(s)
Cd_to1_s	1	U(6.1e-3, 7e-3)	6	240	240	0.996	0.270	64.80
Cd_to1_uns	1	U(5.5e-3, 5.9e-3)	6	0	240	0.004	0.042	10.08
Cd_to2_s	1	400	U(0.131, 6)	240	240	0.996	0.231	55.36
Cd_to2_uns	1	400	U(0.1, 0.129)	0	240	0.004	0.038	9.15
Cd_to12_s	2	N(400, 1e-4)	N(6, 1e-4)	240	240	0.996	0.091	21.87
Cd_to12_uns	2	N(5.5e-3, 10e-6)	N(0.11, 10e-5)	0	240	0.004	0.037	8.90

Table 5.1: Results for the 4-mode atrial fibrillation model ($k = 3$). For each sample generated, *SReach* analyzed systems with 62 variables and 24 ODEs in the unfolded SMT formulae. #RVs = number of random variables in the model, #S_S = number of δ-sat samples, #T_S = total number of samples, Est_P = estimated probability of property, A_T(s) = average CPU time of each sample in seconds, and T_T(s) = total CPU time for all samples in seconds. Note that, we use the same notations in the remaining tables.

5.4.2 Prostate Cancer Treatment

Prostate cancer is the second leading cause of cancer-related deaths among men in United States [195]. Hormone therapy in the form of androgen deprivation has been a cornerstone of the management of advanced prostate cancer for several decades. However, controversy remains regarding its optimum application [47]. Continuous androgen suppression (CAS) therapy has many side effects including anemia, osteoporosis, impotence, etc. Further, most patients experience a relapse after a median duration of 18-24 months of CAS treatment, due to the proliferation of castration resistant cancer cells (CRCs).

In order to reduce side effects of CAS and to delay the time to relapse, intermittent androgen suppression (IAS) was proposed aiming to limit the duration of androgen-poor conditions and avoid emergence of AI cells [41]. In details, IAS therapy switches between on-treatment and off-treatment modes by monitoring the serum level of a tumor marker called prostate-specific antigen (PSA): (i) when the PSA level decreases and reaches a lower threshold value r_0, androgen suppression is suspended; (ii) when the PSA level increases and reaches a upper threshold value r_1, androgen suppression is resumed by the administration of medical agents.

Recent clinical phase II and III trials confirm that IAS has significant advantages in terms of

quality of life and cost. However, with respect to time to relapse and cancer-specific survival, the clinical trials suggest that to what extent IAS is superior to CAS depends on the individual patient and the on- and off-treatment scheme [42, 43, 109]. Thus, a crucial unsolved problem is how to design a personalized treatment scheme for each individual to achieve maximum therapeutic efficacy.

In [154], Liu et al. recently proposed a nonlinear hybrid model to reproduce the clinical observations [42, 43] of prostate cancer cell dynamics in response to the IAS therapy. It is known that the proliferation and survival of prostate cancer cells depend on the levels of androgens, specifically testosterone and 5α-dihydrotestosterone (DHT). This model considers two distinct subpopulations of prostate cancer cells: hormone sensitive cells (HSCs) and castration resistant cells (CRCs). Androgen deprivation can lead to remarkable decreases of the proliferation and survival rates of HSCs, but also up-regulates the conversion from HSCs to CRCs, which will keep proliferating under low androgen level.

The model has two modes which are shown in Figure 5.4. $x(t)$, $y(t)$, and $z(t)$ represent the population of AD cells, the population of AI cells, and the serum androgen concentration, respectively. The growth dynamics of AD and AI cells are governed by their proliferation rate, apoptosis rate and mutation rate from AD to AI phenotype, depending on androgen concentration $z(t)$. The PSA level v (ng ml^{-1}) is defined as $v(t) = x(t) + y(t)$. The treatment is suspended or restarted according to the value of v and dv/dt. In mode 2 (off-treatment), the androgen concentration is maintained at the normal level z_0 by homeostasis. In mode 1 (on-treatment), the androgen is cleared at a rate $1/\tau$.

This two-mode model captures the intermittent androgen suppression (IAS) therapy that switches between treatment ON and OFF according to the serum level thresholds of prostate-specific antigen, namely r_0 and r_1. As suggested by the clinical trials [44], an effective IAS therapy highly depends on the individual patient. Thus, we modified this two-mode model by taking parametric variation caused by personalized differences into account. In detail, according to clinical data from hundreds of patients [45], we replaced six system parameters with random variables having appropriate (continuous) distributions, including α_x (the proliferation rate of androgen-dependent (AD) cells), α_y (the proliferation rate of androgen-independent (AI) cells), β_x (the apoptosis rate of AD cells), β_y (the apoptosis rate of AI cells), m_1 (the mutation rate from AD to AI cells), and z_0 (the normal androgen level). To describe the variations due to individual differences, we assigned α_x to be $U(0.0193, 0.0214)$, α_y to be $U(0.0230, 0.0254)$, β_x to be $U(0.0072, 0.0079)$, β_y to be $U(0.0160, 0.0176)$, m_1 to be $U(0.0000475, 0.0000525)$, and z_0 to be $N(30.0, 0.001)$. We used *SReach* to estimate the probabilities of preventing the relapse of prostate cancer with three distinct pairs of treatment thresholds (*i.e.*, combinations of r_0 and r_1). As shown in Table 5.2, the model with thresholds $r_0 = 10$ and $r_1 = 15$ has a maximum posterior probability that approaches 1, indicating that these thresholds may be considered for the general treatment. The experiment for this stochastic model was conducted on a server with 2* AMD Opteron(tm) Processor 6172 and 32GB RAM (12 cores were used), running on Ubuntu 14.04.1 LTS. In our experiments we used 0.001 as the precision for the δ-decision problem, and Bayesian sequential estimation with 0.01 as the estimation error bound, coverage probability 0.99, and a uniform prior ($\alpha = \beta = 1$).

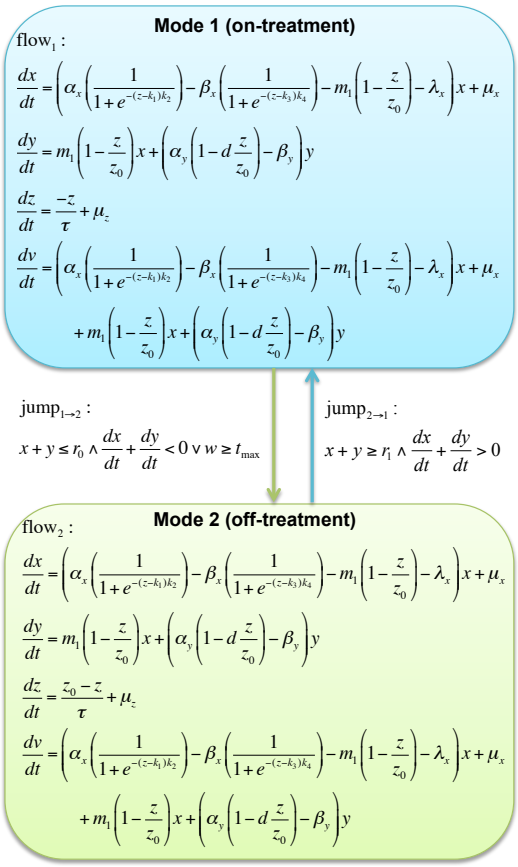


Figure 5.4: A hybrid automaton model for prostate cancer hormone therapy

Model	#RVs	r_0	r_1	Est_P	#S_S	#T_S	A_T(s)	T_T(s)
PCT1	6	5.0	10.0	0.496	8226	16584	0.596	9892
PCT2	6	7.0	11.0	0.994	335	336	54.307	18247
PCT3	6	10.0	15.0	0.996	240	240	506.5	121560

Table 5.2: Results for the 2-mode prostate cancer treatment model ($k = 2$). For each sample generated, *SReach* analyzed systems with 41 variables and 10 ODEs in the unfolded SMT formulae.

5.4.3 Tap Withdrawal Circuit in C. elegans

Note. This case study is led by Md. Ariful Islam.

Due to the simplicity of its nervous system (302 neurons, ∼5,000 synapses) and the breadth of research on the animal, *C. elegans*, the common roundworm, is a model system for neuroscience. The complete connectome of the worm is documented [40, 221], and a number of interesting experiments have been carried out on its locomotory neural circuits connecting sensory neurons to motor neurons [17, 102, 140, 233]. Of particular interest is the *Tap Withdrawal* (TW) neuronal circuit that governs the reactionary motion of the animal when the petri dish in which it swims is subjected to a mechanical tap [134]. (A related circuit, *touch sensitivity*, controls the reaction of

the worm when a stimulus is applied to a single point on the body.) The term "tap withdrawal" refers to the fact that worms swimming in a petri dish tend to *withdraw* (turn around and swim in the opposite direction) when subjected to a tap stimulus. Presumably, this is because the tap causes them to sense danger in their surrounding environment. The worms, however, can be conditioned or *habituated* to ignore this stimulus [184].

Studies of the TW circuit have traditionally involved using lasers to ablate different neurons in the circuit of multiple animals, and then measuring the response behavior when tap stimuli are applied [24]. Such is the case for [222]; see also Fig. 5.6. Such behaviors are logged with the percentage of the experimental population to display that behavior. Moreover, with the aim of predicting synaptic polarities (unknown parameters) of the TW circuit, the dynamics of the membrane potential of different neurons has been mathematically modeled [223]. This model is in the form of a system of nonlinear ODEs with an indication of the polarity (inhibitory or excitatory) of each neuron in the circuit. The Wicks et al. circuit model has a significant number of parameters, including gap-junction conductance, membrane capacitance and leakage current, that decisively affect the circuit's behavior. Fixed values for these parameters have been provided based on the measurements performed on single in-vitro neurons [223]. The model therefore produces the predominant behavior in most ablation groups with a few exceptions. While the experimental work and the model presented in were by no means insubstantial, the exploration of the model is vastly incomplete. Fixed parameter values fit through experimentation cause the model to replicate the predominant behavior seen in the mentioned experiments, but little can be gained beyond that. All such animals are not created equal owing to genetic variation, and, during their lifetime, they are exposed to stimuli of varying intensity, duration, and frequency. Carefully and (semi-)exhaustively varying the circuit parameters of the [223] model should provide us with insights underlying these processes, and ultimately help us to understand the learning process in neural circuits.

Towards this end, we use SReach to perform the bounded-time reachability analysis on the TW circuit model of Wicks et al. (1996) to estimate the probability of various TW responses related to parameter uncertainty, and thus to derive population percentages that exhibit various behaviors in response to tap stimuli. This case study shows that SReach can handle large-scale systems for which traditional reachability analysis may not scale.

Tap Withdrawal Neuronal Circuit in C. elegans

In *C. elegans*, there are three classes of neurons: *sensory*, *inter*, and *motor*. For the TW circuit, the sensory neurons are *PLM*, *PVD*, *ALM,* and *AVM*, and the inter-neurons are *AVD*, *DVA*, *PVC*, *AVA*, and *AVB*. The model we are using abstracts away the motor neurons as simply forward and reverse movement. Neurons are connected in two ways: electrically via bi-directional *gap junctions*, and chemically via uni-directional chemical *synapses*. Each connection has varying degrees of throughput, and each neuron can be *excitatory* or *inhibitory*, governing the polarity of transmitted signals. These polarities were experimentally determined in [223], and used to produce the circuit shown in Fig. 5.5. In [222], Wicks et al. performed a series of laser ablation experiments in which they knocked out neurons in a group of animals (worms), subjected them to a tapped surface, and recorded the magnitude and direction of the resulting behavior. Fig. 5.6 shows the response types for each of their experiments.

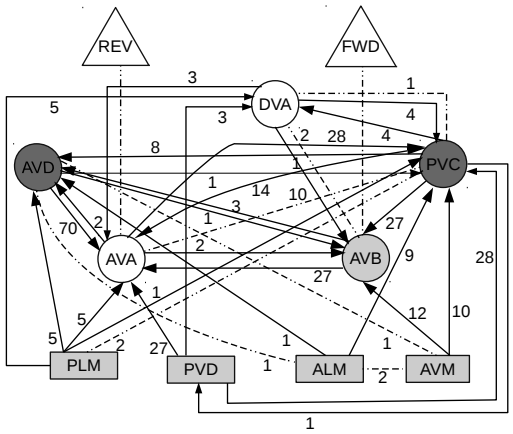


Figure 5.5: Tap Withdrawal Circuit of C. elegans. Rectangle: Sensory Neurons; Circle: Inter-neurons; Dashed Undirected Edge: Gap Junction; Solid Directed Edge: Chemical Synapse; Edge Label: Number of Connections [32]; Dark Gray: Excitatory Neuron; Light Gray: Inhibitory Neuron; White: Unknown Polarity. FWD: Forward Motor system; REV: Reverse Motor System.

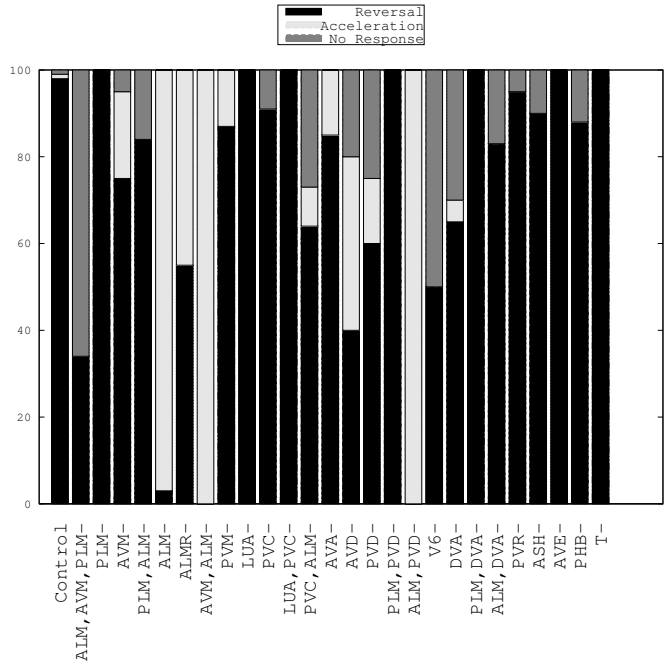


Figure 5.6: Effect of ablation on Tap Withdrawal reflex (experimental results). The length of the bars indicate the fraction of the population demonstrating the particular behavior. [222]

Mathematical Model of Tap Withdrawal Neuronal Circuit

The dynamics of a neuron's membrane potential, *V*, is determined by the internal state of the neuron together with sum of all input currents [145], written as:

$$\frac{dV}{dt} = \frac{1}{CR}(V^{leak} - V) + \frac{1}{C}\sum(I^{gap} + I^{syn} + I^{stim})$$

where V represents the *membrane potential*, C is the *membrane capacitance*, R is the *membrane*

resistance, V^{leak} is the *leakage potential*, I^{gap} and I^{syn} are gap-junction and the chemical synapse currents, respectively, and I^{stim} is the applied external *stimulus* current. The summations are over all neurons with which this neuron has a (gap-junction or synaptic) connection.

The current flows between neuron i and j via n_{ij} gap-junctions can be seen as the current passing through n parallel resistors. Therefore, based on the Ohm's law, one can derive the gap-junction current equation as follows:

$$I_{ij}^{gap} = n_{ij} g_m^{gap} (V_j - V_i)$$

where the constant g_m^{gap} is the *maximum conductance* of the gap junction, and n_{ij} is the number of gap-junctions between neurons i and j. The conductance g_m^{gap} defines the strength of a connection between two neurons. As a consequence, it sets the amount of shared information of the two neurons. This key parameter significantly affects the behavior of the neural circuits.

Chemical synapses transfer information by releasing neurotransmitters [135]. Inspired by Hodgkin-Huxley model of ionic channels [120], one can model such behavior as a synaptic current flowing from presynaptic neuron j to post-synaptic neuron i as below:

$$I_{ij}^{syn} = n_{ij}^{syn} g_{ij}^{syn}(t)(E_j - V_i)$$

where $g_{ij}^{syn}(t)$ is the voltage-dependent synaptic conductance of neuron i, n_{ij}^{syn} is the number of synaptic connections from neuron j to neuron i, and E_j is the *reversal potential* of neuron j for the synaptic conductance.

The chemical synapse is characterized by a synaptic sign, or polarity, specifying if said synapse is excitatory or inhibitory. The value of E_j is assumed to be constant for the same synaptic sign.

For a neuron of C. elegans at equilibrium, the membrane potential on average is around -30mV. According to the Eq. 5.4.3, by setting the reversal potential value to a higher values than the resting potential of a neuron, the synaptic current increases and therefore an excitatory behavior is realized. On the contrary, an inhibitory synapse is developed by placing the value of the reversal potential less than the equilibrium potential of the neuron.

Dynamics of the Synaptic conductance depends on the membrane potential state of the presynaptic neuron V_j. For the sake of simplicity, Wicks et al. model such dynamics by the steady-state response of the synapse as follows

$$g_{ij}^{syn}(t) = g_{\infty}^{syn}(V_j)$$

where the conductance at steady-state is given by:

$$g_{\infty}^{syn}(V_j) = \frac{g_m^{syn}}{1 + \exp\left(k \frac{V_j - V_j^{eq}}{V_{Range}}\right)}$$

g_m^{syn} presents the *maximum synaptic conductance*, V_j^{eq} is the *pre-synaptic equilibrium potential*, and V_{Range} is the *pre-synaptic voltage range* over which the synapse is activated. k is an experimentally derived constant, valued at -4.3944.

Combining all of the above pieces, the mathematical model of the TW circuit is a system of nonlinear ODEs, with each state variable defined as the membrane potential of a neuron in the neural circuit. Consider a circuit with N neurons. The dynamics of the i^{th} neuron of the circuit is given by:

$$\frac{dV_i}{dt} = \frac{V_{l_i} - V_i}{C_i R_i} + \sum_{j=1}^{N} (I_{ij}^{gap} + I_{ij}^{syn} + I_i^{stim}) \tag{5.1}$$

$$I_{ij}^{gap} = n_{ij}^{gap} g_m^{gap} (V_j - V_i) \tag{5.2}$$

$$I_{ij}^{syn} = n_{ij}^{syn} g_{ij}^{syn} (E_j - V_i) \tag{5.3}$$

$$g_{ij}^{syn} = \frac{g_m^{syn}}{1 + \exp\left(k \frac{V_j - V_j^{eq}}{V_{Range}}\right)}. \tag{5.4}$$

The equilibrium potentials (V^{eq}) of the neurons are computed by setting the left-hand side of Eq. (5.1) to zero [223]. This leads to a system of linear equations, that can be solved as follows:

$$V^{eq} = A^{-1} b \tag{5.5}$$

where matrix A is given by:

$$A_{ij} = \begin{cases} -R_i n_{ij}^{gap} g_m^{gap} & \text{if } i \neq j \\ 1 + R_i \sum_{j=1}^{N} n_{ij}^{gap} g_{ij}^{gap} g_m^{syn} / 2 & \text{if } i = j \end{cases}$$

and vector b is written as:

$$b_i = V_{l_i} + R_{m_i} \sum_{j=1}^{N} E_j n_{ij}^{syn} g_m^{syn} / 2.$$

Tap Withdrawal Response Patterns

The Wicks et al. model does not explicitly incorporate nematode locomotion. It simply defines the relationship between the animals locomotion and activation of the TW circuit that controls the behavior.

Wicks et al. assumes that the output of the TW circuit controls locomotory behavior primarily through the action of the inter-neurons AVB and AVA. The AVA interneurons make gap junctions and chemical synapses with motor neurons AS, VA, and DA that excite backward locomotion, whereas the the AVB interneurons form gap junctions with the motor neurons VB and DB that excite forward locomotion. Thus, Wicks et al. simply assume that the degree of backward (forward) locomotion is proportional to the depolarization of the AVA (AVB) interneuron and inversely proportional to the depolarization of the AVB (AVA) interneuron. Recently, Kawano et al. present a study in [143] that supports the assumptions made by Wicks et al. on directional movement of

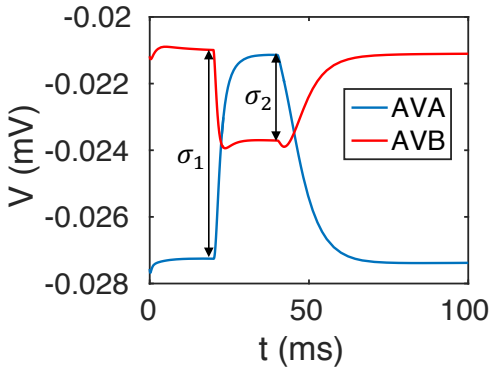


Figure 5.7: TBD.

C. elegans. Through in-vivo calcium imaging, electrophysiology and behavioral analyses of wild-type animals and innexin mutants, they show that the initiation of reversal movement is directly correlated with a increased calcium level in AVA. In contrast, the initiation of forward movement is associated with an increased calcium level in AVB and a decrease of the calcium transient correlated with either a reduced forward velocity or reversals.

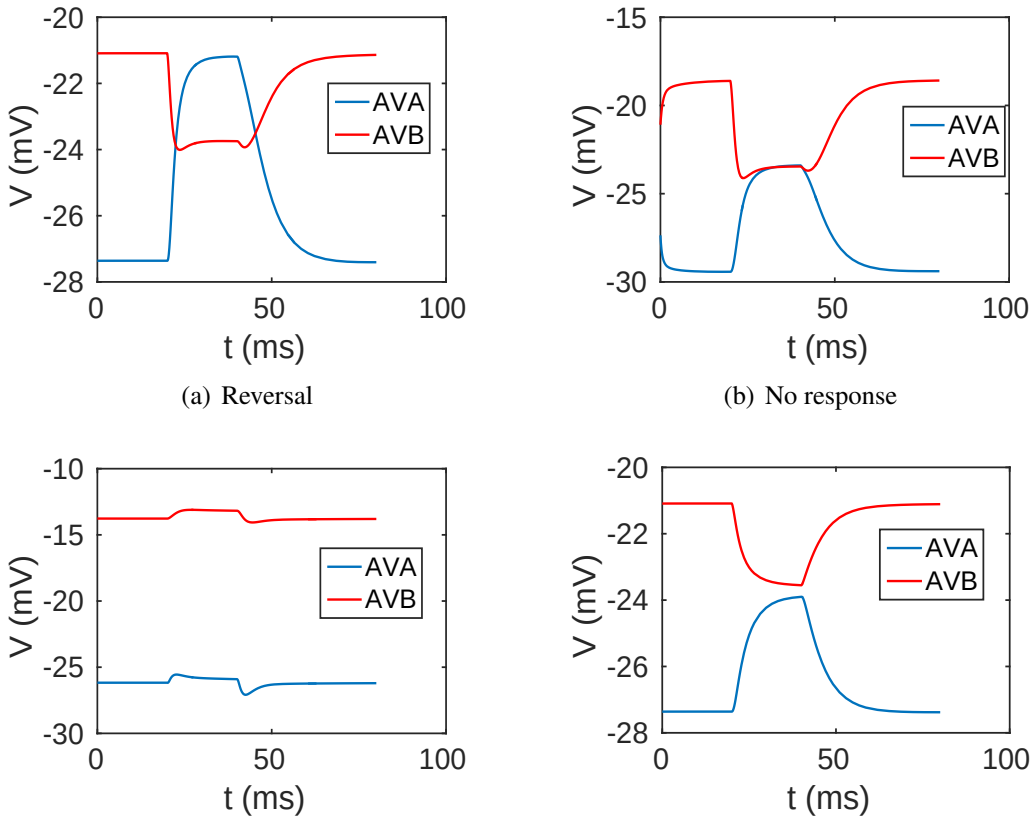
Under standard laboratory culture conditions, the animal predominantly generates continuous forward movement without any tap stimulation [104, 174]. In [222], Wicks et al. experienced, through in vivo experiment, only three tap withdrawal responses: reversal, acceleration and no response. The simulation of the Wicks et al. model for certain ablation group (e.g., AVM,PVC-ablation group), however, shows the animal can also predominantly generates continuous backward movement without any tap stimulation. Additionally, the study of Kawano et al. supports the evidence of new type of response like deceleration (reduced forward velocity). These lead us to believe that, at least in theory, it is possible to have few more tap withdrawal responses as compared to what Wicks et al. experienced in their wet-lab experiments.

Similar to [143, 223], we consider the directional movement can be inferred based on the voltage difference between AVA and AVB interneurons:

- Forward movement: $v_{AVB} > v_{AVA}$
- Backward movement: $v_{AVB} < v_{AVA}$
- No movement: $v_{AVB} \sim v_{AVA}$

Assume that $\sigma_i = v^i_{AVB} - v^i_{AVA}$, $i \in \{1, 2\}$ be the voltage differences between AVB and AVA interneurons during non-stimulation and stimulation period, respectively, as shown in Fig. 5.7 and ϵ is some small positive number. Based σ_1, we categorize the TW responses into two subgroups:

- When $\sigma_1 >= 0$:
  1. Reversal: $\sigma_2 <= -\epsilon$



(a) Reversal

(b) No response

(c) Forward acceleration

(d) Forward deceleration

Figure 5.8: Different tap withdrawal responses when, before the applying tap stimulation, the animal moves in forward direction.

2. No response: $|\sigma_2| <= \epsilon$
3. Forward acceleration: $(\sigma_2 >= \epsilon) \wedge (\sigma_2 >= \sigma_1)$
4. Forward deceleration: $(\sigma_2 >= \epsilon) \wedge (\sigma_2 < \sigma_1)$

- When $\sigma_1 < 0$:

1. Forward: $\sigma_2 >= \epsilon$
2. No response: $|\sigma_2| <= \epsilon$
3. Backward acceleration: $(\sigma_2 <= -\epsilon) \wedge (\sigma_2 <= \sigma_1)$
4. Backward deceleration: $(\sigma_2 >= \epsilon) \wedge (\sigma_2 > \sigma_1)$

Fig. 5.8 shows all four response patterns for the first subgroup. The response patterns for the second subgroup, however, will have the similar structures, if we interchange AVB with AVA in the figure.

Normalization of the Wicks et al. Model

SReach internally uses dReach [147], which relies on numerical computation. As the the values of the parameters in the Wicks et al. model are in the order of 10^{-9} to 10^{-12}, the computation often suffers from numerical instability. To take into account this issue, we normalize the Wicks et al. model with respect to the capacitance, which is a common practice in modeling biological systems [87]. The values of the parameter in this normalized model are in the order of 10 t0 10^3.

To normalize, we combine Eqs.(5.1) to (5.4):

$$\dot{V_i} \quad = \quad \frac{V_{l_i} - V_i}{R_i C_i} \; + \; \frac{g_m^{gap}}{C_i} \sum_{j=1}^{N} n_{ij}^{gap}(V_j - V_i) \; + \; \frac{g_m^{syn}}{C_i} \sum_{j=1}^{N} \frac{n_{ij}^{syn}(E_j - V_i)}{1 + \exp\left(k \frac{V_j - V_j^{EQ}}{V_{Range}}\right)} \; + \; \frac{1}{C_i} I_i^{stim}$$

Now letting $g_i^{leak} = \frac{1}{R_i C_i}$, $g_i^{gap} = \frac{g_m^{gap}}{C_i}$, $g_i^{syn} = \frac{g_m^{syn}}{C_i}$ and $I_i^{ext} = \frac{I_i^{stim}}{C_i}$ the normalized system dynamics can be written as:

$$\dot{V_i} \; = \; g_i^{leak}(V_{l_i} - V_i) + g_i^{gap} \sum_{j=1}^{N} n_{ij}^{gap}(V_j - V_i) + g_i^{syn} \sum_{j=1}^{N} \frac{n_{ij}^{syn}(E_j - V_i)}{1 + \exp\left(k \frac{V_j - V_j^{eq}}{V_{Range}}\right)} + I_i^{ext} \quad (5.6)$$

Hybrid automaton for TW circuit M_{TW}:

For the TW circuit, Wicks et al. model the tap stimulus as a phasic current that is applied to sensory neurons (AVM, ALM and PLM) simultaneously. The phasic current is, typically, a square-wave signal with a fixed duration. Due to the piece-wise continuous nature of this signal, we represent Wicks et al. model as a hybrid automaton by dividing the dynamics into stimulus and non-stimulus modes. Additionally, when a tap is applied to the worm, it is assumed that the worm is operating in a stable condition. To take this into account, we apply the stimulation after a

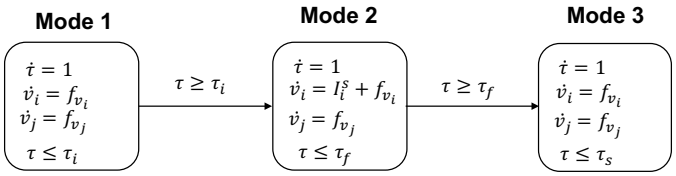


Figure 5.9: The 3-mode hybrid automaton M_{TW} for the Wicks et al. model

transition period. Assume that $[0, \tau_i]$ is the transition period, $[\tau_i, \tau_f]$ is the stimulation period and $[0, \tau_s]$ is the total simulation duration. Fig. 5.9 shows the hybrid automaton M_{TW} for the Wicks et al. model. The subscript i and j in the figure are used to denote the sensory neurons and the interneurons, respectively. We add an additional variable τ to support time-triggered jump from one mode to another.

TW response as Hybrid Automaton, M_ϕ:

In above, we enumerated all possible TW responses and formalized them in terms of σ_1 and σ_2, the steady-state voltage difference between AVB and AVA interneurons during non-stimulus and stimulation period. Hence, to encode a TW response ϕ as hybrid automaton M_ϕ, we augment M_{TW} by adding two additional state variables σ_1 and σ_2. As these two variables measure the steady-state voltage differences, which are constant in time, the vector-fields for them are set to zero. However, as shown in Fig. 5.10, both σ_1 and σ_2 are reset to $v_{AVB} - v_{AVA}$ during the jump from "Mode 1" to "Mode 2" and "Mode 2" to "Mode 3", respectively. This ensures the correct values of σ_1 and σ_2 in "Mode 3".

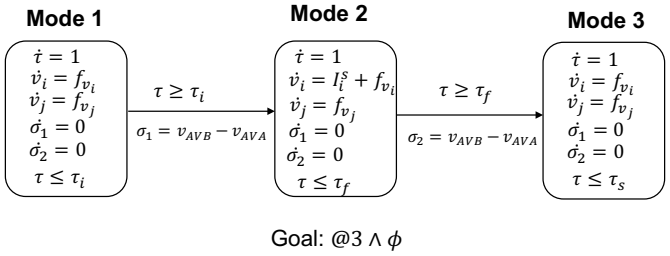


Figure 5.10: The 3-mode hybrid automaton M_ϕ for response ϕ of the TW circuit

Parameter Uncertainty

C. elegans nervous system has been used for the study of fundamental problems in the function of neurons and neuronal circuits for many years. Due to its small size, the technique to record electrophysiological data, however, was simply not developed during the time when Wicks et al. derived the mathematical model for the TW circuit.

For this model, Wicks et al. extrapolated the electrophysiological data from Ascaris, a larger nematode related to C. elegans. Based on [181], they first assumed a standard membrane properties of each neuron in the TW circuit, such as, membrane capacitance, resistance etc., for a unit

area. They then estimated the area based on the process lengths and cell diameter, measured using electron micrographs and branching morphology [220, 221, 226].

However, the diameter of the cell varies from 0.2 to 1.0 μm and the soma diameters of the process lengths, assuming the worm length of 1 mm, vary between 2 to 10 μm [226]. As a result, the surface area greatly varies, which, in turns, causes variability in membrane properties. Like the membrane properties, the gap-junction and synaptic conductance varied among the population of the worms [170]. Both variability causes parameter uncertainty in the mathematical model of the TW circuit.

Due to this parameter uncertainty, we consider each parameter p of the Wicks et al. model as a random variable. As a result, M_ϕ becomes a stochastic hybrid automaton. Now for each response ϕ, we formulate a probabilistic reachability problem as to estimate the probability such that "Mode 3" $\wedge$ ϕ is reachable in M_ϕ. We solve this problem using SReach.

Model Analysis

As we discussed above, the values of the parameters are determined based on the size of surface area of neurons, gap-junctions and synapses. But these surface areas vary among population. According to [170], the area of gap-junction vary between 1 to 10 ncm^2 and standard gap-junction conductance per unit area (1 cm^2) is 1 S (Siemens) [28]. We use this information as the basis to consider 1 to 10 nS as the biologically relevant range for gap-junction conductance and g_i^{gap} is chosen by dividing capacitance of the corresponding neuron. However, as we could not able to find any biologically relevant ranges for synaptic and leakage conductance from the literature, we considered those parameters as constants according to the Table 5.3 from [223].

We performed our analysis on the control and five ablation groups. In each analysis, we consider Bayesian sequential estimation with 0.05 as the estimation error bound, 0.95 as the coverage probability, and a uniform prior ($\alpha = \beta = 1$). For initial condition, we first simulated the Wicks et al. model without applying any stimulation and then considered the steady state values from simulation as the initial conditions. We set the initial value of σ_1 and σ_2 to zero, stimulus current as 100 pA/pF and ϵ to 10^{-4} (0.1 mV). For computation, we used parallel version of SReach on a 32-core machine.

Table 5.3 shows the estimated probability of each TW response for all six groups, where we considered g_i^{gap} as a uniform random variable in the given range described above. In contrast, Table 5.4 shows the results where we considered them as a normal random variables. For the random distribution, we considered the values of g_i^{gap} chosen by Wicks et al. in [223] as mean. But we chose the variance in such a way so that the normal distribution cover 99% of the range of g_i^{gap}. In all cases, the predominant response in each group are highlighted in bold on both tables.

The predominant responses that we determined from our analysis for the four groups conform with the predominant responses that Wicks et al. obtained based on their ablation experiments on actual worm in [222]. Note that Wicks et al. did not differentiate the acceleration and deceleration responses in both forward and backward directions. As a result, their distributions on the TW responses have only three responses, as opposed to the seven responses in our distributions. In

Group	REV		NR		F-ACC		F-DEC		FWD		B-ACC		B-DEC	
	Pr	RT (s)	Pr	RT (s)	Pr	RT (s)	Pr	RT (s)	Pr	RT (s)	Pr	RT (s)	Pr	RT (s)
Control	**0.95**	801.78	0.030	87.45	0.015	16.48	0.038	282.67	0.015	16.48	0.015	16.48	0.015	16.48
PLM	**0.95**	639.06	0.015	10.49	0.015	10.49	0.04	164.72	0.015	10.48	0.015	10.48	0.015	10.48
ALM-AVM	0.015	8.57	0.015	8.57	**0.861**	973.60	0.158	728.97	0.015	8.57	0.015	8.57	0.015	8.57
ALM-DVA	**0.433**	1399.37	0.062	286.47	0.015	15.325	0.585	1518.54	0.015	16.48	0.015	16.48	0.015	16.48
AVM-PVC	0.015	3.33	0.015	3.33	0.015	3.33	0.015	3.33	0.015	3.33	**0.984**	255.21	0.015	3.33
AVM-PLM	0.015	19.27	0.015	19.27	0.015	19.27	**0.984**	458.66	0.015	19.27	0.015	19.27	0.015	19.27

Table 5.3: Estimated probability and runtime for all response patterns by considering all g_i^{gap} as normal random variables

Group	REV		NR		F-ACC		F-DEC		FWD		B-ACC		B-DEC	
	Pr	RT (s)	Pr	RT (s)	Pr	RT (s)	Pr	RT (s)	Pr	RT (s)	Pr	RT (s)	Pr	RT (s)
Control	**0.83**	2343.87	0.039	252.58	0.015	26.37	0.121	897.47	0.015	26.37	0.015	26.37	0.015	26.37
PLM	**0.83**	1309.73	0.015	11.33	0.015	11.33	0.127	862.53	0.015	11.33	0.015	11.33	0.015	11.33
ALM-AVM	0.015	9.57	0.015	9.57	**0.689**	1578.83	0.33	1442.56	0.015	9.57	0.015	9.57	0.015	9.57
ALM-DVA	**0.414**	1406.31	0.0303	41.04	0.015	15.70	0.547	1766.48	0.015	15.70	0.015	15.70	0.015	15.70
AVM-PVC	0.015	3.33	0.015	3.33	0.015	3.33	0.015	3.33	0.015	3.33	**0.984**	255.21	0.015	3.33
AVM-PLM	0.03	72.02	0.015	16.49	0.015	16.49	**0.97**	419.39	0.015	16.49	0.015	16.49	0.015	16.49

Table 5.4: Estimated probability and runtime for all response patterns by considering all g_i^{gap} as uniform random variables

addition to these four groups, we performed analysis on two new ablation groups: AVM,PLM- and AVM,PVC-.

By comparing Table 5.3 with Table 5.4, we notice that the estimated probability of predominant response, computed by considering the parameters as normal random variables, is closer to the value obtained by Wicks et al. This indicates that the parameters are more likely to follow normal distribution over uniform distribution.

5.4.4 Additional Benchmarks

Benchmark	#Ms	K	#ODEs	#Vs	#RVs	δ	Est_P	#S_S	#T_S	A_T(s)	T_T(s)
BBK1	1	1	2	14	3	0.001	0.754	5372	7126	0.086	612.836
BBK5	1	5	2	38	3	0.001	0.059	209	3628	0.253	917.884
BBwDv1	2	2	4	20	4	0.001	0.208	2206	10919	0.080	873.522
BBwDv2K2	2	2	4	20	3	0.001	0.845	7330	8669	0.209	1811.821
BBwDv2K8	2	8	4	56	3	0.001	0.207	2259	10901	0.858	9353.058
Tld	2	7	2	33	4	0.001	0.996	227	227	0.213	48.351
Ted	2	7	4	50	4	0.001	0.996	227	227	12.839	2914.448
DTldK3	2	3	4	26	2	0.001	0.996	227	227	0.382	86.714
DTldK5	2	5	4	38	2	0.001	0.161	1442	8961	0.280	2509.078
W4mv1	4	3	8	26	6	0.001	0.381	5953	15639	0.238	3722.082
W4mv2K3	4	3	8	26	6	0.001	0.996	227	227	0.673	152.771
W4mv2K7	4	7	8	50	6	0.001	0.004	0	227	0.120	27.240
DWK1	2	1	4	14	5	0.001	0.996	227	227	0.171	38.817
DWK3	2	3	4	26	5	0.001	0.996	227	227	0.215	48.806
DWK9	2	9	4	62	5	0.001	0.996	227	227	5.144	1167.688
Que	3	2	3	13	4	0.001	0.228	2662	11677	0.095	1109.315
3dOsc	3	2	18	48	2	0.001	0.996	227	227	8.273	1877.969
QuadC	1	0	14	44	6	0.001	0.996	227	227	825.641	187420.507
exPHA01	2	2	4	20	2	0.001	0.524	345	658	5.01	3295.82
exPHA02	2	3	2	17	1	0.001	0.900	5361	5953	0.0004	2.35
KRk5	6	5	84	194	2	0.001	0.544	8946	16457	0.122	2015.64
KRk6	8	6	112	224	6	0.001	0.246	2032	8263	1.385	11444.22
KRk7	10	7	150	271	6	0.001	0.096	558	5795	16.275	94311.18
KRk8	7	8	105	303	6	0.001	0.004	0	227	0.003	0.58
KRk9	9	9	135	335	6	0.001	0.004	0	227	0.015	3.43
KRk10	11	10	165	367	6	0.001	0.004	0	227	0.026	5.92

Table 5.5: #Ms = number of modes, K indicates the unfolding steps, #ODEs = number of ODEs in the unfolded formulae, #Vs = number of total variables in the unfolded formulae, #RVs = number of random variables in the model, δ = precision used in *dReach*.

To further demonstrate SReach's applicability, we also applied it to additional benchmarks including HA_ps, PHAs, and PHA_rs with subtle non-determinism. Table 5.5 shows the results of

these experiments. These experiments were conducted with the sequential version of *SReach* on a machine with 2.9GHz Intel Core i7 processor and 8GB RAM, running OS X 10.9.2. In our experiments we used 0.001 as the precision for the δ-decision problem; and Bayesian sequential estimation with 0.01 half-interval width, coverage probability 0.99, and uniform prior ($\alpha = \beta = 1$). In the following table, "BB" refers to the bouncing ball models, "Tld" the thermostat model with linear temperature decrease, "Ted" the thermostat model with exponential decrease, "DT" the dual thermostat models, "W" the watertank models, "DW" the dual watertank models, "Que" the model for queuing system which has both nonlinear functions and nondeterministic jumps, "3dOsc" the model for 3d oscillator, and "QuadC" the model for quadcopter stabilization control. Following these hybrid systems with parametric uncertainty, we also consider two example PHAs - "exPHA01" and "exPHA02", and PHA$_r$s with trivial non-determinism - "KR" (the stochastic version of our killerred models).

Model description

Synthesized Killerred Model. The ODEs missing in Figure 5.11 are as follows.

$$
\begin{aligned}
\frac{\mathrm{d}[mRNA]}{\mathrm{d}t} &= k_{RNAsyn} \cdot [DNA] - k_{RNAdeg} \cdot [mRNA] \\
\frac{\mathrm{d}[KR_{im}]}{\mathrm{d}t} &= k_{KR_{im}syn} \cdot [mRNA] - (k_{KR_m} + k_{KR_{im}deg}) \cdot [KR_{im}] \\
\frac{\mathrm{d}[KR_{mdS}]}{\mathrm{d}t} &= k_{KR_m} \cdot [KR_{im}] - k_{KR_{mdS}deg} \cdot [KR_{mdS}] \; (before\ turning\ on\ the\ light) \\
\frac{\mathrm{d}[KR_{mdS}]}{\mathrm{d}t} &= k_{KR_m} \cdot [KR_{im}] + k_{KR_f} \cdot [KR_{mdS^*}] + k_{KR_{ic}} \cdot [KR_{mdS^*}] + k_{KR_{nrd}} \\
&\quad \cdot [KR_{mdT^*}] + k_{KR_{SOXd1}} \cdot [KR_{mdT^*}] - k_{KR_{ex}} \cdot [KR_{mdS}] - k_{KR_{mdS}deg} \\
&\quad \cdot [KR_{mdS}] \quad (after\ adding\ light) \\
\frac{\mathrm{d}[KR_{mdS^*}]}{\mathrm{d}t} &= k_{KR_{ex}} \cdot [KR_{mdS}] - k_{KR_f} \cdot [KR_{mdS^*}] - k_{KR_{ic}} \cdot [KR_{mdS^*}] \\
&\quad - k_{KR_{isc}} \cdot [KR_{mdS^*}] - k_{KR_{mdS^*}deg} \cdot [KR_{mdS^*}] \\
\frac{\mathrm{d}[KR_{mdT^*}]}{\mathrm{d}t} &= k_{KR_{isc}} \cdot [KR_{mdS^*}] - k_{KR_{nrd}} \cdot [KR_{mdT^*}] - k_{KR_{SOXd1}} \cdot [KR_{mdT^*}] \\
&\quad - k_{KR_{SOXd2}} \cdot [KR_{mdT^*}] - k_{KR_{mdT^*}deg} \cdot [KR_{mdT^*}] \\
\frac{\mathrm{d}[SOX]}{\mathrm{d}t} &= k_{KR_{SOXd1}} \cdot [KR_{mdT^*}] + k_{KR_{SOXd2}} \cdot [KR_{mdT^*}] - \frac{\mathrm{d}[SOX_{sod}]}{\mathrm{d}t} \\
\frac{\mathrm{d}[SOX_{sod}]}{\mathrm{d}t} &= k_{SOD} \cdot V_{maxSOD} \cdot \frac{[SOX]}{K_m + [SOX]}
\end{aligned}
$$

Atrial Fibrillation. The model has four discrete control locations, four state variables, and nonlinear ODEs. A typical set of ODEs in the model is as follows. The exponential term on the right-hand side of the ODE is the sigmoid function, which often appears in modeling biological

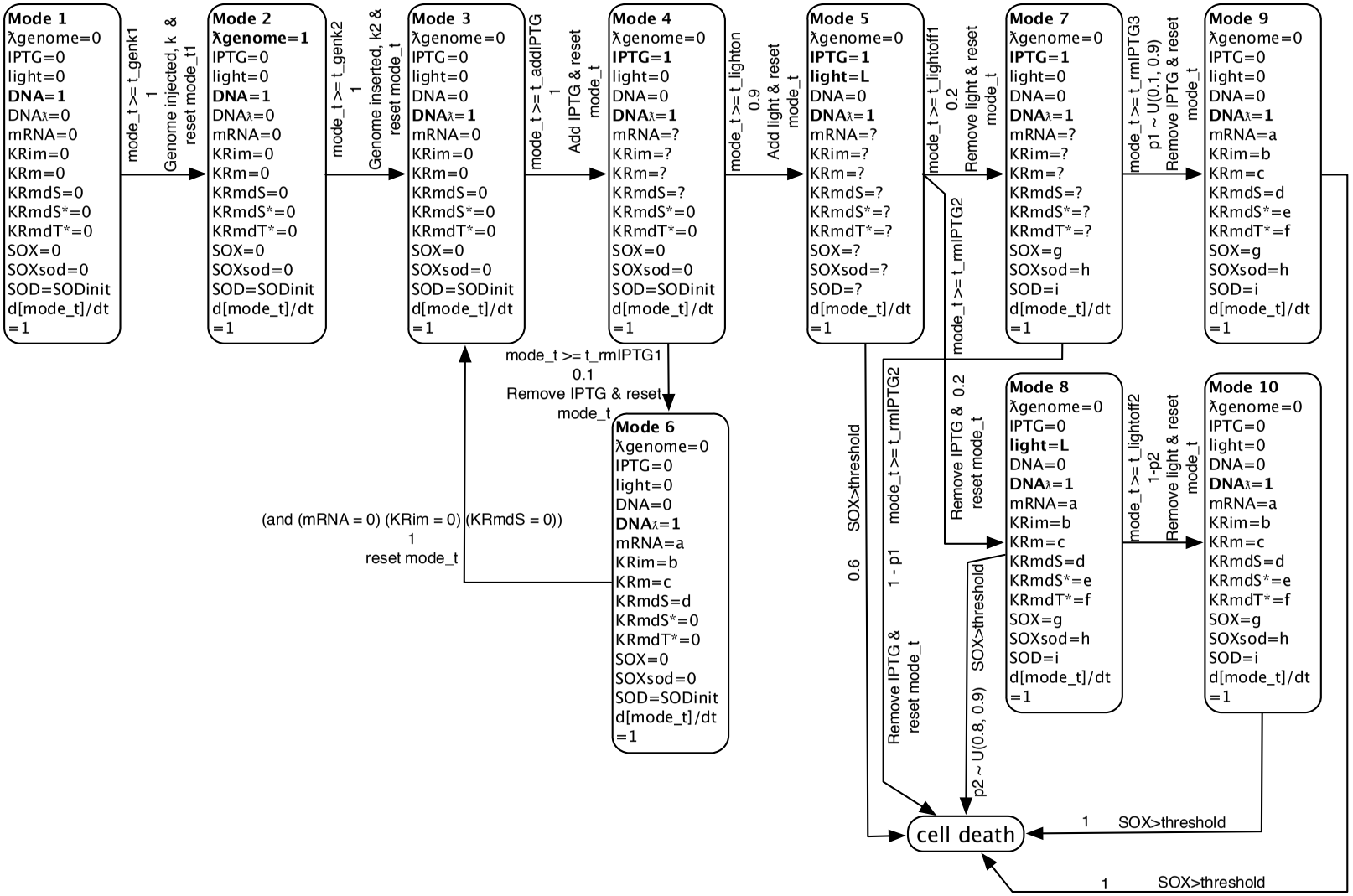


Figure 5.11: A probabilistic hybrid automaton for synthesized phage-based therapy model

switches.

$$\begin{aligned}
\frac{du}{dt} &= e + (u-\theta_v)(u_u - u)vg_{fi} + wsg_{si} - g_{so}(u) \\
\frac{ds}{dt} &= \frac{g_{s2}}{(1+\exp(-2k(u-us)))} - g_{s2}s \\
\frac{dv}{dt} &= -g_v^+ \cdot v \qquad \frac{dw}{dt} = -g_w^+ \cdot w
\end{aligned}$$

Prostate Cancer Treatment. The nonlinear ODEs in the Prostate-Cancer-Treatment model are as follows.

$$\begin{aligned}
\frac{dx}{dt} &= (\alpha_x(k_1 + (1-k_1)\frac{z}{z+k_2} - \beta_x((1-k_3)\frac{z}{z+k_4} + k_3)) - m_1(1-\frac{z}{z_0}))x + c_1x \\
\frac{dy}{dt} &= m_1(1-\frac{z}{z_0})x + (\alpha_y(1-d\frac{z}{z_0}) - \beta_y)y + c_2y \\
\frac{dz}{dt} &= \frac{-z}{\tau} + c_3z \\
\frac{dv}{dt} &= (\alpha_x(k_1 + (1-k_1)\frac{z}{z+k_2} - \beta_x(k_3 + (1-k_3)\frac{z}{z+k_4})) - m_1(1-\frac{z}{z_0}))x + c_1x \\
&\quad + m_1(1-\frac{z}{z_0})x + (\alpha_y(1-d\frac{z}{z_0}) - \beta_y)y + c_2y
\end{aligned}$$

Electronic Oscillator. The 3dOsc model represents an electronic oscillator model that contains nonlinear ODEs such as the following.

$$\begin{aligned}
\frac{dx}{dt} &= -ax \cdot sin(\omega_1 \cdot \tau) \\
\frac{dy}{dt} &= -ay \cdot sin((\omega_1 + c_1) \cdot \tau) \cdot sin(\omega_2) \cdot 2 \\
\frac{dz}{dt} &= -az \cdot sin((\omega_2 + c_2) \cdot \tau) \cdot cos(\omega_1) \cdot 2 \\
\frac{\omega_1}{dt} &= -c_3 \cdot \omega_1 \quad \frac{\omega_2}{dt} = -c_4 \cdot \omega_2 \quad \frac{d\tau}{dt} = 1
\end{aligned}$$

Quadcopter Control. We developed a model that contains the full dynamics of a quadcopter. We use the model to solve control problems by answering reachability questions. A typical set of the

differential equations are the following.

$$
\begin{aligned}
\frac{\mathrm{d}\omega_x}{\mathrm{d}t} &= L \cdot k \cdot (\omega_1^2 - \omega_3^2)(1/I_{xx}) - (I_{yy} - I_{zz})\omega_y\omega_z/I_{xx} \\
\frac{\mathrm{d}\omega_y}{\mathrm{d}t} &= L \cdot k \cdot (\omega_2^2 - \omega_4^2)(1/I_{yy}) - (I_{zz} - I_{xx})\omega_x\omega_z/I_{yy} \\
\frac{\mathrm{d}\omega_z}{\mathrm{d}t} &= b \cdot (\omega_1^2 - \omega_2^2 + \omega_3^2 - \omega_4^2)(1/I_{zz}) - (I_{xx} - I_{yy})\omega_x\omega_y/I_{zz} \\
\frac{\mathrm{d}\phi}{\mathrm{d}t} &= \omega_x + \frac{\sin(\phi)\sin(\theta)}{\left(\frac{\sin(\phi)^2\cos(\theta)}{\cos(\phi)} + \cos(\phi)\cos(\theta)\right)\cos(\phi)}\omega_y \\
&\quad + \frac{\sin(\theta)}{\frac{\sin(\phi)^2\cos(\theta)}{\cos(\phi)} + \cos(\phi)\cos(\theta)}\omega_z \\
\frac{\mathrm{d}\theta}{\mathrm{d}t} &= -\Big(\frac{\sin(\phi)^2\cos(\theta)}{\left(\frac{\sin(\phi)^2\cos(\theta)}{\cos(\phi)}\omega_y + \cos(\phi)\cos(\theta)\right)\cos(\phi)^2} \\
&\quad + \frac{1}{\cos(\phi)}\Big)\omega_y - \frac{\sin(\phi)\cos(\theta)}{\left(\frac{\sin(\phi)^2\cos(\theta)}{\cos(\phi)} + \cos(\phi)\cos(\theta)\right)\cos(\phi)}\omega_z \\
\frac{\mathrm{d}\psi}{\mathrm{d}t} &= \frac{\sin(\phi)}{\left(\frac{\sin(\phi)^2\cos(\theta)}{\cos(\phi)} + \cos(\phi)\cos(\theta)\right)\cos(\phi)}\omega_y \\
&\quad + \frac{1}{\frac{\sin(\phi)^2\cos(\theta)}{\cos(\phi)} + \cos(\phi)\cos(\theta)}\omega_z \\
\frac{\mathrm{d}xp}{\mathrm{d}t} &= (1/m)(\sin(\theta)\sin(\psi)k(\omega_1^2 + \omega_2^2 + \omega_3^2 + \omega_4^2) - k \cdot d \cdot xp) \\
\frac{\mathrm{d}yp}{\mathrm{d}t} &= (1/m)(-\cos(\psi)\sin(\theta)k(\omega_1^2 + \omega_2^2 + \omega_3^2 + \omega_4^2) - k \cdot d \cdot yp) \\
\frac{\mathrm{d}zp}{\mathrm{d}t} &= (1/m)(-g - \cos(\theta)k(\omega_1^2 + \omega_2^2 + \omega_3^2 + \omega_4^2) - k \cdot d \cdot zp \\
\frac{\mathrm{d}x}{\mathrm{d}t} &= xp, \frac{\mathrm{d}y}{\mathrm{d}t} = yp, \frac{\mathrm{d}z}{\mathrm{d}t} = zp
\end{aligned}
$$

The full descriptions of all the models that mentioned in this paper can be found on the tool website.

Chapter 6

Pancreatic Cancer Microenvironment Model as A Multiscale Hybrid Rule-based Model and Statistical Model Checking

As mentioned in chapter 2, pancreatic cancer (PC), as an extremely aggressive disease, is the fourth leading cause of cancer death in United States [5], and the seventh cause globally [6]. It is anticipated to become the second by 2020. For years, extensive efforts have been placed on understanding the functionality of pancreatic cancer cells (PCCs), and on developing effective therapies solely targeting at them. However, the poor prognosis for PC remains largely unchanged. To turn this tide, the research focus of pancreatic cancer has been shifted from solely looking into pancreatic cancer cells towards investigating the microenvironment of the pancreatic cancer. Biologists have recently noticed that one contributing factor to the failure of systemic therapies may be the abundant tumor micro-environment. As a characteristic feature of PC, the microenvironment includes pancreatic stellate cells (PSCs), endothelial cells, nerve cells, immune cells, lymphocytes, dendritic cells, the extracellular matrix, and other molecules surrounding PCCs [144]. Over the past decade, evidence has been accumulated to demonstrate the potentially critical functions of these cells in regulating the growth, invasion, and metastasis of PC [81, 85, 86, 144]. Among these cells, PSCs and cancer-associated macrophages play primary roles during the development of PC [144]. Studies have confirmed that PSCs are the primary cells producing the stromal reaction [16, 20]. In a healthy pancreas, PSCs exist quiescently in the periacinar, perivascular, and periductal space. While, in the diseased state, PSCs will be activated by growth factors, cytokines, and oxidant stress secreted or induced by PCCs. Activated PSCs will then transform from the quiescent state to the myofibroblast phenotype. This results in their losinlipid droplets, actively proliferating, migrating, producing large amounts of extracellular matrix, and expressing cytokines, chemokines, and cell adhesion molecules. In return, the activated PSCs promote the growth of PCCs.

In this chapter, to quantitatively understand the microenvironment of PC, we construct a multicellular model. This model consists of intracellular signaling networks of pancreatic cancer cells and stellate cells respectively, and intercellular interactions among them as well. To formally

describe our multicellular and multiscale model and perform formal analysis, we extend the rule-based language BioNetGen [83] to enable the formal specification of not only the signaling network within a single cell, but also interactions among multiple cells. Specifically, we represent the intercellular level dynamics using rules with continuous variables and use Boolean Networks (BNs) to capture the dynamics of intracellular signaling networks, considering the fact that a large number of reaction rate constants are not available in the literature and difficult to be experimentally determined. Our extension saves the virtues of both BNs and rule-based kinetic modeling, while advancing the specification power to multicellular and multiscale models. We employ stochastic simulation NFsim [197] and statistical model checking (StatMC) [132] to analyze the systems properties. The formal analysis results show that our model reproduces existing experimental findings with regard to the mutual promotion between pancreatic cancer and stellate cells. The model also provides insights into how treatments latching onto different targets could lead to distinct outcomes. Using the validated model, we predict novel (poly)pharmacological strategies for improving PC treatment.

6.1 Signalling Networks within Pancreatic Cancer Microenvironment

We construct a multicellular model for pancreatic cancer microenvironment based on a comprehensive literature search. The reaction network of the model is summarized in Figure 6.1. It consists of three parts that are colored with green, blue, and purple respectively: (i) the intracellular signaling network of PCCs, (ii) the intracellular signaling network of PSCs, and (iii) the signaling molecules (such as growth factors and cytokines) in the extracellular space of the microenvironment, which are ligands of the receptors expressed in PCCs and PSCs. Note that $\rightarrow$ denotes activation/promotion/up-regulation, and $-\bullet$ represents inhibition/suppression/down-regulation.

Intracellular signaling network of PCCs

Pathways regulating proliferation

KRas mutation enhances proliferation [23]. Mutations of the KRas oncogene occur in the precancerous stages with a mutational frequency over 90%. It can lead to the continuous activation of the RAS protein, which then constantly triggers the RAF$\rightarrow$MEK cascade, and promotes PCCs' proliferation through the activation of ERK and JNK.

EGF activates and enhances proliferation [167]. Epidermal growth factor (EGF) and its corresponding receptor (EGFR) are expressed in $\sim$95% of PCs. EGF promotes proliferation through the RAS$\rightarrow$RAF$\rightarrow$MEK$\rightarrow$JNK cascade. It can also trigger the RAS$\rightarrow$RAF$\rightarrow$MEK$\rightarrow$ ERK$\rightarrow$cJUN cascade to secrete EGF molecules, which can then quickly bind to overexpressed EGFR again to promote the proliferation of PCCs, which is believed to confer the devastating nature on PCs.

HER2/neu mutation also intensifies proliferation [23]. HER2/neu is another oncogene fre-

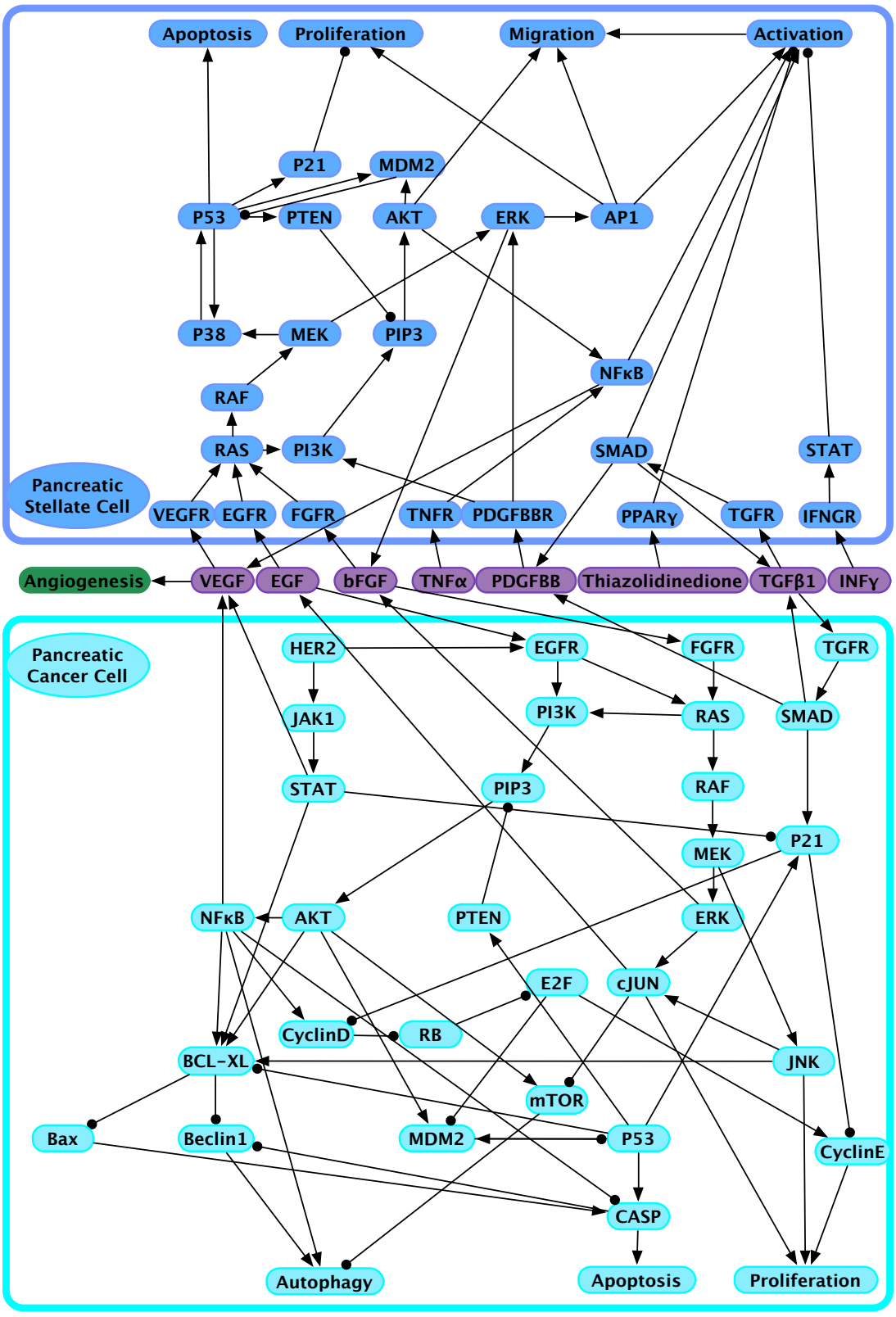


Figure 6.1: The pancreatic cancer microenvironment model

quently mutated in the initial PC formation. Mutant HER2 can bind to EGFR to form a heterodimer, which can activate the downstream signaling pathways of EGFR.

bFGF promotes proliferation [30]. As a mitogenic polypeptide, bFGF can promote proliferation through both RAF→MEK→ERK and RAF→MEK→JNK cascades. In addition, bFGF molecules are released through RAF→MEK→ERK pathway to trigger another autocrine signaling pathway in the PC development.

Pathways regulating apoptosis

Apoptosis is the most common mode of programmed cell death. It is executed by caspase proteases that are activated by death receptors or mitochondrial pathways.

TGFβ1 initiates apoptosis [194]. In PCCs, transforming growth factor β 1 (TGFβ1) binds to and activates its receptor (TGFR), which in turn activates receptor-regulated SAMDs that heterooligomerize with the common SAMD3 and SAMD4. After translocating to the nucleus, the complex initiates apoptosis in the early stage of the PC development.

Mutated oncogenes inhibit apoptosis. Mutated KRas and HER2/neu can inhibit apoptosis by downregulating caspases (CASP) through PI3K→AKT→NFκB cascade and by inhibiting Bax (and indirectly CASP) via the PI3K→PIP3→AKT→$\cdots$→BCL-XL pathway.

Pathways regulating autophagy

Autophagy is a catabolic process involving the degradation of a cell's own components through the lysosomal machinery. This pro-survival process enables a starving cell to reallocate nutrients from unnecessary processes to essential processes. Recent studies indicate that autophagy is important in the regulation of cancer development and progression and also affects the response of cancer cells to anticancer therapy [119, 146].

mTOR regulates autophagy [166]. The mammalian target of rapamycin (mTOR) is a critical regulator of autophagy. In PCCs, the upstream pathway PI3K→PIP3→AKT activates mTOR and inhibits autophagy. The MEK→ERK cascade downregulates mTOR via cJUN and enhances autophagy.

Overexpression of anti-apoptotic factors promotes autophagy [157]. Apoptosis and autophagy can mutually inhibit each other due to their crosstalks. In the initial stage of PC, the upregulation of apoptosis leads to the inhibition of autophagy. Along with the progression of cancer, when apoptosis is suppressed by the highly expressed anti-apoptotic factors (e.g. NFκB and Beclin1), autophagy gradually takes the dominant role and promotes PCC survival.

Intracellular signaling network of PSCs

Pathways regulating activation

PCCs can activate the surrounding inactive PSCs by cancer-cell-induced release of mitogenic and fibrogenic factors, such as PDGFBB and TGFβ1. As a major growth factor regulating cell functions of PSCs, **PDGFBB activates PSCs** [107] through the downstream ERK→AP1 signaling pathway. The **activation of PSCs is also mediated by TGFβ1** [107] via TGFR→SAMD pathway. The autocrine signaling of TGFβ1 maintains the sustained activation of PSCs. Further-

more, the cytokine **TNF**α, which is a major secretion of tumor-associated macrophages (TAMs) in the microenvironment, **is also involved in activating PSCs** [159] through binding to TNFR, which indirectly activates NFκB.

Pathways regulating migration

Migration is another characteristic cell function of PSCs. Activated PSCs move towards PCCs, and form a cocoon around tumor cells, which could protect the tumor from therapies' attacks [16, 86].

Growth factors promote migration. Growth factors existing in the microenvironment, including EGF, bFGF, and VEGF, can bind to their receptors on PSCs and activate the migration through the MAPK pathway.

PDGFBB contributes to the migration [173]. PDGFBB regulates the migration of PSCs mainly through two downstream pathways: (i) the PI3K$\rightarrow$PIP3$\rightarrow$AKT pathway, which mediates PDGF-induced PSCs' migration, but not proliferation, and (ii) the ERK$\rightarrow$AP1 pathway that regulates activation, migration, and proliferation of PSCs.

Pathways regulating proliferation

Growth factors activate proliferation. In PSCs, as key downstream components for several signaling pathways initiated by distinct growth factors, such as EGF and bFGF, the ERK$\rightarrow$AP1 cascade activates the proliferation of PSCs. Compared to inactive PSCs, active ones proliferate more rapidly.

Tumor suppressers repress proliferation. Similar to PCCs, P53, P21, and PTEN act as suppressers for PSCs' proliferation.

Pathways regulating apoptosis

P53 upregulates modulator of apoptosis [130]. The apoptosis of PSCs can be initiated by P53, whose expression is regulated by the MAPK pathway.

Interactions between PCCs and PSCs

The mechanism underlying the interplay between PCCs and PSCs is complex. In a healthy pancreas, PSCs exist quiescently in the periacinar, perivascular, and periductal space. However, in the diseased state, PSCs will be activated by growth factors, cytokines, and oxidant stress secreted or induced by PCCs, including EGF, bFGF, VEGF, TGFβ1, PDGF, sonic hedgehog, galectin 3, endothelin 1 and serine protease inhibitor nexin 2 [77]. Activated PSCs will then transform from the quiescent state to the myofibroblast phenotype. This results in their losinlipid droplets, actively proliferating, migrating, producing large amounts of extracellular matrix, and expressing cytokines, chemokines, and cell adhesion molecules. In return, the activated PSCs promote the growth of PCCs by secreting various factors, including stromal-derived factor 1, FGF, secreted protein acidic and rich in cysteine, matrix metalloproteinases, small leucine-rich proteoglycans, periostin and collagen type I that mediate effects on tumor growth, invasion, metastasis and resistance to chemotherapy [77]. Among them, EGF, bFGF, VEGF, TGFβ1, and PDGFBB are essential mediators of the interplay between PCCs and PSCs that have been considered in our model.

Autocrine and paracrine involving EGF/bFGF [156]. EGF and bFGF can be secreted by both PCCs and PSCs. In turn, they will bind to EGFR and FGFR respectively on both PCCs and PSCs to activate their proliferation and further secretion of EGF and FGF.

Interplay through VEGF [214]. As a proangiogenic factor, VEGF is found to be of great importance in the activation of PSCs and angiogenesis during the progression of PCs. VEGF, secreted by PCCs, can bind with VEGFR on PSCs to activate the PI3K pathway. It further promotes the migration of PSCs through PIP3$\rightarrow$AKT, and suppresses the transcription activity of P53 via MDM2.

Autocrine and paracrine involving TGFβ1 [156]. PSCs by themselves are capable of synthesizing TGFβ1, suggesting the existence of an autocrine loop that may contribute to the perpetuation of PSC activation after an initial exogenous signal, thereby promoting the development of pancreatic fibrosis.

Interplay through PDGFBB [77]. PDGFBB exists in the secretion of PCCs, whose production is regulated by TGFβ1 signaling pathway. PDGFBB can activate PSCs and initiate migration and proliferation as well.

6.2 The Modeling Language

Rule-based modeling languages are often used to specify protein-to-protein reactions within cells and to capture the evolution of protein concentrations. BioNetGen language is a representative rule-based modeling formalism [83], which consists of three components: basic building blocks, patterns, and rules. In our setting, in order to simultaneously simulate the dynamics of multiple cells, interactions among cells, and intracellular reactions, we advance the specifying power of BioNetGen by redefining basic building blocks and introducing new types of rules for cellular behaviors as follows.

Basic building blocks. In BioNetGen, basic building blocks are molecules that may be assembled into complexes through bonds linking components of different molecules. To handle multi-scale dynamics (i.e. cellular and molecular levels), we allow the fundamental blocks to be also cells or extracellular molecules. Specifically, a cell is treated as a fundamental block with subunits corresponding to the components of its intracellular signaling network. Furthermore, extracellular molecules (e.g. EGF) are treated as fundamental blocks without subunits.

As we use BNs to model intracellular signaling networks, each subunit of a cell takes binary values (it is straightforward to extend BNs to discrete models). The Boolean values - "True (T)" and "False (F)" - can have different biological meanings for distinct types of components within the cell. For example, for a subunit representing cellular process (e.g. apoptosis), "T" means the cellular process is triggered, and "F" means it is not triggered. For a receptor, "T" means the receptor is bound, and "F" means it is free. For a protein, "T" indicates this protein has a high concentration, and "F" indicates that its concentration level is below the value to regulate downstream targets.

Patterns. As defined in BioNetGen, patterns are used to identify a set of species that share features. For instance, the pattern $C(c_1)$ matches both $C(c_1, c_2 \sim T)$ and $C(c_1, c_2 \sim F)$. Using patterns offers a rich yet concise description in specifying components.

Rules. In BioNetGen, three types of rules are used to specified: binding/unbinding, phosphorylation, and dephosphorylation. Here we introduce nine rules in order to describe the cellular processes in our model and the potential therapeutic interventions. For each type of rules, we present its formal syntax followed by examples that demonstrate how it is used in our model.

Rule 1: Ligand-receptor binding

$$< Lig > + < Cell > (< Rec > \sim F) \rightarrow < Cell > (< Rec > \sim T) \ < binding\_rate >$$

Remark: On the left-hand side, the "F" value of a receptor $< Rec >$ indicates that the receptor is free. When a ligand $< Lig >$ binds to it, the reduction of number of extracellular ligand is represented by its elimination. In the meanwhile, "$< Rec > \sim T$", on the right-hand side, indicates that the receptor is not free any more. Note that, the multiple receptors on the surface of a cell can be modeled by setting a relatively high rate on the following downstream regulating rules, which indicates the rapid "releasing" of bound receptors. An example in our microenvironment model is the binding between EGF and EGFR for PCCs: "$EGF + PCC(EGFR \sim F) \rightarrow PCC(EGFR \sim T) \ 1$".

Rule 2: Mutated receptors form a heterodimer

$$\begin{aligned} &< Cell > (< Rec_1 > \sim F, < Rec_2 > \sim F) \rightarrow \\ &\quad < Cell > (< Rec_1 > \sim T, < Rec_2 > \sim T) \ \ < mutated\_binding\_rate > \end{aligned}$$

Remark: Unbound receptors can bind together and form a heterodimer. For example, in our model, the mutated HER2 can activate downstream pathways of EGFR by binding with it and forming a heterodimer: "'$PCC(EGFR \sim F, HER2 \sim F) \rightarrow PCC(EGFR \sim T, HER2 \sim T) \ 10$".

Rule 3: Downstream signaling transduction

Rule 3.1 (Single parent) upregulation (activation, phosphorylation, etc.)

$$\begin{aligned} &< Cell > (< Act > \sim T, < Tar > \sim F) \rightarrow \\ &\quad < Cell > (< Act > \sim T, < Tar > \sim T) \ \ < trate > \end{aligned}$$

Rule 3.2 (Single parent) downregulation (inhibition, dephosphorylation, etc.)

$$\begin{aligned} &< Cell > (< Inh > \sim T, < Tar > \sim T) \rightarrow \\ &\quad < Cell > (< Inh > \sim T, < Tar > \sim F) \ \ < trate > \end{aligned}$$

Rule 3.3 (Multiple parents) Downstream regulation

$$\begin{aligned} &< Cell > (< Inh > \sim F, < Act > \sim T, < Tar > \sim F) \rightarrow \\ &\quad < Cell > (< Inh > \sim F, < Act > \sim T, < Tar > \sim T) \ \ < trate > \end{aligned}$$

$$< Cell > (< Inh > \sim T, < Tar > \sim T) \rightarrow$$
$$< Cell > (< Inh > \sim T, < Tar > \sim F) \quad < trate >$$

Remark: Instead of using kinetic rules (such as in ML-Rules), our language use logical rules of BNs to describe intracellular signal cascades. Downsteam signal transduction rules are used to describe the logical updating functions for all intracellular molecules constructing the signaling cascades. For instance, Rule 3.3 presents the updating function $< Tar >^{(t+1)} = \neg < Inh >^{(t)} \times (< Act >^{(t)} + < Tar >^{(t)})$, where "$< Inh >$" is the inhibitor, and "$< Act >$" is the activator. In this manner, concise rules can be devised to handle complex cases, where there exists multiple regulatory parents. Note that our model follows the biological assumption that inhibitors hold higher priorities than activators with respect to their impacts on the target. "+" and "$\times$" in logical functions represent logical "OR" and "AND" respectively. An example in our model is that, in PCCs, $STAT$ can be activated by $JAK1$: "$PCC(JAK1 \sim T, STAT \sim T) \rightarrow PCC(JAK1 \sim F, STAT \sim T)$ 0.012" and "$PCC(JAK1 \sim T, STAT \sim F) \rightarrow PCC(JAK1 \sim F, STAT \sim T)$ 0.012".

Rule 4: Cellular processes

Rule 4.1 Proliferation

$$< Cell > (Pro \sim T) \rightarrow$$
$$< Cell > (Pro \sim F) + < Cell > (Pro \sim F, \cdots) \quad < pro\_rate >$$

Remark: When a cell proliferates, we keep the current values of subunits for the cell that initiates the proliferation, and assume the new cell to have the default values of subunits. The "$\cdots$" in the rule denotes the remaining subunits with their default values.

Rule 4.2 Apoptosis

$$< Cell > (Apo \sim T) \rightarrow Null() \quad < apop\_rate >$$

Remark: A type "Null()" is declared to represent dead cells or degraded molecules. In our model, the apoptosis of PSCs is described as "$PSC(Apo \sim T) \rightarrow Null()$
$5e-4$".

Rule 4.3 Autophagy

$$< Cell > (Aut \sim T) \rightarrow < Mol > + \cdots \quad < auto\_rate >$$

Remark: The molecules on the right-hand side of this type of rules will be released into the microenvironment due to autophagy. They are the existing molecules expressed inside this cell when autophagy is triggered.

Rule 5: Secretion

$$< Cell > (< secMol > \sim T) \rightarrow$$
$$< Cell > (< secMol > \sim F) + < Mol > \quad < sec\_rate >$$

Remark: When the secretion of "$< Mol >$" has been triggered, its amount in the microenvironment will be added by 1. Note that, we can differentiate the endogenous and exogenous

molecules by labeling the secreted "$< \ Mol \ >$" with the cell name. In our model, we have "$PCC(secEGF \sim T) \rightarrow PCC(secEGF \sim F) + EGF \ \ 2.7e-4$".

Rule 6: Mutation

$$< Cell > (< Mol > \sim < unmutated >) \rightarrow \\ < Cell > (< Mol > \sim < mutated >) \quad < mrate >$$

Remark: For mutant proteins that are constitutively active, we set a very high value to the mutation rate "mrate". In this way, we can almost keep the value of the mutated molecule as what it should be. For example, in our model, the mutation of oncoprotein Ras in PCCs is captured by "$PCC(RAS \sim F) \rightarrow PCC(RAS \sim T) \ \ 10000$".

Rule 7: Constantly over-expressed extracellular molecules

$$CancerEvn \rightarrow CancerEvn + < Mol > \quad < sec\_rate >$$

Remark: We use this type of rules to mimic the situation that the concentration of an over-expressed extracellular molecule stays in a high level constantly.

Rule 8: Degradation of extracellular molecules

$$< Mol > \rightarrow Null() \quad < deg\_rate >$$

Remark: Here, "Null()" is used to represent dead cells or degraded molecules. For instance, $bFGF$ in the microenvironment will be degraded via "$bFGF \rightarrow Null() \ \ 0.05$".

Rule 9: Therapeutic intervention

$$< Cell > (< Mol > \sim < untreated >) \rightarrow \\ < Cell > (< Mol > \sim < treated >) \quad < treat\_rate >$$

Remark: Given a validated model, intervention rules allow us to evaluate the effectiveness of a therapy targeting at certain molecule(s). Also, the well-tuned value of the intervention rate could, more or less, give indications when deciding the dose of medicine used in this therapy, based on the Law of Mass Action.

Our extension allows the BioNetGen language to be able to model not only the signaling network within a single cell, but also interactions among multiple cells. It also allows one to simulate the dynamics of cell populations, which is crucial to cancer study. Moreover, describing the intracellular dynamics using the style of BNs improves the scalability of our method by overcoming the difficulty of obtaining values of a large amount of model parameters from wet laboratory, which is a common bottleneck of conventional rule-based languages and ML-Rules. Note that, similar to other rule-based languages, our extended one allows different methods for model analysis, since more than one semantics can be defined for the same syntax.

6.3 Statistical Model Checking

Simulation can recapitulate a number of experimental observations and provide new insights into the system. However, it is not easy to manually analyze a significant amount of simulation trajectories, especially when there is a large set of system properties to be tested. Thus, for our model, we

employ statistical model checking (StatMC), which is a fully automated formal analysis technique. In this section, we provide an intuitive and brief description of StatMC. The interested reader can find more details in [132].

In general, given a system property expressed as a Bounded Linear Temporal Logic (BLTL) [132] formula and the set of simulation trajectories generated by applying the network-free stochastic simulation (NFsim) [197] to our rule-based model, StatMC estimates the probability of the model satisfying the property.

Bounded Linear Temporal Logic

Before looking into the main ideas of StatMC, we first introduce BLTL formally. As mentioned in Chapter 3.2, linear temporal logic (LTL) [178], as a modal temporal logic with modalities referring to time, is widely used to formally encode formulae about the future of paths, such as a condition will eventually be true or a condition will be true until another fact becomes true. BLTL extends LTL with time bounds on temporal operators. For example, the following BLTL formula can be used to express the specification it is not the case that within 5 seconds, variable v_0 will keep the value 1 and variable v_1 will keep the value 0 for 10 seconds.

$$\neg F^5 G^{10}(v_0 = 1 \wedge v_1 = 0)$$

where the F^5 operator encodes future 5 seconds, G^{10} expresses globally for 10 seconds, and v_0 and v_1 are state variables of the model.

The syntax of BLTL is given by:

$$\psi ::= x \sim v | \neg\psi | \psi_1 \vee \psi_2 | \psi_1 U^t \psi_2$$

where $x \in SV$ (the finite set of state variables), $\sim \in \{<, \leq, =, \geq, >\}$, $v \in \mathbb{Q}$, and $t \in \mathbb{Q}_{\geq 0}$. Note that the operators $\wedge$, F^t, and G^t referenced above can be defined as follows: $F^t\psi = True\ U^t\psi$, $G^t\psi = \neg F^t \neg\psi$, and $\psi_1 \wedge \psi_2 = \neg(\neg\psi_1 \vee \neg\psi_2)$

The semantics of BLTL is defined with respect to traces (or executions) of the model. For this work, a trace will be a simulation trajectory of our multiscale hybrid rule-based model. Formally, a trace is a sequence of time-stamped state transitions of the form $\sigma = (s_0, t_0), (s_1, t_1), \cdots$, indicating that the system moved to state s_{i+1} after duration t_i in state s_i. The fact that a trace σ satisfies the BLTL property ψ is denoted by $s \models \psi$. We denote the execution trace starting at state i by σ^i. The value of the state variable x in σ at the state i is denoted by $V(\sigma, i, x)$. The semantics of BLTL for a trace σ^k starting at the kth state ($k \in N$) is defined as follows.

- $\sigma^k \models x \sim v$ if and only if $V(\sigma, k, x) \sim v$;
- $\sigma^k \models \neg\psi$ if and only if $\sigma^k \models \psi$ does not hold;
- $\sigma^k \models \psi_1 \vee \psi_2$ if and only if $\sigma^k \models \psi_1$ or $\sigma^k \models \psi_2$;
- $\sigma^k \models \psi_1 U^t \psi_2$ if and only if there exists $i \in \mathbb{N}^+$ such that (a) $\sum_{j=k}^{k+i-1} t_j \leq t$, (b) $\sigma^{k+i} \models \psi_2$, and (c) for each $0 \leq j < i$, $\sigma^{k+j} \models \psi_1$.

Statistical Model Checking
In this work, we use StatMC to estimate the probability with which a model satisfies a given bounded LTL property. Essentially, given our model M, we first run NFsim on it. For each generated trajectory with a predefined small length, we run the trace checker on it against the given bounded LTL formula. The trace checker will decide whether to stop the simulation for this certain trajectory and pass the checking result to the statistical tester, or to continue the simulation for this trace by another k steps. Checking results of multiple trajectories will be used by the statistical tester to estimate the probability. The tester will decide when to stop the whole simulation, and return the final estimated probability with a preset small error bound. The whole checking process is illustrated in Figure 6.2.

In detail, since the underlying semantic model of the stochastic simulation method NFsim that we used for our model is essentially a discrete-time Markov chain, we need to verify stochastic models. StatMC treats the verification problem for stochastic models as a statistical inference problem, using randomized sampling to generate traces (or simulation trajectories) from the system model, and then performing model checking and statistical analysis on those traces. For a (closed) stochastic model and a BLTL property ψ, the probability p that the model satisfies ψ is well defined (but unknown in general). For a fixed $0 < \theta < 1$, we ask whether $p \leq \theta$, or what the value of p is. In StatMC, the first question is solved via hypothesis testing methods, while the second via estimation techniques. Intuitively, hypothesis tests are probabilistic decision procedures, i.e., algorithms with a yes/no reply, and which may give wrong answers. Estimation techniques instead compute (probabilistic) approximations of the unknown probability p. The main assumption of StatMC is that, given a BLTL property ψ, the behavior of a (closed) stochastic model can be described by a Bernoulli random variable of parameter p, where p is the probability that the system satisfies ψ. It is known that discrete-time Markov chains satisfy this requirement [211]. Therefore StatMC can be applied to our setting. More specifically, given σ is a system execution and ψ a BLTL formula, we have that $Prob\{\sigma|\sigma \models \psi\} = p$, and the Bernoulli random variable mentioned above is the following function M defined as follows: $M(\sigma) = 1$ if $\sigma \models \psi$, or $M(\sigma) = 0$ otherwise. Therefore, M will be 1 with probability p and 0 with probability $1 - p$. In general, StatMC works by first obtaining samples of M, and then by applying statistical techniques to such samples to solve the verification problem.

6.4 Results and Discussion

In this section, we present and discuss formal analysis results for our pancreatic cancer microenvironment model. The model file is available at `http://www.cs.cmu.edu/~qinsiw/mpc_model.bngl`. All the experiments reported below were conducted on a machine with a 1.7 GHz Intel Core i7 processor and 8GBRAM, running on Ubuntu 14.04.1 LTS. In our experiments, we use Bayesian sequential estimation with 0.01 as the estimation error bound, coverage probability 0.99, and a uniform prior (α = β = 1). The time bounds and thresholds given in following properties are determined by considering the model's simulation results. The parameters in our model include initial state (e.g. abundance of extracellular molecules) and reaction rate constants. The

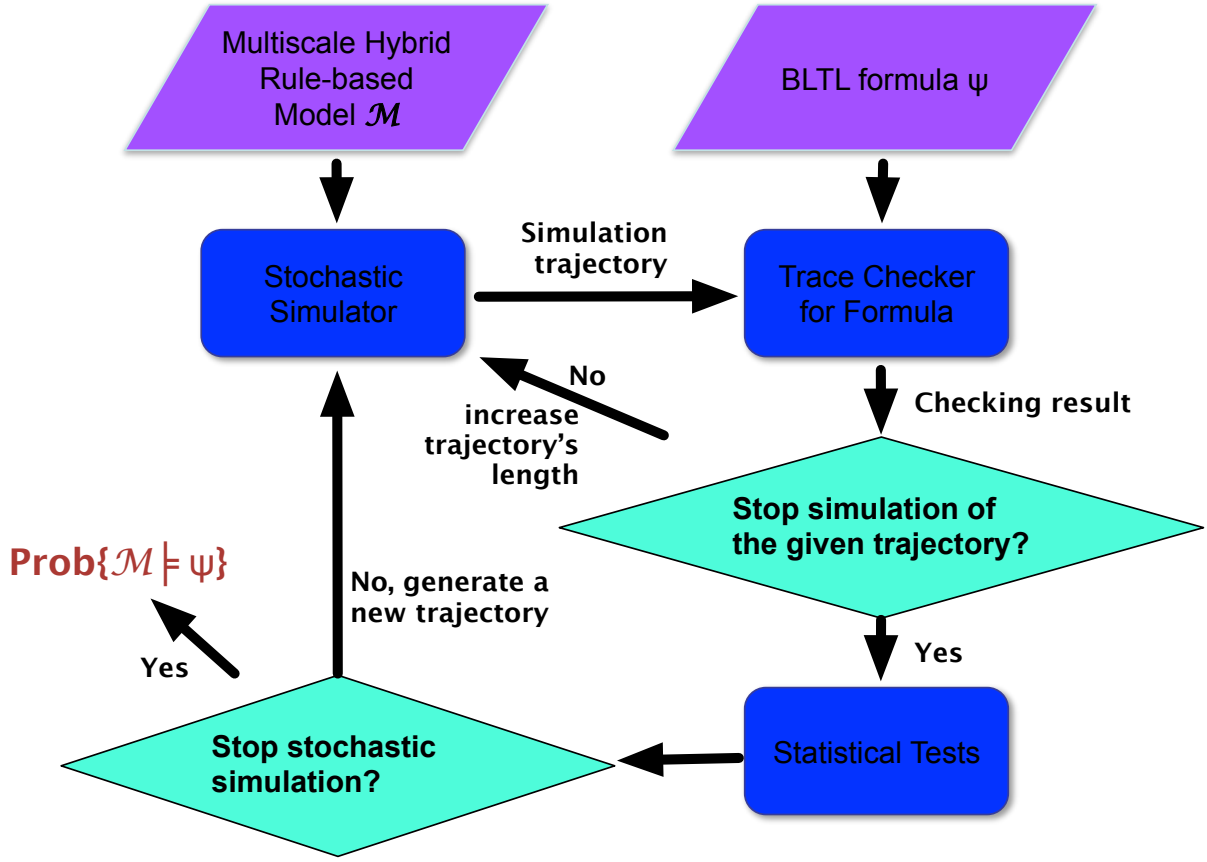


Figure 6.2: The statistical model checking process for the pancreatic cancer microenvironment model

initial state was provided by biologists based on wet-lab measurements. The rate constants were estimated based on the general ones in the textbook [13]. The results in scenario I & II demonstrate that using these parameters the model is able to reproduce key observations reported in the literature. We also performed a sensitivity analysis and the results show that the system behavior is robust to most of the parameters (the two sensitive parameters have been labeled in our model file).

Scenario I: mutated PCCs with no treatments

In scenario I, we validate our model by studying the role of PSCs in the PC development.

Property 1: This property aims to estimate the probability that the population of PCCs will eventually reach and maintain in a high level.

$$Prob_{=?}\ \{(PCCtot = 10) \wedge F^{1200}\ G^{100}\ (PCCtot > 200)\}$$

First, we take a look at the impact from the presence of PSCs on the dynamics of PCC population. As shown in Table 6.1, with PSCs, the probability of the number of PCCs reaching and keeping in a high level ($Pr = 0.9961$) is much higher than the one when PSCs are absent ($Pr = 0.405$). This indicates that PSCs promote PCCs proliferation during the progression of PC. This is consistent with experimental findings [16, 77, 214].

Property 2: This property aims to estimate the probability that the number of migrated PSCs will eventually reach and maintain in a high amount.

$$Prob_{=?}\ \{(MigPSC = 0) \wedge F^{1200}\ G^{100}\ (MigPSC > 40)\}$$

We then study the impacts from PCCs on PSCs. As shown in Table 6.1, without PCCs, it is quite unlikely (($Pr = 0.1191$) for quiescent PSCs to be activated. While, when PCCs exist, the chance of PSCs becoming active (($Pr = 0.9961$) approaches to 1. This confirms the observation [107] that, during the development of PC, PSCs will be activated by growth factors, cytokines, and oxidant stress secreted or induced by PCCs.

Property 3: This property aims to estimate the probability that the number of PCCs entering the apoptosis phase will be larger than the number of PCCs starting the autophagy process and this situation will be reversed eventually.

$$Prob_{=?}\ \{F^{400}\ (G^{300}\ (ApoPCC > 50 \wedge AutoPCC < 50)$$

$$\wedge F^{700}\ G^{300}\ (ApoPCC < 50 \wedge AutoPCC > 50))\}$$

We are also interested in the mutually exclusive relationship between apoptosis and autophagy for PCCs reported in [119, 157]. In detail, as PC progresses, apoptosis firstly overwhelms autophagy, and then autophagy takes the leading place after a certain amount of time. This situation is described as property 3 and its estimated probability is close to 1 (see Table 6.1).

Property 4: This property aims to estimate the probability that, it is always the case that, once the population of activated PSCs reaches a high level, the number of migrated PSCs will also increase.

$$Prob_{=?}\ \{G^{1600}\ (ActPSC > 10 \rightarrow F^{100}\ (MigPSC > 10))\}$$

Property	Estimated Prob	# Succ	# Sample	Time (s)	Note
Scenario I: mutated PCCs with no treatments					
1	0.4053	10585	26112	208.91	w.o. PSCs
	0.9961	256	256	1.83	w. PSCs
2	0.1191	830	6976	49.69	w.o. PCCs
	0.9961	256	256	1.75	w. PCCs
3	0.9961	256	256	5.21	-
4	0.9961	256	256	4.38	-
Scenario II: mutated PCCs with different exsiting treatments					
5	0.0004	0	2304	17.13	cetuximab and erlotinib
	0.0012	10	9152	68.67	gemcitabine
	0.7810	8873	11360	114.25	nab-paclitaxel
	0.8004	7753	9686	73.83	ruxolitinib
Scenario III: mutated PCCs with blocking out on possible target(s)					
6	0.0792	38363	484128	3727.99	w.o. inhibiting ERK in PSCs
	0.9822	2201	2240	17.37	w. inhibiting ERK in PSCs
7	0.1979	3409	17232	136.39	w.o. inhibiting ERK in PSCs
	0.9961	256	256	2.01	w. inhibiting ERK in PSCs
8	0.2029	2181	10752	92.57	w.o. inhibiting MDM2 in PSCs
	0.9961	256	256	2.18	w. inhibiting MDM2 in PSCs
9	0.0004	0	2304	15.77	w.o. inhibiting RAS in PCCs and ERK in PSCs
	0.9961	256	256	3.15	w. inhibiting RAS in PCCs and ERK in PSCs
10	0.9797	1349	1376	11.98	w.o. inhibiting STAT in PCCs and NFκB in PSCs
	0.1631	1476	9056	81.61	w. inhibiting STAT in PCCs and NFκB in PSCs

Table 6.1: Statistical model checking results for properties under different scenarios

One reason why PC is hard to be cured is that activated PSCs will move towards mutated PCCs, and form a cocoon for the tumor cells, which can protect tumor from attacks caused by therapies [16, 86]. We investigate this by checking property 4, and obtain an estimated probability approaching to 1 (see Table 6.1).

Scenario II: mutated PCCs with different existing treatments

Property 5: This property aims to estimate the probability that the population of PCCs will eventually drop to and maintain in a low amount.

$$Prob_{=?}\ \{(PCCtot = 10) \wedge F^{1200}\ G^{400}\ (PCCtot < 100)\}$$

Property 5 means that, after some time, the population of PCCs can be maintained in a comparatively low amount, implying that PC is under control. We now consider 5 different drugs that are widely used in PC treatments - cetuximab, erlotinib, gemcitabine, nab-paclitaxel, and ruxolitinib, and estimate the probabilities for them to satisfy property 5. As shown in Table 6.1, monoclonal antibody targeting EGFR (cetuximab), as well as direct inhibition of EGFR (erlotinib) broadly do not provide a survival benefit in PCs. Inhibition of MAPK pathway (gemcitabine) has also not been promising. These results are consistent with clinical feedbacks from patients [3]. While, strategies aiming at depleting the PSCs in PCs (i.e. nab-paclitaxel) can be successful (with an estimated probability 0.7810). Also, inhibition of Jak/Stat can be very promising (with an estimated probability 0.8004). These results are supported by [213] and [127], respectively.

Scenario III: mutated PCCs with blocking out on possible target(s)

Scenario I and II have demonstrated the descriptive and predictive power of our model. In scenario III, we use the validated model to identify new therapeutic strategies targeting molecules in PSCs. Here we report 4 potential target(s) of interest from our screening.

Property 6: This property aims to estimate the probability that the number of PSCs will eventually drop to and maintain in a low level.

$$Prob_{=?}\ \{(PSCtot = 5) \wedge F^{1200}\ G^{400}\ (PSCtot < 30)\}$$

Property 7: This property aims to estimate the probability that the population of migrated PSCs will eventually stay in a low amount.

$$Prob_{=?}\ \{(MigPSC = 0) \wedge F^{1200}\ G^{100}\ (MigPSC < 30)\}$$

The verification results of these two properties (Table 6.1) suggest that inhibiting ERK in PSCs not only lowers the population of PSCs, but also inhibits PSC migration. The former function can reduce the assistance from PSCs in the progression of PCs indirectly. The later one can prevent PSCs from moving towards PCCs and forming a cocoon to protect PCCs against cancer treatments.

Property 8: This property aims to estimate the probability that the number of PSCs entering the proliferation phase will eventually be less than the number of PSCs starting the apoptosis programme and this situation will maintain.

$$Prob_{=?}\ \{F^{1200}\ G^{400}\ ((PSCPro - PSCApop) < 0)\}$$

The increased probability (from 0.2029 to 0.9961 as shown in Table 6.1) indicates that inhibiting MDM2 in PSCs may reduce the number of PSCs by inhibiting PSCs' proliferation and/or promoting their apoptosis. Similar to the former role of inhibiting ERK in PSCs, it can help to treat PCs by alleviating the burden caused by PSCs.

Property 9: This property aims to estimate the probability that the number of bFGF will eventually stay in such a low level.

$$Prob_{=?}\ \{F^{1200}\ G^{400}\ (bFGF < 100)\}$$

As mentioned in property 5, 6, and 7, inhibiting RAS in PCCs can lower the number of PCCs, and downregulating ERK in PSCs can inhibit their proliferation and migration. Besides these, we find that, when inhibiting RAS in PCCs and ERK in PSCs simultaneously, the concentration of bFGF in the microenvironment drops (see Table 6.1). As bFGF is a key molecule that induces proliferation of both cell types, targeting RAS in PCCs and ERK in PSCs at the same time could synergistically improve PC treatment.

Property 10: This property aims to estimate the probability that the concentration of VEGF will eventually reach and keep in a high level.

$$Prob_{=?}\ \{F^{400}\ G^{100}\ (VEGF > 200)\}$$

Furthermore, inhibiting STAT in PCCs and NFκB in PSCs simultaneously postpones and lowers the secretion of VEGF (see Table 6.1). VEGF plays an important role in the angiogenesis and metastasis of pancreatic tumors. Thus, the combinatory inhibition of STAT in PCCs and NFκB in PSCs may be another potential strategy for PC therapies.

Chapter 7

Joint Efforts of Formal Methods and Machine Learning to Automate Biological Model Design

The works discussed in previous chapters have demonstrated that constructing and analyzing appropriate models can help in explaining biological systems that we are studying via presenting their core dynamics, so as to allow us to discover new questions and challenge existing theories. However, the creation of models often requires intense human effort. For example, to construct the multicellular and multiscale model of pancreatic cancer environment, I had read about 300 related papers, and had weekly meetings with experts to understand the behavior of the system and to handle conflicting statements found in distinct publications. This laborious process results in a slow development of models, let alone validating and extending them with recently reported findings in newly published literature. To order to allow researchers to re-use existing findings about a certain biological system in a comprehensive and timely manner, we really need a framework that provides functionalities for the automation of information extraction from existing literature, for the correct and smart information assembly, and for the accurate and efficient model analysis.

Over the last decade, several automated reading engines have been developed and successfully adopted to extract interactions among biological entities from literature (e.g. [208]). These auto-readers are quite efficient. That is, they are capable of finding hundreds of thousands of interactions from thousands of papers in a few hours [208]. The existence of these auto-readers, together with a given set of keywords, can promise to offer a vast amount of related system information extracted from published papers, which will be used for the later model assembly.

However, in order to accurately and efficiently incorporate these pieces of knowledge into a model, selection methods are needed to choose correct and useful ones from the given huge amount of extracted information. The integrated model should satisfy important system properties. When a baseline model is used as the initial model, the extended model should retain system properties that are satisfied by the baseline model, or even reflect other system properties that the baseline model fails to satisfy. Moreover, it is also useful to detect extended models where minimal interventions

in the model can lead to significant changes in outcomes.

To achieve this, in this chapter, we propose and implement a pipeline framework *LEaRn*, as illustrated in Figure 7.1. As the first step, given related papers and a set of keywords, the auto-reader REACH [208] is used to extract casual relations with regards to biological entities of interest. Then, together with a baseline model, the extracted relations are preselected using different heuristics that will be discussed in detail in the later section. A set of extended models will be generated through this preselection phase. Afterwards, these generated models will be checked against a set of system properties as the final selection standard. This pipeline offers a platform to utilize previously reported findings about a certain system, to validate existing knowledge, and to test new hypotheses. To demonstrate the feasibility of this proposed framework, we have looked into a case study with regards to the pancreatic cancer microenvironment.

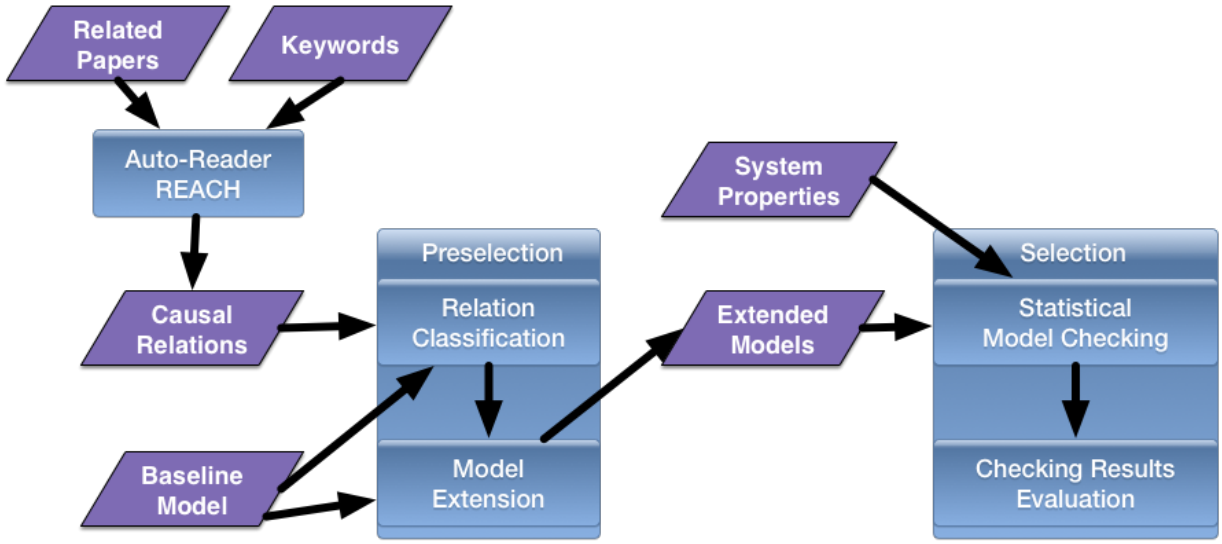


Figure 7.1: The framework of *LEaRn*

7.1 Outputs from Auto-reading

Auto-reading output and data structure

For the current version of our pipeline framework *LEaRn*, the biological systems that we are studying are cellular signaling networks. Cell signaling is part of a complex system of communication that governs basic activities of cells and coordinates cell actions. Signal transduction along a pathway occurs when an extracellular molecule activates a specific receptor located on the cell surface or sometimes inside the cell. This receptor then triggers a chain of events within the cell, creating a response. Multiple pathways interact with one another to form a network.

Within published papers, descriptions of interactions forming a signaling network can be classified into three groups according to how complete the information about a certain interaction can

be found in papers. The first group contains qualitative relations. One example can be "the signaling enzymes encoded by PIK3CA and BRAF are, in part, regulated by direct binding to activated forms of the Ras protein" in [189]. From this type of relations, we can only obtain information indicating that one biological entity places either positive or negative impact on another entity. Relations belonging to the second group, named as semi-quantitative relations, also provide rough information about the amounts of involved entities, the extent of a certain impact, location where a certain interaction takes place, and so on. For instance, we found "we treated CHO-KI cells expressing EGFR T669A with HRG ligand to induce maximal ERBB3 phosphorylation" in [207]. The last group, quantitative relations, offers complete and precise information about all the details of a certain interaction. This kind of relations are the ones that we are expecting from the auto-reading. But, unfortunately, in reality, only a small part of auto-reading outputs can be grouped as quantitative relations.

The automated reading engine REACH [208] that is adopted in our pipeline framework can extract events in the form of frames that contain an interaction with two biological entities. Considering that only a very small amount of extracted interactions have complete quantitative information, in *LEaRn*, we choose to use a triple $< P, Q, +/- >$ to represent individual interactions. Within a triple, P and Q are biological entities, such as genes, proteins, and cell functions. "+" means that P places a positive impact on Q, which can be activating, phosphorylating, and so on. While, "-" indicates that the impact from P to Q is negative, including inhibition, dephosphorylation, etc. We use this data structure to capture the discrete structure of signaling networks.

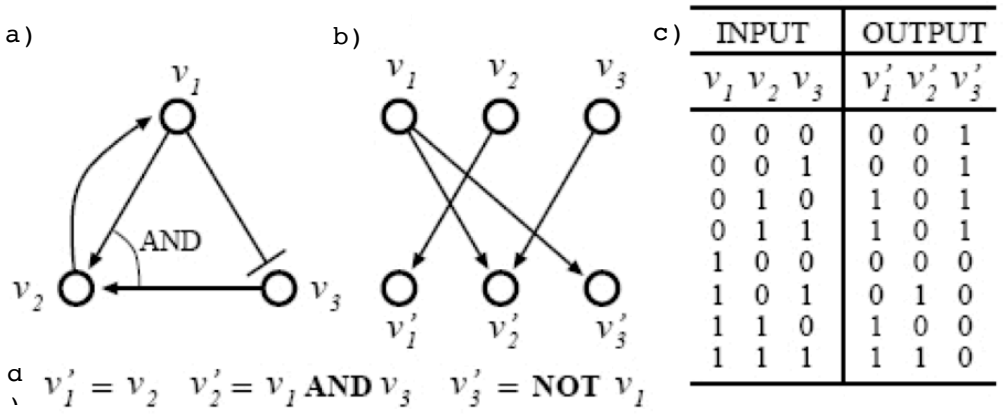
Modeling formalism for baseline and extended models

To be consistent with of the modeling format used to represent extracted relations, in *LEaRn*, Boolean Networks (BNs) is used to describe both the baseline model and extended models. As discussed in Chapter 2.2, when using BNs to capture the dynamics of signaling networks, each node in a BN represents a biological entity in a corresponding signaling network, and can have binary values. The state evolution of a node from discrete time point t to $t + 1$ is described by a Boolean updating function involving this certain node and its parent nodes. To recall how BNs can be used to model signaling networks, we provide the following simple example. In this example pathway see Figure 7.2a, v_1 is activated by v_2, v_2 is activated by both v_1 and v_3, and v_3 is inhibited by v_1. By describing it as a boolean network, v_1, v_2, and v_3 are treated as boolean variables whose next time values can be computed by using boolean functions listed in Figure 7.2d.

7.2 Preselection on Causal Relations and Model Generation

Classification of causal relations

Given a set of well formatted causal relations, to carry out the model assembly, one can start with or without a baseline model. When there is no baseline model, extracted relations are usually needed to be scored via counting the occurrence of a certain relation learned from multiple papers, total citation of papers reporting this relation, and so on. After filtering out incomplete or duplicated relations, and handling conflicting ones, a set of relations can be chosen, according to their scores,



INPUT			OUTPUT		
v_1	v_2	v_3	v_1'	v_2'	v_3'
0	0	0	0	0	1
0	0	1	0	0	1
0	1	0	1	0	1
0	1	1	1	0	1
1	0	0	0	0	0
1	0	1	0	1	0
1	1	0	1	0	0
1	1	1	1	1	0

Figure 7.2: Boolean network model of a simple signaling network

their relationships with biological entities of interest, and their connectivity, to construct the final model(s). In the current version of *LEaRn*, we consider the situation where there is a validated baseline model.

Then, given a baseline model, extracted relations can be first classified into three types according to their relationship to a given model as following.

- Corroborations: an interaction extracted from papers confirms one interaction existing in the baseline model.
- Extensions: a relation learned from literature adds new information into the baseline model. According to the type of newly added information, relations in the "Extension" category can be further classified into the following three groups.
  - When the new information is a new connection between two biological entities within the baseline model, it means that both elements within this certain relation are already in the baseline model. Adding this kind of extension usually causes a direct influence on the behavior of the resulting model, as structural changes of a signaling network may lead to a significant difference in the regulatory behavior.
  - When the new information is a relation between a biological entity in the baseline model and a new entity, there are two cases. In cases where the regulated element is not in the baseline model, the regulated element will just hang from a pathway without having direct influence on the model. While, in cases where the regulator is outside the baseline model, the regulator can act as a new model input, allowing for the additional network control.

  - When the new information is a relation between two biological entities not mentioned in the baseline model, adding such an interaction alone into the baseline model will not affect the behavior of the model. However, when we are considering multiple extensions connecting this interaction with elements in the baseline model concurrently, additional regulatory pathways will be constructed to place impacts on the model behavior.

- Contradictions: an interaction from auto-reading suggests a conflicting mechanism mentioned in the baseline model. For example, in the baseline model, entity A can activate B. While, an extracted relation says the opposite, i.e. A inhibits B in this system.

In this work, we only consider relations belonging to the "Extension" group, that is, new interactions that can be added to the model. Using the information of corroborations to add weights / scores to relations, and handling contradictions are parts of our future work.

Heuristics for model generation

Although we only consider extracted relations belonging to the "Extension" group, there are still a vast amount of interactions. Given n newly learned causal relations, there will be 2^n possible extended models if we enumerate all configurations of whether or not to add a new relation. This exponentially growing number is impossible to handle, therefore, we need heuristic methods to search for suitable configurations of model extensions. Note that, there are several ways to design selection heuristic, such as using scoring functions, considering the connectivity of the extended models, and so on. In here, we introduce four heuristics by considering whether and how selected relation will influence the satisfiability of the resulting model against a given set of important system properties.

Since the given set of system properties is key to our selection heuristics, we define all the biological entities mentioned in this given set of properties as "elements of interest". Note that, in general, the set of "elements of interest" can also be defined by user, depending on the questions asked or hypotheses tested. Then, a concept "layer" is introduced for individual biological entities mentioned either in the baseline model or in extracted interactions. The value of "layer" for each entity is the length of the shortest path connecting this certain entity with any element in the set of "elements of interest", and can be computed iteratively. That is, elements within "elements of interest" are in layer 0. Layer 1 contains direct regulators of elements in layer 0, which are not listed in layer 0. Similarly, layer $i + 1$ includes direct regulators of elements in layer i, which are not listed in layer $\{0, 1, \cdots, i\}$. With these two concepts, we propose four heuristics to select extension configurations.

- **Cumulative parent-set with direct extensions ($CD(n)$)**: using this heuristic, given a layer n, we select all extracted interactions that contains any element from layer 0 up to layer n, and add the selected relations into the baseline model. For each different n, one extended model can be generated until the set of selected relations cannot be extended anymore.

- **Non-cumulative parent-set with direct extensions ($ND(n)$)**: given a layer n, unlike CD, ND only chooses relations containing elements in layer n. One reason to use this heuristic

is that, sometimes it is interesting to the influence only from the n^{th} layer to the resulting model, so as to identify individual extension layers that may cause significant changes to the performance considering different properties. The other reason is that, biological entities mentioned in extracted interactions sometimes impact "elements of interest" indirectly through long interacting paths.

- **Cumulative parent-set with indirect extensions ($CI(n)$)**: for the previous two heuristics, we find biological entities in each layer only by looking for direct regulators of elements in the previous layer. While, when using CI, we look for indirect regulators as well. In detail, given a layer n, we select all extracted relations that place either direct or indirect impact on elements in layer n. This heuristic usually includes pathways outside the baseline model more often then the other methods.

- **Non-cumulative parent-set with indirect extensions ($NI(n, m)$)**: this method is the combination of CI and ND. The goal of this heuristic is to provide information about the influence caused by relations belonging to m layers containing indirect edges, starting from the n^{th} layer. In other words, we first find biological entities in the n^{th} layer using the ND heuristic, and perform the operation of CI for m times to find all interactions we are interested in. The reason why we propose this heuristic is that, it is important to consider the impacts from interactions happening in the nearby cells on the signaling network consisting of biological entities in "elements of interest".

7.3 Selection using Statistical Model Checking

After obtaining a set of extended models using the above heuristics, the final model selection is carried out by applying statistical model checking to each generated model against a set of important system properties for a certain biological system. Given the BN representations of these extended models, although simulating these logical models is known to be able to recapitulate certain experimental observations [164], verifying simulation results against the given properties manually is tedious and error-prone, especially when the number of models or properties are large. A feasible way to tackle this problem is to use formal methods. In our pipeline framework, statistical model checking (StatMC) is adopted. As discussed in Chapter 6.3, StatMC, as a fully automated formal analysis technique, can be used to estimate the probability with which a model satisfies a given bounded LTL property. As illustrated in Figure 6.2 in Chapter 6.3, StatMC starts with carrying out the stochastic simulation on the given model. In this work, a publicly available stochastic simulator [1, 165] is used on our extended BN models. In the simulator, there are several distinct simulation schemes that can be used to consider different timing and element update approaches occurring in biological systems. The simulation scheme that we use for this work is called "Random Sequential Step-Based Uniform". That is, in each discrete simulation step, one element in the given model is chosen randomly. Then, its Boolean updating function is applied to compute the new value of this chosen element. Before starting the stochastic simulation, the upper bound of sequential steps is defined. In the case using the *uniform* updating approach, all model elements have the same

probability of being chosen. Values of variable in each step, starting from the initial state till the given upper bound, are recorded for the later trace checking against a given bounded LTL property.

Since the underlying semantic model of the stochastic simulation method that we use for the extended BN models is essentially a discrete-time Markov chain, the verification problem is to compute the probability with which a given temporal logic formula is satisfied by the system. For large and complex models, numerical methods aiming at computing the exact probability suffer from the state explosion problem. While, StatMC, instead of searching the entire state space, uses statistical testing methods to provide an efficient way to estimate the probability with a preset small error bound.

7.4 Result and Discussion

The framework is implemented in Python. The simulator described in Chapter 7.3 is implemented in Java [1]. We use PRISM [150] as our statistical model checker, which is a C++ tool for formal modeling and analysis of stochastic systems. Evaluating a model against one property, including running the simulations, takes about 10 minutes on a regular laptop. The other components in the framework take less than 1 minute.

Baseline model

The system that we studied is pancreatic cancer microenvironment, especially the interplay between pancreatic cancer cells (PCCs) and pancreatic stellate cells (PSCs). We use the BN representation of our microenvironment model (see Figure 6.1 in Chapter 6.1) as the baseline model. This model contains three parts - intracellular signaling networks of PCC and PSC, and the their interplay supported by extracellular molecules in the tumor microenvironment. In this model, several cellular functions, such as autophagy, apoptosis, proliferation, migration, are also implemented as variables inside the model, which allows for better understanding of the system's behavior. In detail, there are 30 variables encoding intracellular molecules in PCC and 3 variables encoding the cell functions of PCC. For PSC, there are 24 variables for intracellular molecules and 4 variables for its cell functions. In extracellular microenvironment, there are 8 variables encoding extracellular growth factors, and 1 environmental function variable. In total, there are 70 variables and 114 interactions in the baseline model. The interaction rules of this model are summarized in Table 1 in the Supplementary material (`http://ppt.cc/XlWF7`).

System properties as the selection standard

To demonstrate the feasibility of our framework, we apply the proposed pipeline framework to the study of the interplay between PCCs and PSCs in the pancreatic cancer microenvironment. We identify a set of system properties according to the experimental observations reported with regards to this biological system, and extract the set of "elements of interest" from the given set of properties. As listed in Table 7.1, we are interested in observing the changes with respect to important growth factors in the tumor microenvironment, oncoproteins in both PCCs and PSCs, tumor suppressors in PCCs, and cell functions of PCCs and PSCs.

Auto-reading outputs and extended models

We used the REACH automated reading engine [2] output produced from 13,000 papers in publicly available domain. This output consists of 500,000 event files, with 170,000 possible extensions of our model (other events are corroborations or contradictions). Although there are 170,000 model extensions produced by reading, many of them are repetitions, and some of the reading outputs were missing one of the interaction participants. Therefore, in this work we used overall 1232 different interactions from reading output, which could lead to 2^{1232} possible models. Studying all possible model versions is impractical, and therefore, we used the four extension methods described in Chapter 7.2, to generate 46 different models. Using the CD method, we generated 2 models by having 1 or 2 layers. For ND, the number of layers we considered varied between 1 and 10, which resulted in 10 models. With CI, we used either $0, 1, 2$ or 3 layers, which led to 4 different models. Finally, for NI, we have n ranges from 1 layer to 10 layer, and m ranges from 1 to 3, resulting in 30 models. We also test the model with all extensions being added to the baseline model.

Fig. 7.3(a) summarizes results of our extension methods on 1232 interactions with respect to new node connections to the model: (i) number of new nodes regulating baseline model elements, not regulated by baseline model elements (dark blue); (ii) number of new nodes regulating baseline model elements, not regulated by any element, baseline or new (red); (iii) number of new nodes regulated by baseline model elements, not regulating any elements in the baseline model (yellow); (iv) number of new nodes regulated by baseline model elements, not regulating any element, baseline or new (purple); (v) number of new nodes inserted into existing pathway - new regulators of baseline model elements that are also regulated by baseline model elements (green); (vi) number of new nodes as intermediate elements of new pathways when multiple extensions are connected (light blue); (vii) total number of all elements used in the extension method (dark red). In Fig. 7.3(a), four different sections can be observed, and each section corresponds to one of the extension methods. Each method has its unique feature. For example, the ND method only includes relationships relevant to one layer, and this makes the number of new elements added to the model significantly smaller than other methods. Also, the light blue nodes tell us the number of newly added elements that are in a newly formed pathway. Since CD and ND do not include indirect parent interactions, we can see that the number of elements in new pathway is 0. While in CI and NI, we can tell that indirect interactions are included. The numbers within one method show higher similarity, but we can still observe some patterns. For example, the cumulative parent-set methods, CD and CI show an increase in the number of new nodes when more layers are considered. Furthermore, since NI has cumulative parents when they finish the noncumulative part, they also experience an increase when the step of noncumulative part is fixed. The numbers saturate at around 600, which is due to the limited size of baseline model and extensions we have. This is also the reason we choose to perform cumulative steps for at most 3 steps.

In general, choosing the method to extend the model depends on the scenario a user is interested in. For example, if the focus is on the regulation of a specific element, one can track down each layer of parents using ND, and see the change of the model after modifying that specific layer. On the other hand, if the goal is to include as many new stimuli as possible with a fewer number of layers, cumulative methods such as CI or CD will fit better. We selected 20 elements as part of the

base layer, since these elements appear in properties that we are testing, leading to relatively large base layer given the size of the baseline model. Therefore, by incorporating elements related to more than one layer, we capture almost all extensions related to the baseline model. Thus, the 'All In' method, which adds all extension interactions to the baseline model at once, does not change the counts shown in Fig. 7.3(a), when compared to many cases of CD, CI and NI methods.

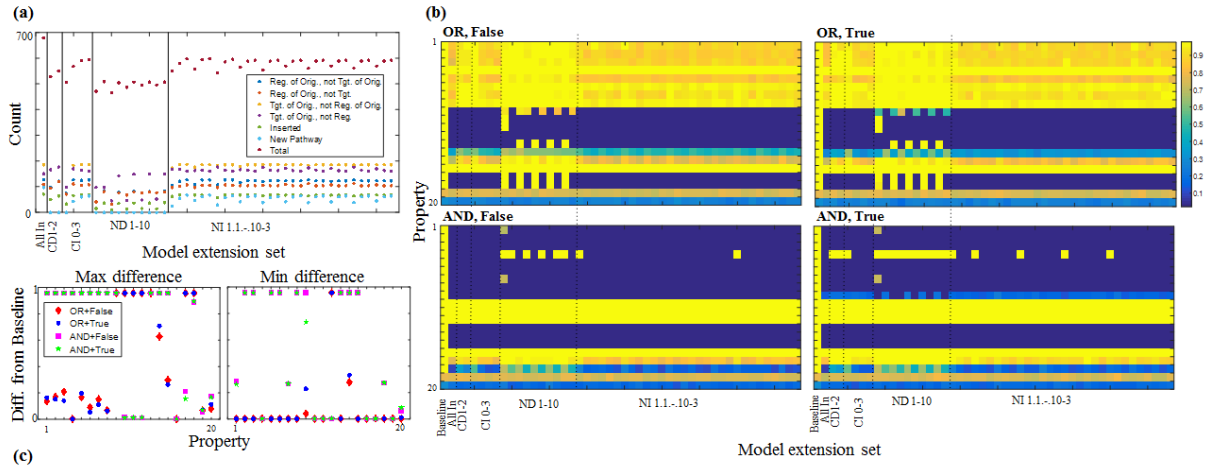


Figure 7.3: (a) Counts for newly added elements with certain structure (**Reg**. - regulator, **Tgt**. - regulated element, **Orig**. - baseline model elements, **New** - newly added element). All models studied are listed on x-axis, and y-axis is the count of new elements having certain structure. (b) Results of statistical model checking of 20 properties in 68 different models. Each entity in x-axis is a model, and each row is the estimated probability for the corresponding property. (c) The Max and min difference from the baseline model of each property [153]

Impact of model extension on system properties

Fig. 7.3(b) shows the results of testing 48 models (baseline + All-In + 46 extended models) with different extension method (AND/OR) and different initialization of the newly added elements ($True$/$False$) against the 20 properties in Table 7.1. The values displayed are the estimated probabilities of each property. Just like the basic numbers of each model, different extension methods lead to different results of the properties. For example, we can see that the results from ND are different from other methods. The reason is that each ND method only deals with one layer at a time, and it will not insert new edges between elements mentioned in the properties. This leads to a more conservative extension. Also, for example, there are differences between OR-based ND models in properties 9 to 13 or property 4 in AND-based ND models, which are related to Inhibition of tumor suppressors and autophagy in PCCs. By comparing the extension interactions added to those models, we found that the EGF (Epidermal Growth Factor) pathway plays the most important role. The p21 (regulator of cell cycle progression) pathway also influences the difference.

If we compare the models with different initialization of newly added nodes, we can see the results are actually quite similar. This means that the model is mostly influenced by the input elements in the baseline model, and to some degree, it tells the robustness of the original model.

On the other hand, if we compare extending the models with OR operations and those with AND operations, there is a huge difference. But the interesting part is that the behavior of models with the two types of extensions is opposite. They behave similarly only in properties 9, 13, 16, 19 and 20, while differently in all other 15 properties. This shows a drastic difference between AND-based and OR-based extension, and can be further designed according to the property we want to fit. Fig. 7.3(c) shows the maximum / minimum difference compared to baseline that each model can achieve for each property. If a property probability is low in both max and min difference, it is relatively conservative to the extension interaction. An example is property 16, which depicts the relationship between p53 and Apoptosis. On the other hand, if a property probability is high in both max and min difference, it is a property susceptible to change value with extensions.

#	Property	Description
Growth factors in the tumor microenvironment		
1	F[1000] (G[10000] VEGF)	Within 1000 time units, the concentration of VEGF (/ bFGF / PDGFBB / TGFβ1) in the tumor microenvironment will eventually reach a high amount, and stay in this high level for at least another 10000 time units.
2	F[1000] (G[10000] bFGF)	
3	F[1000] (G[10000] PDGFBB)	
4	F[1000] (G[10000] TGFβ1)	
Oncoproteins in pancreatic cancer and stellate cells		
5	F[1400] (G[10000] PCCHER2)	Within 1400 time units, the concentration of HER2 in PCCs (/ RAS in PCCs / VEGFR in PSCs / ERK in PCCs) will eventually reach a high amount, and stay in this high level for at least another 10000 time units.
6	F[1400] (G[10000] PCCRAS)	
7	F[1400] (G[10000] PSCVEGFR)	
8	F[1400] (G[10000] PCCERK)	
Tumor suppressors in pancreatic cancer cells		
9	F[1000] (PCCP21 ∧ F[2000] (G[10000] (! PCCP21)))	The concentration of P21 (/ PTEN / RB / P53) in PCCs will reach a high level and act as a tumor suppressor within the first 1000 time units. Then, after at most 2000 time units, its concentration will eventually drop to a low level, and stay in this low level for at least another 10000 time units.
10	F[1000] (PCCPTEN ∧ F[2000] (G[10000] (! PCCPTEN)))	
11	F[1000] (PCCRB1 ∧ F[2000] (G[10000] (! PCCRB)))	
12	F[1000] (PCCP53 ∧ F[2000] (G[10000] (! PCCP53)))	
Cell functions of pancreatic cancer cells		
13	F[1000] ((! PCCAutophagy) ∧ F[2000](G[10000] PCCAutophagy))	In the development of pancreatic cancer, apoptosis firstly overwhelms autophagy, and then autophagy takes the leading place after a certain time point.
14	F[1000] (PCCApoptosis ∧ F[2000] (G[10000] (! PCCApoptosis)))	
15	F[1000] (G[10000] PCCProliferation)	Within 1000 time units, PCCs' proliferation will eventually be activated, and becomes a steady state for at least another 10000 time units.
16	!(! PCCP53 U[12000] PCCApoptosis)	It is not the case that, within 12000 steps, P53 in PCCs has to have a low concentration level until PCCs' Apoptosis being triggered.
Cell functions of pancreatic stellate cells		
17	F[1000] (G[10000] PSCActivation)	Within 1000 time units, PSCs' activation (/ migration) will eventually be activated, and becomes a steady state for at least another 10000 time units.
18	F[1000] (G[10000] PSCMigration)	
19	F[1000] (PSCApoptosis ∧ F[1000] (G[10000] (! PSCApoptosis)))	Within 1000 time units, PSCs' apoptosis will be triggered. Then, after at most 1000 time units, the initially functional apoptosis in PSCs will be inhibited and stay in inactive status for at least 10000 time units.
20	F[12000] PSCProliferation	Within 12000 steps, PSCs' proliferation will eventually be triggered.

!: logical not, ∧: logical and, F: eventually, G: always, U: until

Table 7.1: System properties used for the model selection using statistical model checking

Chapter 8

Conclusion and Future Work

In computer science, formal specification and analysis methods are used to design and prove properties of computer systems. If a desired property of a computer system turns out to fail, then we can in principle adapt and refine the system at hand. Orthogonal to this usage in computer science, for biology as an empirical science, where biological systems are a fact of life, formal methods serve to better understand the inner workings and emergent properties of such systems. This thesis developed new modeling language, formal analysis methods, and models for biological systems at different levels - the molecular, cellular, tissue, organ, and whole organism levels. It show how the study of biological and biomedical systems considering nonlinearity, nondeterminism, and stochasticity can benefit from formal modeling formalisms with different abstraction levels and model checking techniques.

The works presented in this thesis can be classified into three groups according to three research motivations. The first group is motivated by the study of pancreatic cancer, including the single cell analysis of pancreatic cancer cells, the study of the interplay between pancreatic cancer and stellate cells, and the development of a framework where formal methods and machine learning algorithms are used to automate the model construction and refinement for pancreatic cancer.

In Chapter 2, we have presented and formally checked an in silico model for a single cell of pancreatic cancer. The model incorporates important signaling pathways which are implicated with high frequency in pancreatic cancer. We have verified temporal logic properties encoding behavior related to cell fate, cell cycle, and oscillation of expression level in key proteins. As shown above, the model agrees well qualitatively with experiments. We have also suggested several properties which could be tested by future experiments.

Considering that, for these years, due to the poor treatment results for the pancreatic cancer, the research focus has been shifted from solely looking into pancreatic cancer cells towards investigating the pancreatic cancer microenvironment. So, it is of great importance and interest to understand the microenvironment. In Chapter 6, we have presented a multicellular and multiscale model of the PC microenvironment. The model is formally described using the extended BioNetGen language, which can capture the dynamics of multiscale biological systems using a combination of continuous and discrete rules. We have carried out stochastic simulation and StatMC to analyze

system behaviors under distinct conditions. Our verification results have confirmed the experimental findings with regard to the mutual promotion between PCCs and PSCs. We have also gained insights on how existing treatments latching onto different targets can lead to distinct outcomes. These results have demonstrated that our model might be used as a prognostic platform to identify new drug targets. We have then identified four potentially (poly)pharmacological strategies for depleting PSCs and inhibiting the PC development. We plan to test our predictions empirically. Another interesting direction is to extend the model by considering spatial information [82] and TAMs in the PC microenvironment.

From the above two works, we can tell that constructing and analyzing biological models can help to explain systems that we are studying, discover new questions, and even challenge existing findings. However, the creation of models often relies on intense human effort. This results in a slow development of models, let alone extending them with thousands of other possible component interactions that reported in newly published literature. Considering this situation, there is an urgent need for the automation of information extraction from literature, smart integration into models, and efficient and correct analysis of models, so as to allow researchers to re-use previously published work, in a comprehensive and timely manner. In Chapter 7, we have proposed a framework that utilizes published work to collect extensions for existing models, and then analyzes these extensions using stochastic simulation and statistical model checking. With biological properties being formulated as temporal logic, model checker can use the trace generated by the simulator to estimate the probability that a certain property holds. This gives us an efficient approach (speed-up from decades to hours) to re-use previously published results and observations for the purpose of conducting hundreds of *in silico* experiments with different setups (models). Our methods and the framework that we have developed comprise a promising new approach to comprehensively utilize published work. Moreover, this framework can also be used to search for pathways or interactions that are vital to certain functions, and to suggest targets for drug development. For example, using the ND models and statistical model checker, we can study closely how each layer of elements influences the elements we are interested in. Then, we can pin-point the models that satisfy several properties that we desire, and we should be able to identify a few candidates that play important roles in the regulation. Or, by using NI method, we can further observe whether there is actually an upstream network that controls the behavior of the elements. This gives us a deeper understanding of the network and helps us in further model development.

The second group is looking into the bounded reachability problems for hybrid systems and stochastic hybrid systems that are widely used to model biological systems. In detail, in Chapter 4, we have studied a novel method of killing bacteria using bacteriophage instead of antibiotics. A bacteriophage can be engineered to include code for proteins, which when inside bacteria can get activated and result in bacteria killing. Specifically, in this work we studied photosensitizing proteins, those that produce reactive oxygen species (ROS) when exposed to light. Excess amounts of ROS result in cell death. We created a hybrid model expressing both continuous and discrete dynamics. We defined this model within each of the stages that bacteria can go through, and used our tool (implemented the δ-decisions technique) for hybrid system reachability analysis to define parameters of the model that are otherwise hard or not possible to be found in experiments. We were especially interested in the timing effects, when the cells should be exposed to light, how long

the light exposure should be, and how long it takes photosensitized proteins to kill bacteria cells after exposure to light. Our analysis shows that the timing will be critical if this treatment, using bacteriophage and photosensitized proteins, is used for killing bacteria: the delay in exposure to light can significantly delay bacteria killing and could potentially lead to complications such as sepsis; and the duration of exposure to light is critical - turning light off too early may also not result in killing. Interestingly, we found that a broader range of SOX could kill bacteria, although the time to reach this effect may again be too long for practical purposes. We noticed that very low levels of SOX are efficient in bacteria killing, while medium levels result in the longest time to killing, and we are further investigating these results, as they point to potential improvements in our model. The results that we obtained will offer hints for the design of wet lab experiments.

Randomness happens naturally in real-world systems. So, To be able to analyze biological systems with uncertainties, in Chapter 5, we have developed and implemented the SReach algorithm, which solves probabilistic bounded reachability problems for two classes of models of stochastic hybrid systems. The first one is (nonlinear) hybrid automata with parametric uncertainty. The second one is probabilistic hybrid automata with additional randomness for both transition probabilities and variable resets. Standard approaches to reachability problems for linear hybrid systems require numerical solutions for large optimization problems, and become infeasible for systems involving both nonlinear dynamics over the reals and stochasticity. SReach encodes stochastic information by using a set of introduced random variables, and combines δ-complete decision procedures and statistical tests to solve δ-reachability problems in a sound manner. Compared to standard simulation-based methods, it supports non-deterministic branching, increases the coverage of simulation, and avoids the zero-crossing problem. To demonstrate SReach's applicability, it has been used to analyze three representative examples - a prostate cancer treatment model, a cardiac model, and a model of the tap withdrawal circuit in C. elegans - and other benchmarks, which are currently out of the reach of other formal tools.

In the third thread, as discussed in Chapter 3, we have developed and implemented an efficient LTL bounded model checking algorithm for Qualitative Networks, which extend Boolean networks by using discrete variables and algebraic functions as updating functions. Our technique utilizes the unique structure of Qualitative Networks to construct "decreasing reachability sets". These sets form part of a compact representation of paths in the QN and lead to significant acceleration in an implementation of bounded model checking. We find the experimental results very encouraging especially given the iterative development methodology biologists have been using when employing our tool BMA. As mentioned, our users "try out" several options and refine them according to results of simulation and verification. In this iterative process it is most important to be able to give fast answers to queries of the user. Considering the speed-ups afforded by this new technique, we have shown that model checking can be incorporated into the workflow of using our tools.

The work we have presented in this thesis suggests several ideas for future works.

Stochastic Hybrid Systems with Stochastic Differential Equations (SDEs). In Chapter 5, we have proposed and implemented a probabilistic reachability analysis method for two classes of stochastic hybrid systems, where randomness is introduced by system parameters and discrete

transitions. However, in real-world biological systems, many processes are stochastic intrinsically, such as species' population changes in ecosystems. Their dynamics are usually modeled by SDEs. Thus, one future topic is to design a model checking technique for general stochastic hybrid systems (GSHSs) where, besides probabilistic transitions, stochastic differential equations are used to capture continuous dynamics. One approach may be to introduce a new quantifier symbol for random variables and SDE constraints for stochastic processes, where a new SDE solver, which make use of numerical solutions to SDEs and simulation-based methods estimating distributions of hitting times for stochastic processes, can be integrated into existing nonlinear SMT solvers.

Considering Multiple Types of Existing Knowledge for the Automated Model Construction Framework. Following our work in Chapter 7, several works can done to make the framework more general and robust, including designing the way to consider the contradictions, implementing the detection of causal relations causing the failure of satisfying a certain system property, and adding more information into the extended model, for example, the probability for a relation to exist. Besides these, one future topic is to integrate more types of existing knowledge, such as time series datasets, small models published by others, and images describing related models. To achieve this, appropriate machine learning algorithms will be needed to learn model (fragments) from distinct types of biological knowledge. Figure 8.1 offers a schematic view of a more general framework where formal methods and machine learning can take joint efforts to automate the model design for biological models.

Formal Analysis Framework for Models using Various Modeling Languages. In reality, executable models are constructed with different levels of details. Which abstraction level to choose is mainly decided considering how much we know about a certain system. For example, since a large amount of work has been carried out to study the Ras signaling pathway in pancreatic cancer cells due to its importance in the cancer development, rule-based models are usually used to capture the reactions among involved proteins in detail. While, for the signaling interactions between pancreatic stellate cells and tumor-associated macrophages, there has not yet been as much work so far as people recently realized their interactions may be critical during the cancer progress and one reason for the poor prognosis of pancreatic cancer. Thus, logical models are used to only describe the structure of the interacting network. Currently, different analysis frameworks are adopted for distinct types of models that are used to describe different parts of a biological system. While, to throughly study a biological system, it is important to be able to analyze the system as a whole. So, we really need a formal analysis framework which allows to put models using various modeling languages together and offer a general way to analyze them.

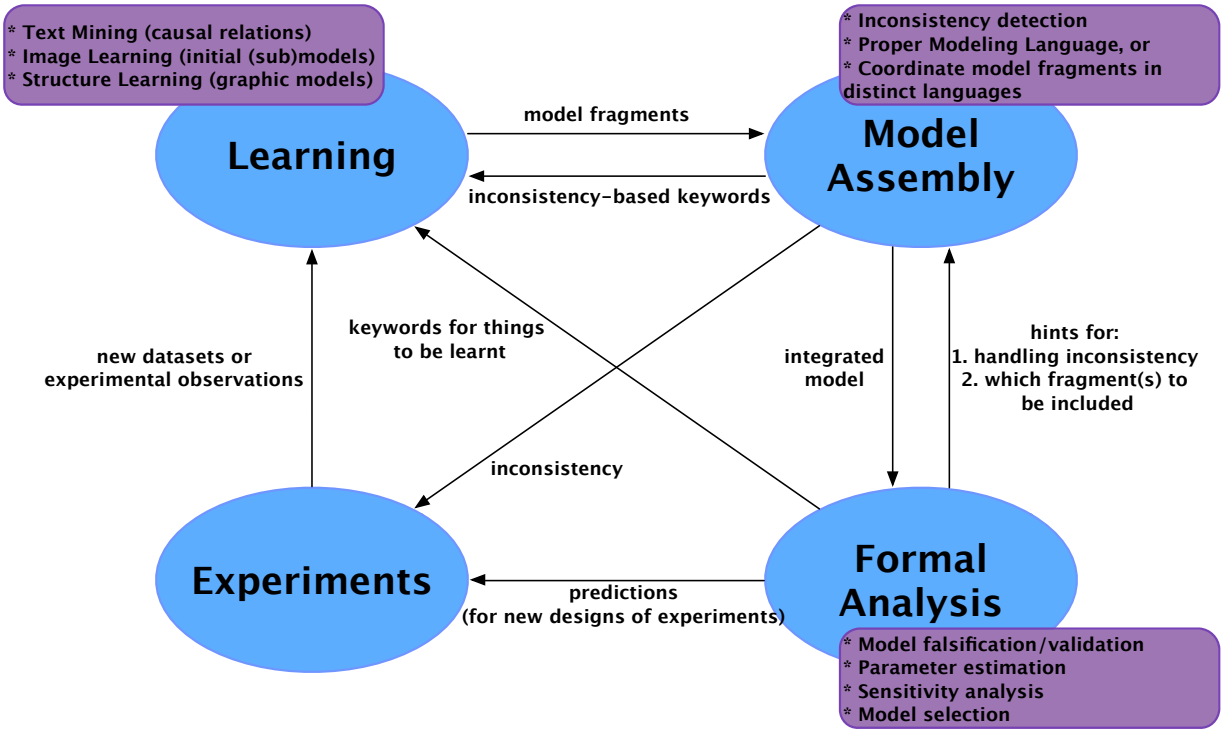


Figure 8.1: Schematic view of how formal methods and machine learning can take joint efforts to automate the model design for biological models.

Bibliography

[1] https://github.com/Yu-Hsin/simulator_java. 7.3, 7.4

[2] https://www.dropbox.com/s/mmks9xs0w4rjkcx/16K-fries_160331.tgz?dl=0. 7.4

[3] Personal communication with Jeffrey Melson Clarke, MD (Medical Instructor in the Department of Medicine, Duke University). 6.4

[4] Antibiotic resistance threats in the United States. The Center for Disease Control, 2013. 4.1

[5] Cancer facts and figures 2014, American Cancer Society, 2014. 2, 6

[6] World cancer report 2014, World Health Organization, 2014. 6

[7] Alessandro Abate. *Probabilistic reachability for stochastic hybrid systems: theory, computations, and applications*. ProQuest, 2007. 1.2

[8] Alessandro Abate, Joost-Pieter Katoen, John Lygeros, and Maria Prandini. A two-step scheme for approximate model checking of stochastic hybrid systems. In *Proceedings of the 18th IFAC World Congress. IFAC*, 2011. 1.2

[9] Alessandro Abate, Joost-Pieter Katoen, and Alexandru Mereacre. Quantitative automata model checking of autonomous stochastic hybrid systems. In *Proceedings of the 14th International Conference on Hybrid Systems: Computation and Control*, pages 83–92. ACM, 2011. 1.2

[10] Tatsuya Akutsu, Satoru Miyano, Satoru Kuhara, et al. Identification of genetic networks from a small number of gene expression patterns under the boolean network model. In *Pacific Symposium on Biocomputing*, volume 4, pages 17–28. Citeseer, 1999. 1.1

[11] Tatsuya Akutsu, Satoru Miyano, and Satoru Kuhara. Inferring qualitative relations in genetic networks and metabolic pathways. *Bioinformatics*, 16(8):727–734, 2000. 1.1

[12] Réka Albert and Hans G Othmer. The topology of the regulatory interactions predicts the expression pattern of the segment polarity genes in drosophila melanogaster. *Journal of Theoretical Biology*, 223(1):1–18, 2003. 1.1

[13] Uri Alon. *An introduction to systems biology: design principles of biological circuits*. CRC press, 2006. 6.4

[14] Rajeev Alur, Costas Courcoubetis, Thomas A Henzinger, and Pei-Hsin Ho. Hybrid automata: an algorithmic approach to the specification and verification of hybrid systems. In *Hybrid Systems*, pages 209–229. Springer, 1993. 1.1

[15] Saurabh Amin, Alessandro Abate, Maria Prandini, John Lygeros, and Shankar Sastry. Reachability analysis for controlled discrete time stochastic hybrid systems. In *Proceedings of the 9th International Conference on Hybrid Systems: Computation and Control*, pages 49–63. Springer, 2006. 1.2

[16] MV Apte, S Park, PA Phillips, N Santucci, D Goldstein, RK Kumar, GA Ramm, M Buchler, H Friess, JA McCarroll, et al. Desmoplastic reaction in pancreatic cancer: role of pancreatic stellate cells. *Pancreas*, 29(3):179–187, 2004. 6, 6.1, 6.4, 6.4

[17] Evan L Ardiel and Catharine H Rankin. An elegant mind: learning and memory in Caenorhabditis elegans. *Learning & Memory*, 17(4):191–201, 2010. 5.4.3

[18] Ludwig Arnold. *Stochastic Differential Equations: Theory and Applications*. Wiley - Interscience, 1974. 1.1

[19] Nichole Boyer Arnold and Murray Korc. Smad7 abrogates transforming growth factor-β1-mediated growth inhibition in COLO-357 cells through functional inactivation of the retinoblastoma protein. *Journal of Biological Chemistry*, 280(23):21858–21866, 2005. 2.1

[20] Max G Bachem, Marion Schünemann, Marco Ramadani, Marco Siech, Hans Beger, Andreas Buck, Shaoxia Zhou, Alexandra Schmid-Kotsas, and Guido Adler. Pancreatic carcinoma cells induce fibrosis by stimulating proliferation and matrix synthesis of stellate cells. *Gastroenterology*, 128(4):907–921, 2005. 6

[21] Valentina Baldazzi, Pedro T Monteiro, Michel Page, Delphine Ropers, Johannes Geiselmann, and Hidde de Jong. Qualitative analysis of genetic regulatory networks in bacteria. In *Understanding the Dynamics of Biological Systems*, pages 111–130. Springer, 2011. 1.1

[22] Thomas Ball and Sriram K Rajamani. The SLAM toolkit. In *Computer-Aided Verification*, pages 260–264. Springer, 2001. 1.2

[23] Nabeel Bardeesy and Ronald A DePinho. Pancreatic cancer biology and genetics. *Nature Reviews Cancer*, 2(12):897–909, 2002. 2.1, 2.4, 6.1

[24] Cornelia I Bargmann and Leon Avery. Laser killing of cells in Caenorhabditis elegans. *Methods in Cell Biology*, 48:225–250, 1995. 5.4.3

[25] Ezio Bartocci and Pietro Lió. Computational modeling, formal analysis, and tools for systems biology. *PLoS Computational Biology*, 12(1):e1004591, 2016. 1

[26] Dipak Barua, James R Faeder, and Jason M Haugh. Structure-based kinetic models of modular signaling protein function: focus on shp2. *Biophysical Journal*, 92(7):2290–2300, 2007. 1.1

[27] Dipak Barua, James R Faeder, and Jason M Haugh. Computational models of tandem src homology 2 domain interactions and application to phosphoinositide 3-kinase. *Journal of Biological Chemistry*, 283(12):7338–7345, 2008. 1.1

[28] MVL Bennett. A comparison of electrically and chemically mediated transmission. In *Structure and Function of Synapses*, pages 221–256. Raven Press New York, 1972. 5.4.3

[29] David Benque, Sam Bourton, Caitlin Cockerton, Byron Cook, Jasmin Fisher, Samin Ishtiaq, Nir Piterman, Alex Taylor, and Moshe Y Vardi. BMA: Visual tool for modeling and analyzing biological networks. In *Computer-Aided Verification*, pages 686–692. Springer, 2012. 3, 3.1, 3.4

[30] M Bensaid, N Tahiri-Jouti, C Cambillau, N Viguerie, B Colas, C Vidal, JP Tauber, JP Esteve, C Susini, and N Vaysse. Basic fibroblast growth factor induces proliferation of a rat pancreatic cancer cell line. inhibition by somatostatin. *International Journal of Cancer*, 50 (5):796–799, 1992. 6.1

[31] Antje Beyer, Peter Thomason, Xinzhong Li, James Scott, and Jasmin Fisher. Mechanistic insights into metabolic disturbance during type-2 diabetes and obesity using qualitative networks. In *Transactions on Computational Systems Biology XII*, pages 146–162. Springer, 2010. 1.1, 3, 3.4

[32] Nikhil Bhatla. Wormweb. `http://wormweb.org/`. (document), 5.5

[33] Armin Biere, Alessandro Cimatti, Edmund Clarke, and Yunshan Zhu. *Symbolic model checking without BDDs*. Springer, 1999. 1.2

[34] Arne T Bittig, Fiete Haack, Carsten Maus, and Adelinde M Uhrmacher. Adapting rule-based model descriptions for simulating in continuous and hybrid space. In *Proceedings of the 9th International Conference on Computational Methods in Systems Biology*, pages 161–170. ACM, 2011. 1.1

[35] Michael L Blinov, James R Faeder, Byron Goldstein, and William S Hlavacek. A network model of early events in epidermal growth factor receptor signaling that accounts for combinatorial complexity. *Biosystems*, 83(2):136–151, 2006. 1.1

[36] Roderick Bloem, Kavita Ravi, and Fabio Somenzi. Efficient decision procedures for model checking of linear time logic properties. In *Computer-Aided Verification*, pages 222–235. Springer, 1999. 1.2

[37] Henk AP Blom and Edwin A Bloem. Particle filtering for stochastic hybrid systems. In *43rd IEEE Conference on Decision and Control*, volume 3, pages 3221–3226. IEEE, 2004. 1.2

[38] Benjamin Bowen and Neal Woodbury. Single-molecule fluorescence lifetime and anisotropy measurements of the red fluorescent protein, DsRed, in solution. *Photochemistry and Photobiology*, 77(4):362–369, 2003. 4.1

[39] Robert K Brayton, Gary D Hachtel, Alberto Sangiovanni-Vincentelli, Fabio Somenzi, Adnan Aziz, Szu-Tsung Cheng, Stephen Edwards, Sunil Khatri, Yuji Kukimoto, Abelardo Pardo, et al. VIS: A system for verification and synthesis. In *Computer-Aided Verification*, pages 428–432. Springer, 1996. 1.2

[40] Sydney Brenner. The genetics of Caenorhabditis elegans. *Genetics*, 77(1):71–94, 1974. 5.4.3

[41] N. Bruchovsky, S. L. Goldenberg, P. S. Rennie, and Gleave M. E. Theoretical considerations and initial clinical results of intermittent hormone treatment of patients with advanced prostatic carcinoma. *Urologe A*, 34:389–392, 1995. 5.4.2

[42] N. Bruchovsky, L. Klotz, J. Crook, S. Malone, C. Ludgte, W. Morris, M. E. Gleave, S. L. Goldenberg, and P. S. Rennie. Final results of the canadian prospective phase II trial of intermittent androgen suppression for men in biochemical recurrence after radiotherapy for locally advanced prostate cancer. *Cancer*, 107:389–395, 2006. 5.4.2

[43] N. Bruchovsky, L. Klotz, J. Crook, S. Malone, C. Ludgte, W. Morris, M. E. Gleave, S. L. Goldenberg, and P. S. Rennie. Locally advanced prostate cancer biochemical results from a prospective phase II study of intermittent androgen suppression for men with evidence of prostate-specific antigen recurrence after radiotherapy. *Cancer*, 109:858–867, 2007. 5.4.2

[44] Nicholas Bruchovsky, Laurence Klotz, et al. Final results of the Canadian prospective phase II trial of intermittent androgen suppression for men in biochemical recurrence after radiotherapy for locally advanced prostate cancer. *Cancer*, 107(2):389–395, 2006. 5.4.2

[45] Nicholas Bruchovsky, Laurence Klotz, Juanita Crook, and Larry Goldenberg. Locally advanced prostate cancer: biochemical results from a prospective phase II study of intermittent androgen suppression for men with evidence of prostate-specific antigen recurrence after radiotherapy. *Cancer*, 109(5):858–867, 2007. 5.4.2

[46] Randal E Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Transactions on Computers*, 100(8):677–691, 1986. 1.2, 2.3

[47] Nicholas C. Buchan and S. Larry Goldenberg. Intermittent androgen suppression for prostate cancer. *Nature Reviews Urology*, 7:552–560, 2010. 5.4.2

[48] Peter Büchler, Amiq Gazdhar, Mario Schubert, Nathalia Giese, Howard A Reber, Oscar J Hines, Thomas Giese, Güralp O Ceyhan, Michael Müller, Markus W Büchler, et al. The Notch signaling pathway is related to neurovascular progression of pancreatic cancer. *Annals of Surgery*, 242(6):791, 2005. 2.1

[49] Alfonso Bueno-Orovio, Elizabeth M Cherry, and Flavio H Fenton. Minimal model for human ventricular action potentials in tissue. *Journal of Theoretical Biology*, 253(3):544–560, 2008. 1.1, 5.4.1

[50] Manuela L Bujorianu and John Lygeros. Toward a general theory of stochastic hybrid systems. In *Stochastic Hybrid Systems*, pages 3–30. Springer, 2006. 1.1

[51] Jerry Burch, Edmund M Clarke, and David Long. Symbolic model checking with partitioned transition relations. *Computer Science Department*, page 435, 1991. 1.2

[52] Jerry R Burch, Edmund M Clarke, Kenneth L McMillan, and David L Dill. Sequential circuit verification using symbolic model checking. In *27th ACM/IEEE Design Automation Conference*, pages 46–51. IEEE, 1990. 1.2

[53] Jerry R Burch, Edmund M Clarke, Kenneth L McMillan, David L Dill, and Lain-Jinn Hwang. Symbolic model checking: 10^{20} states and beyond. In *5th Annual IEEE Symposium on Logic in Computer Science*, pages 428–439. IEEE, 1990. 1.2

[54] Jerry R Burch, Edmund M Clarke, David E Long, Kenneth L McMillan, and David L Dill. Symbolic model checking for sequential circuit verification. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 13(4):401–424, 1994. 2.3

[55] Claudine Chaouiya. Petri net modeling of biological networks. *Briefings in Bioinformatics*, 8(4):210–219, 2007. 1.1

[56] Katherine C Chen, Laurence Calzone, Attila Csikasz-Nagy, Frederick R Cross, Bela Novak, and John J Tyson. Integrative analysis of cell cycle control in budding yeast. *Molecular Biology of the Cell*, 15(8):3841–3862, 2004. 1.1

[57] Li Chen, Ge Qi-Wei, Mitsuru Nakata, Hiroshi Matsuno, and Satoru Miyano. Modeling and simulation of signal transductions in an apoptosis pathway by using timed Petri nets. *Journal of Biosciences*, 32(1):113–127, 2007. 1.1

[58] Daniel C Chung, Suzanne B Brown, Fiona Graeme-Cook, Masao Seto, Andrew L Warshaw, Robert T Jensen, and Andrew Arnold. Overexpression of Cyclin D1 occurs frequently in human pancreatic endocrine tumors 1. *The Journal of Clinical Endocrinology & Metabolism*, 85(11):4373–4378, 2000. 2.4, 2.4

[59] Alessandro Cimatti, Edmund Clarke, Fausto Giunchiglia, and Marco Roveri. NuSMV: a new symbolic model checker. *International Journal on Software Tools for Technology Transfer*, 2(4):410–425, 2000. 1.2

[60] Alessandro Cimatti, Edmund Clarke, Enrico Giunchiglia, Fausto Giunchiglia, Marco Pistore, Marco Roveri, Roberto Sebastiani, and Armando Tacchella. NuSMV 2: an open-source tool for symbolic model checking. In *Computer-Aided Verification*, pages 359–364. Springer, 2002. 2.4

[61] Koen Claessen, Jasmin Fisher, Samin Ishtiaq, Nir Piterman, and Qinsi Wang. Model-checking signal transduction networks through decreasing reachability sets. In *Computer-Aided Verification*, pages 85–100. Springer, 2013. (document), 1.1, 1.2

[62] Koen Claessen, Jasmin Fisher, Samin Ishtiaq, Nir Piterman, and Qinsi Wang. Model-checking signal transduction networks through decreasing reachability sets. In *Technical Report MSR-TR-2013-30*. Microsoft Research, 2013. 3.3, 3.4

[63] Edmund Clarke, Orna Grumberg, Somesh Jha, Yuan Lu, and Helmut Veith. Counterexample-guided abstraction refinement. In *Computer-Aided Verification*, pages 154–169. Springer, 2000. 1.2

[64] Edmund Clarke, Armin Biere, Richard Raimi, and Yunshan Zhu. Bounded model checking using satisfiability solving. *Formal Methods in System Design*, 19(1):7–34, 2001. 3.2

[65] Edmund M Clarke and E Allen Emerson. Design and synthesis of synchronization skeletons using branching time temporal logic. In *Workshop on Logic of Programs*, pages 52–71. Springer, 1981. 2.3

[66] Edmund M Clarke and E Allen Emerson. *Design and synthesis of synchronization skeletons using branching time temporal logic*. Springer, 1982. 1.2

[67] Edmund M Clarke, Orna Grumberg, and David E Long. Model checking and abstraction. *ACM transactions on Programming Languages and Systems*, 16(5):1512–1542, 1994. 1.2

[68] Earl A Coddington and Norman Levinson. *Theory of ordinary differential equations*. Tata McGraw-Hill Education, 1955. 1.1

[69] Byron Cook, Jasmin Fisher, Elzbieta Krepska, and Nir Piterman. Proving stabilization of biological systems. In *Verification, Model Checking, and Abstract Interpretation*, pages 134–149. Springer, 2011. 3.1, 3.3

[70] Lucas Cordeiro, Bernd Fischer, and Joao Marques-Silva. SMT-based bounded model checking for embedded ANSI-C software. *IEEE Transactions on Software Engineering*, 38(4): 957–974, 2012. 1.2, 5

[71] Vincent Danos, Jérôme Feret, Walter Fontana, Russell Harmer, and Jean Krivine. Rule-based modeling of cellular signaling. In *International Conference on Concurrency Theory*, pages 17–41. Springer, 2007. 1.1

[72] Hidde De Jong. Modeling and simulation of genetic regulatory systems: a literature review. *Journal of computational biology*, 9(1):67–103, 2002. 1.1

[73] Leonardo De Moura and Nikolaj Bjørner. Z3: An efficient SMT solver. In *Tools and Algorithms for the Construction and Analysis of Systems*, pages 337–340. Springer, 2008. 1.2

[74] Patrik D'haeseleer, Shoudan Liang, and Roland Somogyi. Genetic network inference: from co-expression clustering to reverse engineering. *Bioinformatics*, 16(8):707–726, 2000. 1.1

[75] Julian Downward. Targeting RAS signaling pathways in cancer therapy. *Nature Reviews Cancer*, 3(1):11–22, 2003. 2.1

[76] Oliver Dreesen and Ali H Brivanlou. Signaling pathways in cancer and embryonic stem cells. *Stem Cell Reviews*, 3(1):7–17, 2007. 2.1

[77] Siri Dunér, Jacob Lopatko Lindman, Daniel Ansari, Chinmay Gundewar, and Roland Andersson. Pancreatic cancer: the role of pancreatic stellate cells in tumor progression. *Pancreatology*, 10(6):673–681, 2011. 6.1, 6.4

[78] Rick Durrett. *Probability: Theory and Examples*. Cambridge University Press, 2010. 5.2

[79] Bruno Dutertre and Leonardo De Moura. The yices SMT solver. *Tool paper at http://yices.csl.sri.com/tool-paper. pdf*, 2:2, 2006. 1.2

[80] Volker Ellenrieder, Martin E Fernandez Zapico, and Raul Urrutia. TGFβ-mediated signaling and transcriptional regulation in pancreatic development and cancer. *Current Opinion in Gastroenterology*, 17(5):434, 2001. 2.1

[81] M Erkan, C Reiser-Erkan, CW Michalski, and J Kleeff. Tumor microenvironment and progression of pancreatic cancer. *Experimental Oncology*, 32(3):128–131, 2010. 6

[82] Mert Erkan, Simone Hausmann, Christoph W Michalski, Alexander A Fingerle, Martin Dobritz, Jörg Kleeff, and Helmut Friess. The role of stroma in pancreatic cancer: diagnostic and therapeutic implications. *Nature Reviews Gastroenterology and Hepatology*, 9(8):454–467, 2012. 8

[83] James R Faeder, Michael L Blinov, and William S Hlavacek. Rule-based modeling of biochemical systems with BioNetGen. *Systems Biology*, pages 113–167, 2009. 1.1, 6, 6.2

[84] Mattia Falconi, Peter O'Neill, Maria Elena Stroppolo, and Alessandro Desideri. Superoxide dismutase kinetics. *Methods in Enzymology*, 349:38–49, 2002. 4.1

[85] Buckminster Farrow, Daniel Albo, and David H Berger. The role of the tumor microenvironment in the progression of pancreatic cancer. *Journal of Surgical Research*, 149(2): 319–328, 2008. 6

[86] Christine Feig, Aarthi Gopinathan, Albrecht Neesse, Derek S Chan, Natalie Cook, and David A Tuveson. The pancreas cancer microenvironment. *Clinical Cancer Research*, 18(16):4266–4276, 2012. 6, 6.1, 6.4

[87] Flavio H Fenton and Elizabeth M Cherry. Models of cardiac cell. *Scholarpedia*, 3(8):1868, 2008. 5.4.3

[88] Jasmin Fisher and Thomas A Henzinger. Executable cell biology. *Nature Biotechnology*, 25(11):1239–1249, 2007. 1

[89] Jasmin Fisher, Nir Piterman, Alex Hajnal, and Thomas A Henzinger. Predictive modeling of signaling crosstalk during C. elegans vulval development. *PLoS Computational Biology*, 3(5):e92, 2007. (document), 3.1, 3.1

[90] Jasmin Fisher, Ali Sinan Köksal, Nir Piterman, and Steven Woodhouse. Synthesising executable gene regulatory networks from single-cell gene expression data. In *International Conference on Computer Aided Verification*, pages 544–560. Springer, 2015. 1.1

[91] Martin Fränzle, Holger Hermanns, and Tino Teige. Stochastic satisfiability modulo theory: A novel technique for the analysis of probabilistic hybrid systems. In *Hybrid Systems: Computation and Control*, pages 172–186. Springer, 2008. 1.2

[92] Martin Fränzle, Ernst Moritz Hahn, Holger Hermanns, Nicolás Wolovick, and Lijun Zhang. Measurability and safety verification for stochastic hybrid systems. In *Proceedings of the 14th international conference on Hybrid Systems: Computation and Control*, pages 43–52. ACM, 2011. 1.1, 1.2

[93] Malay K Ganai, Aarti Gupta, and Pranav Ashar. Efficient SAT-based unbounded symbolic model checking using circuit co-factoring. In *Proceedings of the 2004 IEEE/ACM International conference on Computer-Aided Design*, pages 510–517. IEEE, 2004. 1.2

[94] Sicun Gao, Soonho Kong, and Edmund M Clarke. dReal: an SMT solver for nonlinear theories over the reals. In *Automated Deduction–CADE-24*, pages 208–214. Springer, 2013. 4.2, 4.2, 5.2

[95] Sicun Gao, Soonho Kong, and Edmund M Clarke. Satisfiability modulo ODEs. In *FMCAD*, pages 105–112, Oct. 2013. 4, 4.2, 4.2

[96] Hartmann Genrich, Robert Küffner, and Klaus Voss. Executable Petri net models for the analysis of metabolic pathways. *International Journal on Software Tools for Technology Transfer*, 3(4):394–404, 2001. 1.1

[97] Naama Geva-Zatorsky, Nitzan Rosenfeld, Shalev Itzkovitz, Ron Milo, Alex Sigal, Erez Dekel, Talia Yarnitzky, Yuvalal Liron, Paz Polak, Galit Lahav, et al. Oscillations and variability in the p53 system. *Molecular Systems Biology*, 2(1), 2006. 2.4

[98] Ronojoy Ghosh and Claire Tomlin. Symbolic reachable set computation of piecewise affine hybrid automata and its application to biological modeling: Delta-notch protein signaling. *Systems Biology*, 1(1):170–183, 2004. 1.1

[99] Patrice Godefroid. Using partial orders to improve automatic verification methods. In *Computer-Aided Verification*, pages 176–185. Springer, 1991. 1.2

[100] Haijun Gong, Qinsi Wang, Paolo Zuliani, James R Faeder, Michael Lotze, and E Clarke. Symbolic model checking of signaling pathways in pancreatic cancer. In *BICoB*, page 245, 2011. (document), 1.1, 2

[101] Haijun Gong, Paolo Zuliani, Qinsi Wang, and Edmund M Clarke. Formal analysis for logical models of pancreatic cancer. In *Decision and Control and European Control Conference*, pages 4855–4860. IEEE, 2011. (document), 1.1, 2

[102] Miriam B Goodman, David H Hall, Leon Avery, and Shawn R Lockery. Active currents regulate sensitivity and dynamic range in C. elegans neurons. *Neuron*, 20(4):763–772, 1998. 5.4.3

[103] Peter JE Goss and Jean Peccoud. Quantitative modeling of stochastic systems in molecular biology by using stochastic Petri nets. *Proceedings of the National Academy of Sciences*, 95(12):6750–6755, 1998. 1.1

[104] Jesse M Gray, Joseph J Hill, and Cornelia I Bargmann. A circuit for navigation in Caenorhabditis elegans. *Proceedings of the National Academy of Sciences of the United States of America*, 102(9):3184–3191, 2005. 5.4.3

[105] Gerd Gruenert, Bashar Ibrahim, Thorsten Lenser, Maiko Lohel, Thomas Hinze, and Peter Dittrich. Rule-based spatial modeling with diffusing, geometrically constrained molecules. *BMC Bioinformatics*, 11(1):1, 2010. 1.1

[106] Simone Gupta, Siddharth S Bisht, Ritushree Kukreti, Sanjeev Jain, and Samir K Brahmachari. Boolean network analysis of a neurotransmitter signaling pathway. *Journal of Theoretical Biology*, 244(3):463–469, 2007. 1.1

[107] Paul S Haber, Gregory W Keogh, Minoti V Apte, Corey S Moran, Nancy L Stewart, Darrell HG Crawford, Romano C Pirola, Geoffrey W McCaughan, Grant A Ramm, and Jeremy S Wilson. Activation of pancreatic stellate cells in human and experimental pancreatic fibrosis. *The American Journal of Pathology*, 155(4):1087–1095, 1999. 6.1, 6.4

[108] Ernst Moritz Hahn, Gethin Norman, David Parker, Björn Wachter, and Lijun Zhang. Game-based abstraction and controller synthesis for probabilistic hybrid systems. In *8th International Conference on Quantitative Evaluation of Systems*, pages 69–78. IEEE, 2011. 1.2

[109] Gerhard Hamilton and Gerhard Theyer. *Advances in Prostate Cancer*, chapter 13, pages 305–330. INTECH, 2013. 5.4.2

[110] RH Hardin, RP Kurshan, SK Shukla, and MY Vardi. A new heuristic for bad cycle detection using BDDs. In *Computer-Aided Verification*, pages 268–278. Springer, 1997. 1.2

[111] Leonard A Harris, Justin S Hogg, and James R Faeder. Compartmental rule-based modeling of biochemical systems. In *Winter Simulation Conference*, pages 908–919. Winter Simulation Conference, 2009. 1.1

[112] Susan Haupt, Michael Berger, Zehavit Goldberg, and Ygal Haupt. Apoptosis - the p53 network. *Journal of Cell Science*, 116(20):4077–4085, 2003. 2.4

[113] Ygal Haupt, Ruth Maya, Anat Kazaz, and Moshe Oren. Mdm2 promotes the rapid degradation of p53. *Nature*, 387(6630):296–299, 1997. 2.1

[114] Klaus Havelund and Natarajan Shankar. Experiments in theorem proving and model checking for protocol verification. In *FME'96: Industrial Benefit and Advances in Formal Methods*, pages 662–681. Springer, 1996. 1.2

[115] Zarifeh Heidary, Jafar Ghaisari, Shiva Moein, Mahmood Naderi, and Yousof Gheisari. Stochastic Petri net modeling of hypoxia pathway predicts a novel incoherent feed-forward loop controlling sdf-1 expression in acute kidney injury. *IEEE Transactions on Nanobioscience*, 15(1):19–26, 2016. 1.1

[116] Ernst Heinmöller, Wolfgang Dietmaier, Hubert Zirngibl, Petra Heinmöller, William Scaringe, Karl-Walter Jauch, Ferdinand Hofstädter, and Josef Rüschoff. Molecular analysis of microdissected tumors and preneoplastic intraductal lesions in pancreatic carcinoma. *The American Journal of Pathology*, 157(1):83–92, 2000. 2.4

[117] Thomas A Henzinger. *The Theory of Hybrid Automata*. Springer, 2000. 5, 5.1

[118] Thomas A Henzinger, Ranjit Jhala, Rupak Majumdar, and Grégoire Sutre. Software verification with BLAST. In *Model Checking Software*, pages 235–239. Springer, 2003. 1.2

[119] Melanie M Hippert, Patrick S O'Toole, and Andrew Thorburn. Autophagy in cancer: good, bad, or both? *Cancer Research*, 66(19):9349–9351, 2006. 6.1, 6.4

[120] Alan L Hodgkin and Andrew F Huxley. A quantitative description of membrane current and its application to conduction and excitation in nerve. *The Journal of Physiology*, 117(4): 500, 1952. 5.4.3

[121] Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *J American Statistical Association*, 58(301):13–30, 1963. 5.2

[122] Alexander Hoffmann, Andre Levchenko, Martin L Scott, and David Baltimore. The IκB-NF-κB signaling module: temporal control and selective gene activation. *Science*, 298 (5596):1241–1245, 2002. 2.1, 2.4

[123] Gerard J Holzmann. The model checker SPIN. *IEEE Transactions on Software Engineering*, 23(5):279–295, 1997. 1.2

[124] Jianghai Hu, John Lygeros, and Shankar Sastry. Towards a theory of stochastic hybrid systems. In *Hybrid Systems: Computation and Control*, pages 160–173. Springer, 2000. 1.1

[125] Jianghai Hu, Wei-Chung Wu, and Shankar Sastry. Modeling subtilin production in bacillus subtilis using stochastic hybrid systems. In *International Workshop on Hybrid Systems: Computation and Control*, pages 417–431. Springer, 2004. 1.1

[126] Sui Huang and Donald E Ingber. Shape-dependent control of cell growth, differentiation, and apoptosis: switching between attractors in cell regulatory networks. *Experimental Cell Research*, 261(1):91–103, 2000. 1.1

[127] H Hurwitz, N Uppal, SA Wagner, JC Bendell, JT Beck, S Wade, JJ Nemunaitis, PJ Stella, JM Pipas, ZA Wainberg, et al. A randomized double-blind phase 2 study of ruxolitinib (RUX) or placebo (PBO) with capecitabine (CAPE) as second-line therapy in patients (pts) with metastatic pancreatic cancer (mPC). *Journal of Clinical Oncology*, 32:55, 2014. 6.4

[128] Bashar Ibrahim, Richard Henze, Gerd Gruenert, Matthew Egbert, Jan Huwald, and Peter Dittrich. Spatial rule-based modeling: a method and its application to the human mitotic kinetochore. *Cells*, 2(3):506–544, 2013. 1.1

[129] Md. Ariful Islam, Qinsi Wang, Edmund Clarke, Scott Smolka, Ramin Hasani, Radu Grosu, and Ondrej Balun. Probabilistic reachability analysis of the tap withdrawal circuit in Caenorhabditis elegans. In *18th IEEE International High-Level Design Validation and Test Workshop*. IEEE, 2016. (document)

[130] Robert Jaster. Molecular regulation of pancreatic stellate cell function. *Molecular Cancer*, 3(1):26, 2004. 6.1

[131] Amarsanaa Jazag, Hideaki Ijichi, Fumihiko Kanai, Takaaki Imamura, Bayasi Guleng, Miki Ohta, Jun Imamura, Yasuo Tanaka, Keisuke Tateishi, Tsuneo Ikenoue, et al. Smad4 silencing in pancreatic cancer cell lines using stable RNA interference and gene expression profiles induced by transforming growth factor-β. *Oncogene*, 24(4):662–671, 2005. 2.1

[132] Sumit K Jha, Edmund M Clarke, Christopher J Langmead, Axel Legay, André Platzer, and Paolo Zuliani. A bayesian approach to model checking biological systems. In *Computational Methods in Systems Biology*, pages 218–234. Springer, 2009. 6, 6.3

[133] Siân Jones, Xiaosong Zhang, D Williams Parsons, Jimmy Cheng-Ho Lin, Rebecca J Leary, Philipp Angenendt, Parminder Mankoo, Hannah Carter, Hirohiko Kamiyama, Antonio Jimeno, et al. Core signaling pathways in human pancreatic cancers revealed by global genomic analyses. *Science*, 321(5897):1801–1806, 2008. 2, 2.1, 2.1

[134] Eric R Kandel and James H Schwartz. Molecular biology of learning: modulation of transmitter release. *Science*, 218(4571):433–443, 1982. 5.4.3

[135] Eric R Kandel, James H Schwartz, Thomas M Jessell, Steven A Siegelbaum, and AJ Hudspeth. *Principles of Neural Science*, volume 4. McGraw-hill New York, 2000. 5.4.3

[136] Rui Kang, Daolin Tang, Nicole E Schapiro, Kristen M Livesey, Adam Farkas, Patricia Loughran, Angelika Bierhaus, Michael T Lotze, and Herbert J Zeh. The receptor for advanced glycation end products (RAGE) sustains autophagy and limits apoptosis, promoting pancreatic tumor cell survival. *Cell Death & Differentiation*, 17(4):666–676, 2010. 2.1

[137] Rui Kang, Daolin Tang, Kristen M Livesey, Nicole E Schapiro, Michael T Lotze, and Herbert J Zeh III. The receptor for advanced glycation end-products (RAGE) protects pancreatic tumor cells against oxidative injury. *Antioxidants & Redox Signaling*, 15(8):2175–2184, 2011. 2.1

[138] H Karadag and R Bilgin. Purification of copper-zinc superoxide dismutase from human erythrocytes and partial characterization. *Biotechnology & Biotechnological Equipment*, 24 (1):1653–1656, 2010. 4.1

[139] Robert E Kass and Adrian E Raftery. Bayes factors. *JASA*, 90(430):773–795, 1995. 5.2

[140] Saul Kato, Harris S Kaplan, Tina Schrödel, Susanne Skora, Theodore H Lindsay, Eviatar Yemini, Shawn Lockery, and Manuel Zimmer. Global brain dynamics embed the motor command sequence of Caenorhabditis elegans. *Cell*, 163(3):656–669, 2015. 5.4.3

[141] Joost-Pieter Katoen, Maneesh Khattri, and Ivan S Zapreev. A Markov reward model checker. In *Second International Conference on the Quantitative Evaluation of Systems*, pages 243–244. IEEE, 2005. 1.2

[142] Stuart Kauffman. Homeostasis and differentiation in random genetic control networks. *Nature*, 224:177–178, 1969. 1.1

[143] Taizo Kawano, Michelle D Po, Shangbang Gao, George Leung, William S Ryu, and Mei Zhen. An imbalancing act: gap junctions reduce the backward motor circuit activity to bias C. elegans for forward locomotion. *Neuron*, 72(4):572–586, 2011. 5.4.3, 5.4.3

[144] Jörg Kleeff, Philipp Beckhove, Irene Esposito, Stephan Herzig, Peter E Huber, J Matthias Löhr, and Helmut Friess. Pancreatic cancer microenvironment. *International Journal of Cancer*, 121(4):699–705, 2007. 6

[145] Christof Koch and Idan Segev. *Methods in Neuronal Modeling: from Ions to Networks*. MIT press, 1998. 5.4.3

[146] Yasuko Kondo, Takao Kanzawa, Raymond Sawaya, and Seiji Kondo. The role of autophagy in cancer development and response to therapy. *Nature Reviews Cancer*, 5(9):726–734, 2005. 6.1

[147] Soonho Kong, Sicun Gao, Wei Chen, and Edmund Clarke. dReach: δ-reachability analysis for hybrid systems. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 200–205. Springer, 2015. 5, 5.2, 5.4.3

[148] Saul Kripke. Semantical considerations of the modal logic. 2007. 2.3

[149] Daniel Kroening and Michael Tautschnig. CBMC – C bounded model checker. In *Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, volume 8413 of *LNCS*, pages 389–391. Springer, 2014. ISBN 978-3-642-54861-1. 1.2

[150] M. Kwiatkowska, G. Norman, and D. Parker. PRISM 4.0: Verification of probabilistic real-time systems. In *Computer-Aided Verification*, volume 6806, pages 585–591. Springer, 2011. 1.2, 7.4

[151] Tze Leung Lai. Nearly optimal sequential tests of composite hypotheses. *AOS*, 16(2):856–886, 1988. 5.2

[152] Fangting Li, Tao Long, Ying Lu, Qi Ouyang, and Chao Tang. The yeast cell-cycle network is robustly designed. *Proceedings of the National Academy of Sciences of the United States of America*, 101(14):4781–4786, 2004. 1.1

[153] Kai-Wen Liang, Qinsi Wang, Cheryl A. Telmer, Divyaa Ravichandran, Peter Spirtes, and Natasa Miskov-Zivanov. LEaRn: Framework for literature exploration and reasoning. 2016. (document), 7.3

[154] Bing Liu, Soonho Kong, Sicun Gao, Paolo Zuliani, and Edmund M Clarke. Towards personalized prostate cancer therapy using delta-reachability analysis. In *Proceedings of the 18th International Conference on Hybrid Systems: Computation and Control*, pages 227–232. ACM, 2015. 5.4.2

[155] Daniel Machado, Rafael S Costa, Miguel Rocha, Eugénio C Ferreira, Bruce Tidor, and Isabel Rocha. Modeling formalisms in systems biology. *AMB Express*, 1(1):1, 2011. 1

[156] Daruka Mahadevan and Daniel D Von Hoff. Tumor-stroma interactions in pancreatic ductal adenocarcinoma. *Molecular Cancer Therapeutics*, 6(4):1186–1197, 2007. 6.1

[157] Guillermo Mariño, Mireia Niso-Santano, Eric H Baehrecke, and Guido Kroemer. Self-consumption: the interplay of autophagy and apoptosis. *Nature Reviews: Molecular Cell Biology*, 15(2):81–94, 2014. 6.1, 6.4

[158] Will Marrero, Edmund Clarke, and Somesh Jha. Model checking for security protocols. Technical report, DTIC Document, 1997. 1.2

[159] Atsushi Masamune, Masahiro Satoh, Kazuhiro Kikuta, Noriaki Suzuki, Kennichi Satoh, and Tooru Shimosegawa. Ellagic acid blocks activation of pancreatic stellate cells. *Biochemical Pharmacology*, 70(6):869–878, 2005. 6.1

[160] Carsten Maus, Stefan Rybacki, and Adelinde M Uhrmacher. Rule-based multi-level modeling of cell biological systems. *BMC Systems Biology*, 5(1):166, 2011. 1.1

[161] Ken L McMillan. Applying SAT methods in unbounded symbolic model checking. In *Computer-Aided Verification*, pages 250–264. Springer, 2002. 1.2

[162] Kenneth L McMillan. *Symbolic model checking*. Springer, 1993. 1.2, 2.3

[163] Kenneth L McMillan. Interpolation and SAT-based model checking. In *Computer-Aided Verification*, pages 1–13. Springer, 2003. 1.2

[164] Natasa Miskov-Zivanov, Michael S Turner, Lawrence P Kane, Penelope A Morel, and James R Faeder. Duration of T cell stimulation as a critical determinant of cell fate and plasticity. *Science Signaling*, 6(300):ra97, 2013. 7.3

[165] Natasa Miskov-Zivanov, Peter Wei, and Chang Sheng Clement Loh. THiMED: Time in hierarchical model extraction and design. In *International Conference on Computational Methods in Systems Biology*, pages 260–263. Springer, 2014. 1.1, 7.3

[166] Diego Muilenburg, Colin Parsons, Jodi Coates, Subbulakshmi Virudachalam, and Richard J Bold. Role of autophagy in apoptotic regulation by AKT in pancreatic cancer. *Anticancer Research*, 34(2):631–637, 2014. 6.1

[167] LO Murphy, MW Cluck, S Lovas, F Ötvös, RF Murphy, AV Schally, J Permert, J Larsson, JA Knezetic, and TE Adrian. Pancreatic cancer cells require an EGF receptor-mediated autocrine pathway for proliferation in serum-free conditions. *British Journal of Cancer*, 84 (7):926, 2001. 6.1

[168] Aurélien Naldi, Denis Thieffry, and Claudine Chaouiya. Decision diagrams for the representation and analysis of logical models of genetic networks. In *International Conference on Computational Methods in Systems Biology*, pages 233–247. Springer, 2007. 1.1, 3

[169] DE Nelson, AEC Ihekwaba, M Elliott, JR Johnson, CA Gibney, BE Foreman, G Nelson, V See, CA Horton, DG Spiller, et al. Oscillations in NFκB signaling control the dynamics of gene expression. *Science*, 306(5696):704–708, 2004. 2.4

[170] Ernst Niebur and Paul Erdös. Theory of the locomotion of nematodes: dynamics of undulatory progression on a surface. *Biophysical Journal*, 60(5):1132, 1991. 5.4.3

[171] Doron Peled. All from one, one for all: on model checking using representatives. In *Computer-Aided Verification*, pages 409–423. Springer, 1993. 1.2

[172] CA Petri. *Kommunikation mit Automaten*. PhD thesis, Rheinisch-Westflisches Institut f. instrumentelle Mathematik an d. Univ, 1962. Ph.D. thesis. 1.1

[173] PA Phillips, MJ Wu, RK Kumar, E Doherty, JA McCarroll, S Park, Ron C Pirola, JS Wilson, and MV Apte. Cell migration: a novel aspect of pancreatic stellate cell biology. *Gut*, 52(5): 677–682, 2003. 6.1

[174] Jonathan T Pierce-Shimomura, Thomas M Morse, and Shawn R Lockery. The fundamental role of pirouettes in Caenorhabditis elegans chemotaxis. *The Journal of Neuroscience*, 19 (21):9557–9569, 1999. 5.4.3

[175] André Platzer. Stochastic differential dynamic logic for stochastic hybrid programs. In *Automated Deduction–CADE-23*, pages 446–460. Springer, 2011. 1.1, 1.2

[176] Sergei Pletnev, Nadya G Gurskaya, Nadya V Pletneva, Konstantin A Lukyanov, Dmitri M Chudakov, Vladimir I Martynov, et al. Structural basis for phototoxicity of the genetically encoded photosensitizer KillerRed. *Journal of Biological Chemistry*, 284(46):32028–32039, 2009. 4, 4.1

[177] Steve Plimpton. Fast parallel algorithms for short-range molecular dynamics. *Journal of Computational Physics*, 117(1):1–19, 1995. 1.1

[178] Amir Pnueli. The temporal logic of programs. In *18th Annual Symposium on Foundations of Computer Science*, pages 46–57. IEEE, 1977. 1.2, 3, 6.3

[179] Amir Pnueli and Aleksandr Zaks. On the merits of temporal testers. In *25 Years of Model Checking*, pages 172–195. Springer, 2008. 3, 3.2

[180] Jean-Pierre Queille and Joseph Sifakis. Specification and verification of concurrent systems in CESAR. In *International Symposium on Programming*, pages 337–351. Springer, 1982. 1.2

[181] Wilfrid Rall. Cable theory for dendritic neurons. In *Methods in Neuronal Modeling*, pages 9–92. MIT press, 1989. 5.4.3

[182] Derek Riley, Xenofon Koutsoukos, and Kasandra Riley. Modeling and simulation of biochemical processes using stochastic hybrid systems: The sugar cataract development process. In *Hybrid Systems: Computation and Control*, pages 429–442. Springer, 2008. 1.2

[183] Joe Rodriguez and Yuri Lazebnik. Caspase-9 and APAF-1 form an active holoenzyme. *Genes & Development*, 13(24):3179–3184, 1999. 2.1

[184] Jacqueline K Rose and Catharine H Rankin. Analyses of habituation in Caenorhabditis elegans. *Learning & Memory*, 8(2):63–69, 2001. 5.4.3

[185] Ester Rozenblum, Mieke Schutte, Michael Goggins, Stephan A Hahn, Shawn Panzer, Marianna Zahurak, Steven N Goodman, Taylor A Sohn, Ralph H Hruban, Charles J Yeo, et al. Tumor-suppressive pathways in pancreatic carcinoma. *Cancer Research*, 57(9):1731–1734, 1997. 2.4

[186] Karen Sachs, Omar Perez, Dana Pe'er, Douglas A Lauffenburger, and Garry P Nolan. Causal protein-signaling networks derived from multiparameter single-cell data. *Science*, 308(5721):523–529, 2005. 1.1

[187] Nicole Samm, Kristin Werner, Felix Rückert, Hans Detlev Saeger, Robert Grützmann, and Christian Pilarsky. The role of apoptosis in the pathology of pancreatic cancer. *Cancers*, 3 (1):1–16, 2010. 2.1

[188] Lucas Sanchez and Denis Thieffry. Segmenting the fly embryo: a logical analysis of the pair-rule cross-regulatory module. *Journal of Theoretical Biology*, 224(4):517–537, 2003. 3, 3.4

[189] Atsuo T Sasaki, Arkaitz Carracedo, Jason W Locasale, Dimitrios Anastasiou, Koh Takeuchi, Emily Rose Kahoud, Sasson Haviv, John M Asara, Pier Paolo Pandolfi, and Lewis C Cantley. Ubiquitination of Ras enhances activation and facilitates binding to select downstream effectors. *Science Signaling*, 4(163):ra13, 2011. 7.1

[190] Marc A Schaub, Thomas A Henzinger, and Jasmin Fisher. Qualitative networks: a symbolic approach to analyze biological signaling networks. *BMC Systems Biology*, 1(1):1, 2007. 1.1, 3, 3.1, 3.1, 3.4

[191] Roberto Sebastiani, Stefano Tonetta, and Moshe Y Vardi. Symbolic systems, explicit properties: on hybrid approaches for LTL symbolic model checking. In *Computer-Aided Verification*, pages 350–363. Springer, 2005. 1.2

[192] Mary Sheeran, Satnam Singh, and Gunnar Stålmarck. Checking safety properties using induction and a SAT-solver. In *Formal Methods in Computer-Aided Design*, pages 127–144. Springer, 2000. 1.2

[193] Ilya Shmulevich, Edward R Dougherty, Seungchan Kim, and Wei Zhang. Probabilistic boolean networks: a rule-based uncertainty model for gene regulatory networks. *Bioinformatics*, 18(2):261–274, 2002. 1.1, 3

[194] Peter M Siegel and Joan Massagué. Cytostatic and apoptotic actions of TGFβ in homeostasis and cancer. *Nature Reviews Cancer*, 3(11):807–820, 2003. 6.1

[195] Rebecca Siegel, Deepa Naishadham, and Ahmedin Jemal. Cancer statistics, 2013. *CA: A Cancer Journal for Clinicians*, 63:11–30, 2013. 5.4.2

[196] E Simao, Elisabeth Remy, Denis Thieffry, and Claudine Chaouiya. Qualitative modeling of regulated metabolic pathways: application to the tryptophan biosynthesis in E. coli. *Bioinformatics*, 21(suppl 2):ii190–ii196, 2005. 1.1

[197] Michael W Sneddon, James R Faeder, and Thierry Emonet. Efficient modeling, simulation and coarse-graining of biological complexity with NFsim. *Nature Methods*, 8(2):177–183, 2011. 6, 6.3

[198] Loling Song, EJ Hennink, I Ted Young, and Hans J Tanke. Photobleaching kinetics of fluorescein in quantitative fluorescence microscopy. *Biophysical Journal*, 68(6):2588, 1995. (document), 4.2

[199] Young Hwa Soung, Jong Woo Lee, Su Young Kim, Won Sang Park, Suk Woo Nam, Jung Young Lee, Nam Jin Yoo, and Sug Hyung Lee. Somatic mutations of CASP3 gene in human cancers. *Human Genetics*, 115(2):112–115, 2004. 2.1

[200] Jeremy Sproston. Decidable model checking of probabilistic hybrid automata. In *Formal Techniques in Real-Time and Fault-Tolerant Systems*, pages 31–45. Springer, 2000. 1.1, 1.2, 5

[201] Jeremy Sproston. Model checking for probabilistic timed and hybrid systems. In *PhD thesis*. School of Computer Science, University of Birmingham, 2001. 1.1, 1.2

[202] Kiwamu Takemoto, Tomoki Matsuda, Naoki Sakai, Donald Fu, Masanori Noda, Susumu Uchiyama, Ippei Kotera, Yoshiyuki Arai, Masataka Horiuchi, Kiichi Fukui, et al. SuperNova, a monomeric photosensitizing fluorescent protein for chromophore-assisted light inactivation. *Scientific Reports*, 3, 2013. 4.1

[203] Sarah P Thayer, Marina Pasca di Magliano, Patrick W Heiser, Corinne M Nielsen, Drucilla J Roberts, Gregory Y Lauwers, Yan Ping Qi, Stephan Gysin, Carlos Fernández-del Castillo, Vijay Yajnik, et al. Hedgehog is an early and late mediator of pancreatic cancer tumorigenesis. *Nature*, 425(6960):851–856, 2003. 2.1

[204] René Thomas, Denis Thieffry, and Marcelle Kaufman. Dynamical behaviour of biological regulatory networks? I. biological role of feedback loops and practical use of the concept of the loop-characteristic state. *Bulletin of Mathematical Biology*, 57(2):247–276, 1995. 3

[205] Cesare Tinelli. SMT-based model checking. In *NASA Formal Methods*, page 1, 2012. 1.2, 5

[206] Roger Y Tsien and Alan Waggoner. Fluorophores for confocal microscopy: photophysics and photochemistry. In *Handbook of Biological Confocal Microscopy, 3rd Edition*. Springer, 2006. 4.1

[207] Alexa B Turke, Youngchul Song, Carlotta Costa, Rebecca Cook, Carlos L Arteaga, John M Asara, and Jeffrey A Engelman. MEK inhibition leads to PI3K/AKT activation by relieving a negative feedback on ERBB receptors. *Cancer Research*, 72(13):3228–3237, 2012. 7.1

[208] Marco A. Valenzuela-Escárcega, Gustave Hahn-Powell, Thomas Hicks, and Mihai Surdeanu. A domain-independent rule-based framework for event extraction. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing: Software Demonstrations*, pages 127–132. ACL-IJCNLP 2015, 2015. 7, 7.1

[209] Antti Valmari. Stubborn sets for reduced state space generation. In *Advances in Petri Nets 1990*, pages 491–515. Springer, 1991. 1.2

[210] Judy R van Beijnum, Wim A Buurman, and Arjan W Griffioen. Convergence and amplification of toll-like receptor (TLR) and receptor for advanced glycation end products (RAGE) signaling pathways via high mobility group B1 (HMGB1). *Angiogenesis*, 11(1):91–99, 2008. 2.1

[211] Moshe Y Vardi. Automatic verification of probabilistic concurrent finite state programs. In *26th Annual Symposium on Foundations of Computer Science*, pages 327–338. IEEE, 1985. 6.3

[212] Bert Vogelstein and Kenneth W Kinzler. Cancer genes and the pathways they control. *Nature Medicine*, 10(8):789–799, 2004. 2.1

[213] Daniel D Von Hoff, Thomas Ervin, Francis P Arena, E Gabriela Chiorean, Jeffrey Infante, Malcolm Moore, Thomas Seay, Sergei A Tjulandin, Wen Wee Ma, Mansoor N Saleh, et al. Increased survival in pancreatic cancer with nab-paclitaxel plus gemcitabine. *New England Journal of Medicine*, 369(18):1691–1703, 2013. 6.4

[214] Alain Vonlaufen, Swapna Joshi, Changfa Qu, Phoebe A Phillips, Zhihong Xu, Nicole R Parker, Cheryl S Toi, Romano C Pirola, Jeremy S Wilson, David Goldstein, et al. Pancreatic stellate cells: partners in crime with pancreatic cancer cells. *Cancer Research*, 68(7):2085–2093, 2008. 6.1, 6.4

[215] Abraham Wald. Sequential tests of statistical hypotheses. *The Annals of Mathematical Statistics*, 16(2):117–186, 1945. 5.2

[216] Kimberly Walter, Noriyuki Omura, Seung-Mo Hong, Margaret Griffith, Audrey Vincent, Michael Borges, and Michael Goggins. Overexpression of smoothened activates the sonic hedgehog signaling pathway in pancreatic cancer–associated fibroblasts. *Clinical Cancer Research*, 16(6):1781–1789, 2010. 2.1

[217] Qinsi Wang, Natasa Miskov-Zivanov, Cheryl Telmer, and Edmund M Clarke. Formal analysis provides parameters for guiding hyperoxidation in bacteria using phototoxic proteins. In *Proceedings of the 25th edition on Great Lakes Symposium on VLSI*, pages 315–320. ACM, 2015. 1.1

[218] Qinsi Wang, Paolo Zuliani, Soonho Kong, Sicun Gao, and Edmund M Clarke. SReach: A probabilistic bounded delta-reachability analyzer for stochastic hybrid systems. In *Computational Methods in Systems Biology: 13th International Conference, CMSB 2015, Nantes, France, September 16-18, 2015, Proceedings*, volume 9308, page 15. Springer, 2015. 1.1, 1.2

[219] Qinsi Wang, Natasa Miskov-Zivanov, Bing Liu, James R Faeder, Michael Lotze, and Edmund M Clarke. Formal modeling and analysis of pancreatic cancer microenvironment. In *International Conference on Computational Methods in Systems Biology*, pages 289–305. Springer, 2016. 1.1

[220] John G White, Eileen Southgate, J Nichol Thomson, and Sydney Brenner. The structure of the ventral nerve cord of Caenorhabditis elegans. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 275(938):327–348, 1976. 5.4.3

[221] John G White, Eileen Southgate, J Nichol Thomson, and Sydney Brenner. The structure of the nervous system of the nematode Caenorhabditis elegans. *Philosophical Transactions of the Royal Society of London B: Biological Sciences*, 314(1165):1–340, 1986. 5.4.3, 5.4.3

[222] Stephen R Wicks and Catharine H Rankin. Integration of mechanosensory stimuli in Caenorhabditis elegans. *The Journal of Neuroscience*, 15(3):2434–2444, 1995. (document), 5.4.3, 5.4.3, 5.6, 5.4.3, 5.4.3

[223] Stephen R Wicks, Chris J Roehrig, and Catharine H Rankin. A dynamic network simulation of the nematode tap withdrawal circuit: Predictions concerning synaptic function using behavioral criteria. *The Journal of Neuroscience*, 16(12):4017–4031, 1996. 5.4.3, 5.4.3, 5.4.3, 5.4.3, 5.4.3

[224] Robb E Wilentz, Christine A Iacobuzio-Donahue, Pedram Argani, Denis M McCarthy, Jennifer L Parsons, Charles J Yeo, Scott E Kern, and Ralph H Hruban. Loss of expression of DPC4 in pancreatic intraepithelial neoplasia: evidence that DPC4 inactivation occurs late in neoplastic progression. *Cancer Research*, 60(7):2002–2006, 2000. 2.4

[225] Andreas Wodarz and Roel Nusse. Mechanisms of Wnt signaling in development. *Annual Review of Cell and Developmental Biology*, 14(1):59–88, 1998. 2.1

[226] William Barry Wood et al. *The nematode Caenorhabditis elegans.* Cold Spring Harbour Laboratory, 1987. 5.4.3

[227] Guang Yao, Tae Jun Lee, Seiichi Mori, Joseph R Nevins, and Lingchong You. A bistable Rb–E2F switch underlies the restriction point. *Nature Cell Biology*, 10(4):476–482, 2008. 2.1

[228] P Ye, E Entcheva, SA Smolka, and R Grosu. Modeling excitable cells using cycle-linear hybrid automata. *IET Systems Biology*, 2(1):24–32, 2008. 1.1

[229] Hakan L Younes. Verification and planning for stochastic processes with asynchronous events. Technical report, DTIC Document, 2005. 5.2

[230] Håkan LS Younes. Ymer: a statistical model checker. In *Computer-Aided Verification*, pages 429–433. Springer, 2005. 1.2

[231] Gang Zeng, Matt Germinaro, Amanda Micsenyi, Navjot K Monga, Aaron Bell, Ajit Sood, Vanita Malhotra, Neena Sood, Vandana Midda, Dulabh K Monga, et al. Aberrant Wnt/β-catenin signaling in pancreatic adenocarcinoma. *Neoplasia*, 8(4):279–289, 2006. 2.1

[232] Lijun Zhang, Zhikun She, Stefan Ratschan, Holger Hermanns, and Ernst Moritz Hahn. Safety verification for probabilistic hybrid systems. *European Journal of Control*, 18(6): 572–587, 2012. 1.2

[233] Mei Zhen and Aravinthan DT Samuel. C. elegans locomotion: small circuits, complex functions. *Current Opinion in Neurobiology*, 33:117–126, 2015. 5.4.3

[234] Paolo Zuliani, André Platzer, and Edmund M Clarke. Bayesian statistical model checking with application to simulink/stateflow verification. In *Proceedings of the 13th ACM international conference on Hybrid Systems: Computation and Control*, pages 243–252. ACM, 2010. 1.2, 5.2