`Z3-4biology` SMT-based analysis of biological computation[*]

Boyan Yordanov Christoph M. Wintersteiger Youssef Hamadi Hillel Kugler

Abstract

Many of the basic principles governing the development and function of living organisms remain poorly understood, despite the significant progress in molecular and cellular biology and the tremendous breakthroughs in experimental methods, which allow the precise measurement and observations of the underlying biological processes. The development of system-level, mechanistic, computational models has the potential to become an indispensable foundation and tool for improving our understanding of biological systems, enabling the integration of different processes into coherent and rigorous representations that can be simulated or analyzed and whose outcomes can be compared to laboratory data and used to guide experimental work.

An intriguing property of biological systems is their ability to perform computation using information processing mechanisms and thus ensuring robust development and survival under different environments. Computational modeling has the potential to lay a foundational theory for this inherent *biological computation*, allowing the study of questions like what cells compute, how do they perform computation, and in what ways can such computation be modified or engineered, for example, to allow intervening with abnormal behavior that can lead to disease or to design computational devices using biological principles.

Inspired by the study of other computational systems, the application of formal methods has recently attracted attention in the context of biology as a strategy for automatic analysis from rich specifications, leading to rigorously defined modeling languages that can be analyzed systematically and efficiently, enabling proving of system properties and not relying only on simulation. However, such methods are often implemented as stand-alone tools focused on specific problems, thereby hindering the reproducibility of results and the collaborative improvement of methods.

Here we focus on several widely used formalisms for modeling biological systems and describe how they can be encoded to allow analysis using Satisfiability Modulo Theories (SMT) based methods. This leads to a framework that is expressive (can capture a variety of formalisms and specifications), scalable (can handle models of practical interest) and extensible (additional models and analysis procedures can be integrated). We implement the proposed translation procedures for the considered model classes, produce an initial repository of models of genetic and signaling networks, developmental systems and DNA computing circuits and demonstrate several analysis procedures for challenging examples.

We share all models and specifications from this study on our web site using the popular SMT-LIB format, which allows a number of SMT solvers to be used on the same problems. In this report we present results from the application of the Z3 solver, which provides an interactive online environment, where the available models and analysis procedures can be explored or novel ones can be developed and tested. By making our work accessible, we hope to stimulate interdisciplinary research on the application of formal methods in biology allowing better understanding of the computational processes within living organisms.

1 Introduction

Biology strives to describe and understand the different processes taking place in living organisms. One strategy for dealing with the complexity of biological systems involves the construction and use of models

[*]First made available March, 2012. Last updated June, 2012.

to systematize observations, describe mechanisms and make predictions that can be tested experimentally. Traditionally, mathematical (*e.g.* differential equation) models have been used in the field but, more recently, computational models (where an executable algorithm is developed to capture the behavior of a system) have gained popularity (see [33, 22] for reviews). The desire to establish more complete descriptions of biological systems leads to increasingly larger models, requiring novel (automated) analysis techniques and tools to aid in model development, testing and exploration.

In the past, biology has focused more on specific details rather than general laws and abstractions but the study of *biological computation* attempts to understand biological mechanisms as information processing and thus provide unified biological theories. Along this direction, it is natural to study biological computation using methods developed for the analysis of other computational systems such as computer software and hardware, where remarkable complexity can be engineered through the use of abstractions and automated analysis and design methods. In this report we review (variants of) several popular formalisms used for modeling biological systems, including Boolean networks [48] (or their extensions [22, 75]), chemical reaction network models, statecharts [38] and scenario-based models. We show that all these model classes share a common structure and can be represented by (finite) transition systems. Transition systems are also used as models of computer hardware and software and their analysis has been studied extensively in the field of formal verification, leading to the development of automated methods for reasoning about (the correctness of) system behavior. Exploiting the similarities between the formalisms used for modeling biological and other computational systems allows us to apply formal methods in the context of biology, with the potential to impact the study of biological computation.

The field of formal verification has shown tremendous progress over the past years, and has produced mature methods capable of dealing with industrial-sizes problems. Early automated verification (model checking) techniques relied on explicit transition system representations and could not handle larger models. To deal with this, symbolic methods were developed to represent and manipulate finite transition systems efficiently. The use of binary decision diagrams (BDDs) provides one such approach but the size of BDD-based symbolic representations cannot be controlled easily. As an alternative, methods based on the encoding of models and specifications as Boolean satisfiability (SAT) problems allowed the use of modern SAT solvers, which can practically handle very large problem instances despite theoretical hardness results. Such techniques have demonstrated great potential for certain applications such as the search for violating behavior ("bugs") using bounded model checking (BMC). In general, BDD and SAT-based techniques can be complementary and better suited for different problems.

A major challenge to the adoption of SAT-based methods for novel application domains such as the analysis of biological computation comes from the difficulty of translating models and specifications to Boolean satisfiability problems. Using theories richer than Boolean logic can offer a more natural framework by allowing higher-level problem descriptions, provided that automatic reasoning procedures are still available. For example, in the context of bounded model checking, the use of quantified Boolean formulas (QBF) has enabled more compact problem encodings at the price of harder decision procedures [24, 44]. In this report we demonstrate that, similarly to computer hardware and software, several formalisms used for the modeling of biological systems can be encoded naturally using bit-vectors. The use of such theories is attractive due to recently developed quantified bit-vector formula procedures [77]. An even more general framework can rely on the set of theories supported by modern satisfiability modulo theories (SMT) solvers such as Z3 [23], which include bit-vectors, integer and real arithmetic, arrays and other (possibly recursive) data types. There, the choice of specific theories is guided by the trade-off between expressivity, which allows more compact and straightforward model encodings, and decision procedure complexity.

In this report, we describe how a variety of biological models from several popular formalisms can be encoded to enable SMT-based analysis. The richness of the SMT problem can accommodate analysis procedures to address a diverse set of biological questions, while the availability of SMT solvers (which are being improved actively [1]) allows the framework to benefit from ongoing progress in the field. We illustrate the capacity of the proposed methods through the analysis of several challenging examples from a range of areas including developmental biology, synthetic biology and DNA computing. By exploiting a unified representation, we focus on studying the computation performed by biological systems independently of the specific formalism used to model them. More practically, this allows models from different classes (or integrating several formalisms) to be explored or compared efficiently to help researchers gain novel biological insight or uncover modeling inconsistencies, while correctness guarantees are also provided.

The proposed methods depend on the availability of (quantified) bit-vectors reasoning procedures but are not tied to a specific SMT solver. In fact, by using the widely adopted SMT-LIB format [8], we allow models and analysis procedures to be exchanged across platforms. As an illustration, in the presented case studies we use the Z3 solver [23] due to its performance in recent trials [1], its continuously ongoing development, and its availability as part of an interactive online environment [2].

We believe that interdisciplinary research on the intersection of formal methods and biology can help address many challenging problems such as the ones encountered in the study of biological computation. We hope that an extensible and open framework can aid in the development of improved biological models and analysis procedures through the combined effort of both communities, and improve the reproducibility of results [43]. Towards this goal, in this report we review theoretical and experimental results from both fields and demonstrate preliminary steps towards the encoding and analysis of several biological modeling formalisms through formal, SMT-based methods. In addition, we compile an initial repository of models and relevant specifications from literature and adopt the standard SMT-LIB format [8] to make them available on our project web site. This will allow both experimenting with different available models and analysis procedures or the development of novel ones, possibly using Z3's interactive online environment [2].

2 Background

Let $\mathbb{B} = \{0, 1\}$ denote the set of Boolean (truth) values. A *bit-vector* $b \in \mathbb{B}^N$ of size N is a vector of Boolean variables $b(0), b(1) \dots b(N-1)$ where $b(i) \in \mathbb{B}$ for each $0 \leq i < N$.. We use the standard logic operations on Boolean variables and bit-vectors[1], the arithmetic operations $+$ and $-$ on (unsigned) bit-vectors and functions defined on these types. All logical and arithmetic operations on Boolean variables and bit-vectors used throughout the report are supported by the Z3 solver [23]. For $b, b' \in \mathbb{B}^N$, we write $b(i) = b(j)$ instead of $b(i) \leftrightarrow b(j)$, $b = b'$ as a shorthand for $\bigwedge_{i=0}^{N-1} b(i) = b'(i)$, and use $|b| = n$ to denote a *cardinality constraint*, indicating that $b(i) = 1$ for exactly n distinct values of i.

3 Transition Systems

A *transition system* is a tuple $\mathcal{T} = (Q, Q_0, T)$ where Q is a set of states, $Q_0 \subseteq Q$ is a set of initial states and $T \subseteq Q \times Q$ is a transition relation. Given states $q, q' \in Q$ a transition $(q, q') \in T$ is denoted by $T(q, q')$. In this report only finite transition systems (where Q is finite) are considered - dealing effectively with the unbounded state spaces which can arise when modeling certain biological systems remains an open challenge and additional abstraction methods and theorem proving techniques may be needed. Finite transition systems can be encoded naturally as logical formulas [10, 24] - here we represent system states as bit-vectors $Q \subseteq \mathbb{B}^N$ where[2] $N \geq \lceil \lg(|S|) \rceil$.

A transition system is *deterministic* if for all $q \in Q$ there exists at most one $q' \in Q$ such that $T(q, q')$. It is *deadlock-free* if for all $q \in Q$ there exists $q' \in Q$ such that $q \neq q'$ and $T(q, q')$ and otherwise, a deadlock state violating this property can be identified. A *path* π is a sequence $\pi = \pi(0)\pi(1)\pi(2)\dots$ such that $\pi(0) \in Q_0$ and, for all $i \geq 0$, $\pi(i) \in Q$ and $T(\pi(i), \pi(i+1))$. A finite path of length k denoted by π_k and can be encoded by "unrolling" the transition relation [10], which results in the following quantifier-free bit-vector formula

$$[\pi_k] := \bigwedge_{i=0}^{k-1} T(\pi_k(i), \pi_k(i+1)). \tag{1}$$

Note that $[\pi_k]$ describes all valid systems paths of length k but a specific (instantiated) path can be obtained *e.g.* using Z3. The path representation described above requires k copies of the transition relation T which,

[1] We use $\wedge_{bv}$ and $\vee_{bv}$ to denote the bitwise AND and OR operations.
[2] $|Q|$ is the cardinality of Q, and $\lceil x \rceil$ denotes the ceiling of x.

in general, is a large sub-formula. Equivalent path encodings that include only a single copy of the transition relation (regardless of k) can be obtained through the use of quantifiers [24]:

$$[\pi_k]_q := \forall q, q', \left(\bigvee_{i=0}^{k-1} (q = \pi_k(i)) \wedge (q' = \pi_k(i+1)) \right) \rightarrow T(\pi_k(i), \pi_k(i+1)). \tag{2}$$

A path without loops is called a *simple path* and can be encoded as

$$[\pi_k]_s := [\pi_k] \wedge \left(\bigwedge_{0 \leq i < j \leq k} \pi_k(i) \neq \pi_k(j) \right). \tag{3}$$

This encoding is quadratic in the path length k but more compact (linear) encodings are possible through the use of quantifiers [44].

The *diameter* d of a transition system is the length of the longest shortest path between any two reachable states in the system while its *recurrence diameter* d_r is the length of the longest simple path[3]. The diameter of a transition system can be computed as the minimal d making the following quantified formula satisfiable [10]

$$\begin{gathered} \forall \pi(0) \ldots \pi(d+1), \exists \pi'(0) \ldots \pi(d), \bigwedge_{i=0}^{d} T(\pi(i), \pi(i+1)) \rightarrow \\ \left(\pi(0) = \pi'(0) \wedge \bigwedge_{i=0}^{d-1} T(\pi'(i), \pi'(i+1)) \wedge \bigvee_{i=0}^{d} \pi'(i) = \pi(d+1) \right), \end{gathered} \tag{4}$$

although some additional optimizations are possible [61]. The recurrence diameter can be computed as the maximal d_r for which the formula from Eqn. (3) is satisfiable or as the minimal d_r making the following formula satisfiable [10]

$$\forall \pi(0), \ldots, \pi(d_r+1), \bigwedge_{i=0}^{d_r} T(\pi(i), \pi(i+1)) \rightarrow \bigvee_{i=0}^{d_r} \pi(i) = \pi(d_r+1). \tag{5}$$

In bounded model checking (BMC), the diameters of a system provide (an over-approximation of) the *completeness threshold* - the minimal unrolling depth that must be considered in order to guarantee that infinite paths of the system satisfy certain properties [10, 50]. When testing the existence of a path that eventually reaches a state with given properties, which is equivalent to finding a counterexample to a specification requiring that a state with these properties is never reached, the completeness threshold is less than the (initialized) diameter d [16]. In contrast, when testing if a path exists along which certain behavior always holds (*i.e.* a specification is satisfied by all states along the path), which corresponds to a counterexample of the reachability of a state where the properties do not hold, d_r can be used as a completeness threshold [50, 16]. For both types of properties, the recurrence diameter provides an over-approximation of the completeness threshold. Identifying the recurrence diameter (*e.g.* using the procedure described in [50]) can be challenging but some of the biological models we study have small recurrence diameters, which we compute by testing the satisfiability of Eqn. (3) at increasing values of k.

4 Boolean Networks

In organisms, certain DNA sequences (genes) are expressed to produce proteins, which perform a variety of biological functions. Proteins serving as transcription factors are capable of binding regulatory DNA segments in order to affect (activate or repress) the expression of other genes. The expression of transcription factors

[3]In practice, the initialized diameters (defined by paths initialized in Q_0) are often considered instead

can also be regulated, which gives rise to intricate gene regulation networks (GRNs) capable of producing complex temporal patterns of gene expression. Similar activation/repression interactions are also observed in cell signalling and, therefore, some of the computational methods developed to study these types of networks are related.

Several formalism have been applied to model signalling and GRNs [22]. With limited experimental data, discrete-state (qualitative) models are suitable for capturing the available knowledge about the system. Here, we focus specifically on the *Boolean network* formalism, where species are considered as either present or absent (genes are either activated or repressed). Finer-grained qualitative formalisms, where species have multiple intermediate concentrations (*e.g.* low, medium, high), are also possible [75, 22] and the methods we present can be extended naturally to such models but, for the sake of presentation, we consider only the simplest formulation.

A wide variety of models have been constructed using the Boolean network formalism. Three different signalling models including plant stomatal opening/closing regulation [57], T cell large granular lymphocyte survival in leukemia [79], and respiratory immune response to bacterial infection [74] were gathered in [3]. In addition, seven Boolean network models were gathered in [27]. These include gene regulation models of *Arabidopsis* flower morphogenesis [14] (which improves the model from [30] and reduces it to the Boolean formalism), T helper cell differentiation [59], fission yeast [20] and budding yeast [56] cell cycles, and *Drosophila* development [4], as well as T-cell receptor signaling [49] and mammalian cell-cycle control [31]. The latter set of models was studied in [27] where the numbers and lengths of attractors and network diameters were computed.

In [19] a model of regulatory and metabolic networks in E. Coli was reconstructed from literature and experimental data. This model was studied as a Boolean network in [68], where extensive simulation was employed for analysis. However, due to the large state space of the system (583 genes, 96 external metabolites, environmental conditions, etc) random simulations can only cover a small subset (and potentially miss important behavior), but achieving more complete coverage through the application of explicit state exploration procedures is infeasible. An argument based on the particular network structure was used in [68] to demonstrate that the system stabilizes but such analysis is manual and case specific.

In the following, we describe SMT-based encodings of Boolean networks (Sec. 4.1) and procedures for exploring system stability (Sec. 4.2) and studying the effects of gene knockouts (Sec. 4.3). We illustrate this approach using the collection of models from [27] and demonstrate that the methods can handle challenging systems, such as the ones from [19, 68].

4.1 SMT encoding of Boolean Networks

A Boolean network [48] can be viewed as a directed graph (Fig 1-a), where nodes are associated with Boolean variables representing the states (available or not, active or inactive) of the various modeled species (*e.g.* genes, proteins or chemicals). We treat a Boolean network with N nodes (equal to the number of species in the model) as a transition system $\mathcal{T} = (Q, Q_0, T)$ where $Q = \mathbb{B}^N$ and $Q_0 = Q$ (*i.e.* all states are initial unless otherwise noted). Given a state $q \in Q$, $q(i) = 1$ iff species i is present (activated). The dynamics of the system are defined by a set of total functions $f_0, \ldots, f_{N-1}$, $f_i : \mathbb{B}^N \to \mathbb{B}$. The state of node i is updated according to function f_i and usually depends on a small number of other nodes - these relationships are represented by the edges of the graph (Fig 1-a). Both *synchronous* (where all nodes of the network are updated at each time step) and *asynchronous* (where only a single node is updated per step) executions are possible. We encode the dynamics of a Boolean network model by constructing a transition relation (Fig 1-b). Given states $q, q' \in Q$ which represent the current and updated state of the network, the transition relation for a synchronous Boolean network (where all nodes are updated at each time step) is defined as

$$T(q, q') \leftrightarrow \left(\bigwedge_{i=0}^{N-1} q'(i) = f_i(q) \right) \tag{6}$$

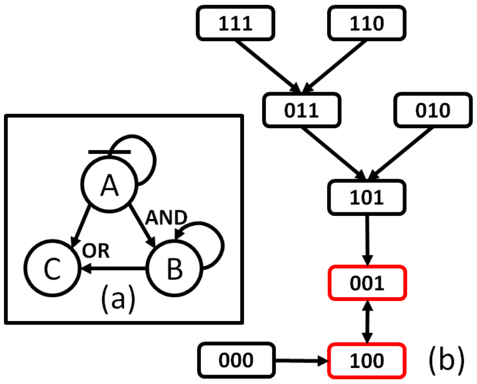

Figure 1: A simple Boolean network with 3 nodes (A,B,C) representing different genes is shown in (a). The state of the system is encoded as a single bit-vector $b \in \mathbb{B}^3$, indicating the activity of each gene. The dynamics of the system are defined by the update functions $f_1(b) = \neg b(1)$,$f_2(b) = b(1) \wedge b(2)$, and $f_3(b) = b(1) \vee b(2)$ according to Eqn. (6) or (7). The state space of the system and its synchronous transition relation representation are shown in (b), where the only attractor is highlighted.

For asynchronous updates, the transition relation is defined by

$$T(q, q') \leftrightarrow \bigvee_{i=0}^{N-1} \left(q'(i) = f_i(q) \wedge \bigwedge_{j=0, j \neq i}^{N-1} q'(j) = q(j) \right) \tag{7}$$

Both the synchronous transition relation defined by Eqn. (6) and the asynchronous one defined by Eqn. (7) result in a deadlock-free transition system $\mathcal{T}$. However, for synchronous updates $\mathcal{T}$ is deterministic while, in general, it is nondeterministic for asynchronous ones. Even when updates are synchronous, nondeterminism can be introduced explicitly, for example, by allowing the choice between several possible update functions $f_i^0, f_i^1 \ldots, f_i^K$ at a node i of the network

$$T(q, q') \leftrightarrow \bigwedge_{i=0}^{N-1} \left(\bigvee_{j=0}^{K} q'(i) = f_i^j(q) \right). \tag{8}$$

When asynchronous updates are considered, additional *fairness requirements* can be introduced (*e.g.* to specify that each node of the network is eventually updated).

4.2 System stability

We are interested in defining and studying the stability of the biological models we consider by analyzing their finite transition system representations. We focus on (nondeterministic) deadlock-free transition systems where all states are initial. Since biological systems operate in noisy environments, it is unreasonable to assume that a set of transitions is followed infinitely often when other possibilities exists - we capture this intuition as fairness. In this setting, a system has the capacity to stabilize only if there exists a state with a self loop but no other outgoing transitions - all other states will be left eventually. For example, a state with a self loop and a transition to another state resembles an unstable fixed point of a dynamical system (*i.e.* the system can remain in such a state for a period of time but will be forced to exit by environmental perturbation). Similarly, the system has the capacity of oscillations only if a cycle that cannot be escaped exists and, in some cases, both stabilization and oscillations are possible.

In the context of the deterministic transition systems resulting from Eqn. (6), stabilization is possible only if formula $T(q, q)$ is satisfiable (*i.e.* a state with a self loop exists). Given a path π_k, the possibility of

Model name	N	max(k)	Behavior	time (sec)
toy (Fig. 1)	3	2	oscil	0.015
arabidopsis [14, 27]	15	10	stabil	0.09
budding yeast [56, 27]	12	18	stabil	0.23
drosophila [4, 27]	52	34	stabil	7.5
fission yeast [20, 27]	10	6	stabil	0.03
mammalian [56, 27]	10	7	both	0.05
tcr [49, 27]	40	6	both	0.06
t helper [59, 27]	23	11	stabil	0.07
metabolic regulation [19, 68]	693	7	stabil	74.44

Table 1: Stability analysis and computation times for the set of Boolean network models available from [27] and [68]. All computation is performed on a cluster of 2.5 Ghz Intel L5420 CPUs with a 2GB memory limit per benchmark.

oscillations can be confirmed when formula $\pi_k(0) \neq \pi_k(1) \wedge \pi_k(0) = \pi_k(k)$ is satisfiable for any k (*i.e.* π_k visits at least two different states and completes a cycle). While the procedure requires testing the existence of cycles of lengths up to $k = d_r$ when no shorter cycles exist, a short recurrence diameter is characteristic of some biological models. This is the case for the Boolean network models from [27] and [68] (Table 1), even for large N (*e.g.* $N = 693$ and $d_r = 7$ for the metabolic regulation network from [68]).

4.3 Gene knockouts

When analyzing biological models, we are often interested in understanding how certain perturbation to the system affect its behavior. To illustrate this, we focus on studying how the inactivation of certain nodes (corresponding to *gene knockout* experiments in the context of gene networks) affects the stability of a Boolean network. To formalize these perturbations for a network with N nodes, we introduce the bit-vector $ko \in \mathbb{B}^N$, where $ko(i) = 1$ indicates that gene i has been knocked out. We construct the modified transition system $\mathcal{T}' = (Q', Q'_0, T')$ where $q \in Q'$ iff $(q \wedge_{bv} \neg ko) = q$ (*i.e.* knocked out genes are never active), $Q'_0 = Q'$ and

$$T'(q, q') \leftrightarrow \left(\bigwedge_{i=0}^{N-1} [q'(i) = (f_i(q) \wedge \neg ko(i))] \right). \tag{9}$$

(*i.e.* for a given knockout $ko(i) = 1$, node i remains inactive ($q'(i) = 0$) regardless the system's state q and dynamics f_i). We focus on deadlock-free, deterministic systems and consider paths π_k and π'_k of $\mathcal{T}$ and $\mathcal{T}'$, defined by unrolling T (Eqn. (6)) and T' (Eqn. (9)), respectively. Similarly to the procedure from Sec. 4.2, we require that these paths are sufficiently long, thus $k = d'_r$ where $d'_r \geq d$ is the recurrence diameter of $\mathcal{T}'$ (*i.e.* the length of the longest simple path for all possible gene knockouts). We specify that, besides inactivated nodes, the initial states of both paths are the same (*i.e.* $(\pi_k(0) \vee_{bv} ko) = (\pi'_k(0) \vee_{bv} ko)$) and the stability of the network has been affected by the knockout:

$$T(\pi_k(k), \pi_k(k)) \leftrightarrow \neg T(\pi'_k(k), \pi'_k(k)) \tag{10}$$

(*i.e.* π_k led to stabilization while, after the perturbation, π'_k leads to oscillations or vice versa). The required number of knockouts can be controlled through cardinality constraints on ko (*e.g.* $|ko| = 1$ for a single knockout). A ko satisfying the constraints described above identifies the gene knockouts that affect network stability - this procedure is applied on the models from [27] and [68], where all satisfying single gene knockouts are identified exhaustively (results are presented in Table 2).

5 Chemical Reaction Networks

In Sec. 4, we focused on a Boolean formalism for modeling the interaction networks involved in gene regulation and signalling. Such systems can also be described at a finer level of detail by a set of species (DNA sequences, proteins, chemicals, complexes), species concentrations (numbers of molecules of each species) and

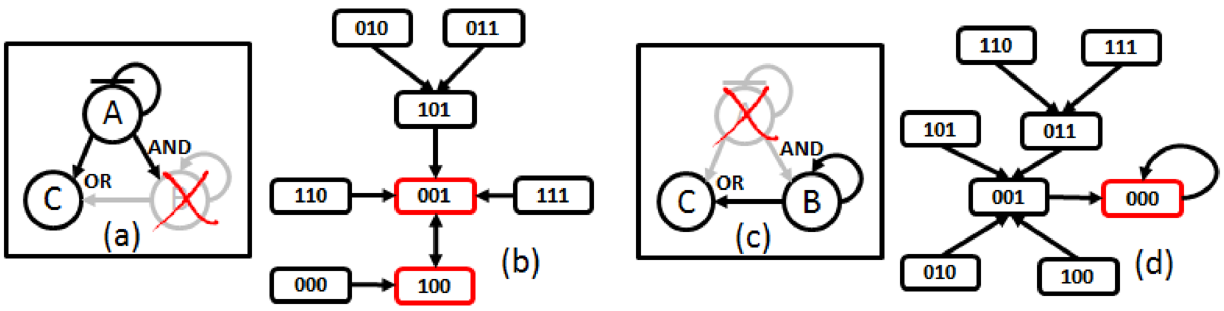

Figure 2: Inactivating gene B of the Boolean network from Fig. 1 as in (a) leads to stable behavior (b), although the original system was oscillating. Inactivating gene A in the system as in (c) changes the structure of the transition relation but not its stability (d).

Model	N	d'_r	Identified genes	Time (sec)
toy (Fig. 1)	3	5	A	0.14 (0.06)
arabidopsis [14, 27]	15	10	LFY, SEP	0.23 (0.12)
budding yeast [56, 27]	12	18	none	0.42 (0.27)
drosophila [4, 27]	52	43	HH2	63.33 (56.38)
fission yeast [20, 27]	10	6	none	0.09 (0.05)
mammalian [56, 27]	10	13	Rb,Cdc20,Cdh1, CycD,E2F,CycA	0.32 (0.14)
tcr [49, 27]	40	22	CD8,CD45,TCRlig, TCRphos, ZAP70 LCK, Fyn, cCbl,PAGCsk,TCRbind	2.76 (1.51)
t helper [59, 27]	23	17	none	0.3 (0.22)
metabolic regulation [19, 68]	693	8	none	688.2 (62.7)

Table 2: Results for the identification of single gene knockouts that affect the system's stability. The times required for the computation of the diameter d'_r (out of the total computation times) are shown in parentheses.

species interactions (reactions transforming certain species into others). Such *Chemical Reaction Networks* (CRNs) can be used to model a variety of biological and chemical processes. For some applications (*e.g.* modeling large metabolic networks), only the availability of species and the activity of reactions is captured but exact species concentrations are ignored [67, 76]. This results in a formalism closely related to the one discussed in Sec. 4 and in Sec. 5.2 we describe how such abstractions of CRNs can be encoded and studied using SMT-based approaches.

In other domains, modeling the precise number of molecules of each species is important in order to formalize certain specifications and analyze the system. This is the case in the emerging field of DNA computing, where exact molecule numbers can be related to the correctness of computational operations [53]. Results in the field have demonstrated that large DNA computing systems can be constructed experimentally [66], while the design of even more complicated systems is powered by domain specific programming languages [64]. This outlines a need for analysis and verification frameworks [53] capable of handling more expressive specifications and larger models.

In Sec. 5.1 we propose an encoding of CRNs suitable for DNA computing applications, where the exact concentrations (molecule numbers) of all species are captured. While the (finite) number of states of such representations can grow exponentially with the number of molecules, this strategy is targeted towards systems where the number of molecules is relatively small, which is the case in some DNA computing systems of practical interest [53] (we demonstrate such applications in Sec. 5.3). Despite the symbolic state representation, additional strategies such as abstraction might be required for systems where many molecules are present (one particular abstraction strategy is discussed in Sec. 5.2). In the proposed encoding, we ignore the rates at which different reactions occur (see Sec. 9) but capture the reachability properties of the system, which is sufficient to study certain correctness specifications in DNA computing [53](see Sec. 5.3).

5.1 SMT encoding of Chemical Reaction Networks

A Chemical Reaction Network (CRN) consists of a set of species S and a set of reactions R (Fig. 3-a). It can be viewed as a bipartite graph (Fig. 3-b), where an edge from a species node $s \in S$ to a reaction node $r \in R$ indicates that the species is a *reactant* (s is consumed by reaction r), while an edge from r to s indicates that the species is a *product* of the reaction[4]. We are interested in tracking how the amounts of species change as reactions take place and assume that the number of molecules of each species does not exceed a certain upper bound. Such an assumption does not hold for general CRN models and, when it does, additional analysis is required to confirm it. For some engineered CRNs as the ones discussed in Sec. 5.3, such bounds can be established from the initial amounts of species present (*e.g.* see Fig. 3).

We encode the amount of species $s \in S$ as a bit-vector of size $\lceil lg(M) \rceil$ where, for notational simplicity, M is the maximal number of molecules of any species. To simplify the subsequent definition of the system's dynamics as well as certain specifications in Sec. 5.3, we explicitly capture whether each reaction is active, leading to $Q \subseteq \mathbb{B}^N$ where $N = |S|\lceil lg(M) \rceil + |R|$ (see Fig. 3-c). Given a state $q \in \mathbb{B}^N$, we use $q(s) \in \mathbb{B}^{\lceil lg(M) \rceil}$ and $q(r) \in \mathbb{B}$ to denote the amount of species $s \in S$ (bit-vector extraction) and the status of reaction $r \in R$, respectively.

The *stoichiometry* of reactions (*i.e.* the amounts of participating species) is denoted using the functions $react : S \times R \to \mathbb{N}^0$ and $prod : S \times R \to \mathbb{N}^0$ (Fig. 3-a). For $s \in S$ and $r \in R$, species s is a reactant of reaction r when $react(s, r) > 0$, it is a product when $prod(s, r) > 0$, and $react(s, r)$ (resp. $prod(s, r)$) is the number of molecules of s consumed (resp. produced) through r. A reaction is active if there are enough molecules of its reactants

$$\bigwedge_{r \in R} (q(r) \leftrightarrow (\bigwedge_{s \in S} q(s) \geq react(r, s)), \tag{11}$$

which defines the valid states of the system Q. The initial set Q_0, which is often a singleton for the applications we consider in Sec. 5.3, is specified for a particular system and describes the amounts of species

[4]In this representation, reversible reactions are treated as two separate non-reversible ones (*e.g.* see r_1 and r_2 in Fig. 3).

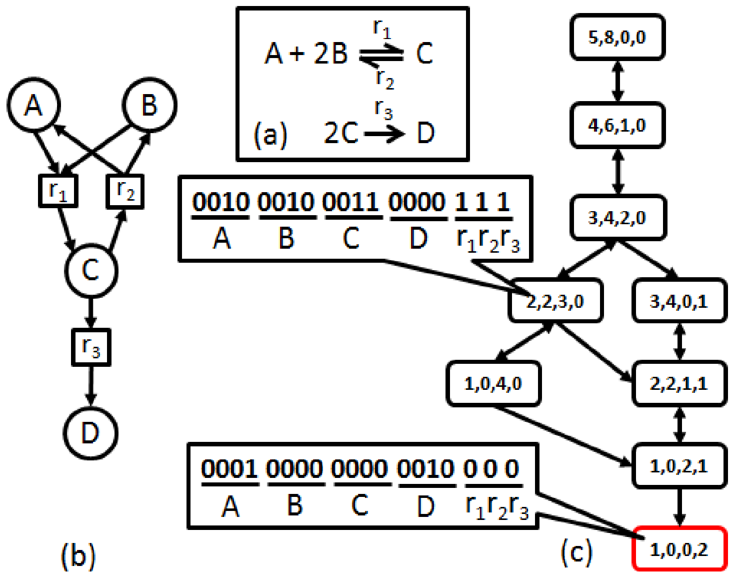

Figure 3: A chemical reaction network (a) consists of a set of reactions ($R = \{r_1, r_2, r_3\}$) and species ($S = \{A, B, C, D\}$) and can be represented as a bipartite graph (b). The dynamics of the CRN can be described by the transition system depicted graphically in (c), where the single initial state is shown in bold. The number of molecules of each species does not increase above a certain bound (determined by the initial state) and system states can be encoded using bit-vectors (c).

available initially. In a given time step, one of the active reactions takes place, which is captured by

$$T(q, q') \leftrightarrow \bigvee_{r \in R} [q(r) \wedge \bigwedge_{s \in S} q'(s) = q(s) - react(s, r) + prod(s, r)]. \tag{12}$$

5.2 Abstract Chemical Reaction Networks

The encoding described in Sec. 5.1 preserves information of the exact number of molecules of each species and the resulting state space can be large, which can make analysis challenging. One possible strategy is to abstract the exact species concentrations to a small number of qualitative values (*e.g.* low, medium, high). Utilizing such an abstraction results in models closely related to the formalism discussed in Sec. 4.

Similarly to [67, 76] we consider a coarse-grained abstraction and model species as either available or not. Given a species $s \in S$, the component $b(s) \in \mathbb{B}$ becomes a Boolean variable (rather than a bit-vector as before) and represents the availability of s and overall state of the system is represented by a bit-vector of size $N = |S| + |R|$. As before, a reaction is active only if all of its reactants are available which, in this case, is captured by the formula

$$\bigwedge_{r \in R} q(r) \leftrightarrow \left(\bigwedge_{s \in S} react(r, s) > 0 \rightarrow q(s) \right) \tag{13}$$

A species $s \in S$ is available at the next time step if and only if it is the product of a reaction that is active in the current step, which defines the transition relation

$$T(q, q') \leftrightarrow \left(\bigwedge_{s \in S} q'(s) = \left(\bigvee_{r \in R} q(r) \wedge prod(r, s) > 0 \right) \right). \tag{14}$$

gates		2	3	4	5	6	7	8	9	10
$\|S\|$		32	56	60	74	88	102	116	130	144
$\|R\|$		16	24	32	40	48	56	64	72	80
good	k	25	25	25	50	50	50	75	75	75
	time	0.8	6.2	44.2	34.3	60.2	264.4	523.3	211.2	958.9
bad	k	10	10	10	10	10	10	10	10	10
	time	0.3	0.5	0.6	0.95	1.4	1.6	1.5	2.5	2.3

Table 3: The reachability of bad (good) states within $k = \{10, 25, 50, 75, 100\}$ steps is verified for systems of 2 to 10 transducer gates in series where $M = 4$ (up to 16 molecules of each species are captured, which is sufficient to represent all devices. Only the computation times (in sec.) for the shortest tested path that led to a solution are reported.

Species that are always present can be represented as being the products of "dummy" reactions with no reactants.

As in [67], an additional modification can be introduced to capture the inactivation of reactions in the presence of certain species acting as *inhibitors*. We define the relation $inhib \subset R \times S$, where $inhib(r,s)$ indicates that reaction r is inhibited by species s. Then, the activation of reactions from Eqn. (13) is modified as

$$\bigwedge_{r\in R} q(r) \leftrightarrow \left(\bigwedge_{s\in S} react(r,s) > 0 \rightarrow q(s) \wedge inhib(r,s) \neq q(s) \right) \tag{15}$$

5.3 Verifying DNA circuits

In DNA computing, certain DNA sequences (species) are designed to interact (according to base-pairing rules), which allows the construction of artificial CRNs and, in principle, can be used to implement arbitrary computational procedures[54]. The feasibility of constructing large DNA computing circuits has been demonstrated recently [66], while progress in the field has been stimulated by the development of tools such as the DSD programming language [64], which enables the computational design and simulation of complicated DNA computing circuits. However additional methods are required to analyze the behavior of these systems [53]. Such an approach based on model-checking is proposed in [53] and is applied to study the correctness of *transducer gates*. Here, we use the encoding from Sec. 5 to analyze these circuits and demonstrate that an SMT-based approach can help uncover unwanted behavior in systems where such analysis was challenging using previous methods.

A transducer gate is a simple computational device constructed from DNA, which is intended to convert all molecules of a certain (input) species to a different (output) species [53]. Computation is initialized with a given amount of each species (including input), which defines the initial state of the system (*i.e.* $q_0 \in Q_0$, which is a singleton). Thus, the finite number of available DNA fragments limits the amount of species that can be produced as computation proceeds and, therefore, the system admits the finite representation from Sec. 5. Computation of a transducer gate terminates when no further reactions are possible. In addition, certain reactive species (denoted by $S_r \subseteq S$) must be fully consumed throughout the computation, but for some system designs this is not always the case. We distinguish between "good" and "bad" termination states, where computation has proceeded successfully and all reactive species have been consumed, or some reactive species from S_r remain and computation has failed. For a state $q \in Q$, we define

$$\begin{aligned} good(q) &\leftrightarrow \bigwedge_{r\in R} \neg q(r) \wedge \bigwedge_{s\in S_r} q(s) = 0 \\ bad(q) &\leftrightarrow \bigwedge_{r\in R} \neg q(r) \wedge \bigvee_{s\in S_r} q(s) > 0. \end{aligned}$$

A "bad" state is reachable if the following formula is satisfiable

$$\bigvee_{i=0}^{k} bad(\pi_k(i)) \wedge \bigwedge_{i=0}^{k} [T(\pi_k(i), \pi(i+1)) \vee bad(\pi_k(i))] \tag{16}$$

and a similar procedure is used to search for reachable "good" states. If (16) if unsatisfiable, a "bad" ("good") state is not reachable by executing k reactions or less but increasing k might lead to the identification of such states. Multiple transducers can be arranged in series, where the input signal must pass through each gate to produce output. We analyzed a set of systems constructed from 2 to 10 transducer gates in series using the specifications described above (see Table 3). Through the use an SMT-based procedure we showed that, while successful termination is possible for all these circuits, in each case computation might also fail.

Besides increasing the number of gates, system complexity can also be controlled by including multiple copies of each gate [53] - in this case the set of species and possible reactions remains the same but the initial state changes (*i.e.* there are twice as many molecules of each species present initially for an $N = 2$ system compared to an $N = 1$). Although this makes analysis more challenging, once a reachable "bad" state is identified in the simplest system (*e.g.* where there are $N = 1$ copies of each gate), it can be shown that a state with the same properties is reachable for a systems with other gate copy numbers. Informally, a system containing multiple gate copy numbers (*e.g.* $N = 2$) can be viewed as two systems (where $N = 1$) and, from our previous analysis, each such system can independently reach a "bad" state which has been identified. To show that a bad state is reachable for the $N = 2$ system it is sufficient to demonstrate that doubling the molecule numbers[5] for all species available in the "bad" state identified for $N = 1$ still results in a state with the same property.

6 Statecharts

Statecharts [38] have been used to model various biological systems in the areas of immunology, developmental biology and stem cell research [45, 29, 32, 70, 71]. A main advantage of this method is its appeal to experimental biologists, allowing the use of concise visual representations to describe and reason about biological models. Similar diagrammatic representations are used to describe "models" in mainstream biological articles and presentations, but they typically lack an agreed notation and semantics, which statecharts can provide, thus opening the way to being able to execute and analyze such models. Among the main features of statecharts are the hierarchy and orthogonal states that allow decomposing system behavior into modular and concise representations, and integration within an object-oriented framework [39, 40]. Biocharts, an extension and specialization of statecharts geared towards biological modeling is described in [52], which allows linking high-level (e.g. intercellular) information with lower-level (e.g. intracellular) information.

A statechart can be viewed as a hierarchical, directed graph (Fig. 4), where a set Q of *nodes*[6] and *super-nodes* (higher level nodes containing several sub-nodes) are connected through edges, labeled by events (or conditions) from a set E. We represent the overall state of the system as a bit-vector $b \in \mathbb{B}^N$ of size $N = |Q| + |E|$. A node $q \in Q$ (resp. event $e \in E$) is active if and only if $b(q)$ (resp. $b(e)$). A super-node $\hat{q} \subseteq Q$ is active if any of its sub-nodes are active (*i.e.* $b(\hat{q}) \leftrightarrow \bigvee_{q \in \hat{q}} b(q)$). Given a super-node $\hat{q} \subseteq Q$, a *XOR-type* (exclusive or) decomposition (represented graphically by encapsulating several states within a super-node) ensures that only one sub-node can be active (*i.e.* $\bigvee_{q \in \hat{q}} b(q) \wedge \bigwedge_{q,q' \in Q} \neg(b(q) \wedge b(q'))$). Similarly, in an *AND-type* decomposition (represented by separating sub-states with a dashed line), all sub-nodes are *orthogonal* and must be active (*i.e.* $\bigwedge_{q \in \hat{q}} b(q)$). An edge between nodes $q, q' \in Q$ labeled by event $e \in E$ indicates that the system can make a transition between states $b, b' \in Q$ where $b(q)$ and $b'(q')$, provided that e is active (*i.e.* $b(e)$). Similar conditions hold for edges between super-nodes, which defines the overall transition relation $T(b, b')$. Note that the events from E might be defined as functions of nodes or as (non-deterministic) external influences. There are many additional intricacies associated with the full semantics of statecharts and a complete presentation of is beyond the scope of this report.

Part of a statechart modeling the behavior of a germ cell in the "Worm" *C. elegans* appears in Fig. 4. Component Fig. 4-A describes the changes of the cell fate from a stem cell fate (Precursor) to a fertilized egg (Zygote). This component runs in parallel to genetic components in relevant pathways (modeled here as being either Active or Inactive) captured by component Fig. 4-B. In addition, the cell cycle is captured in component Fig. 4-C and cell movement in Fig. 4-D. The translation to an SMT transition relation defines

[5]Here, the maximal number of molecules considered must be sufficient to represent the larger system.

[6]although "states" is a more standard term, here this leads to confusion with the overall system state

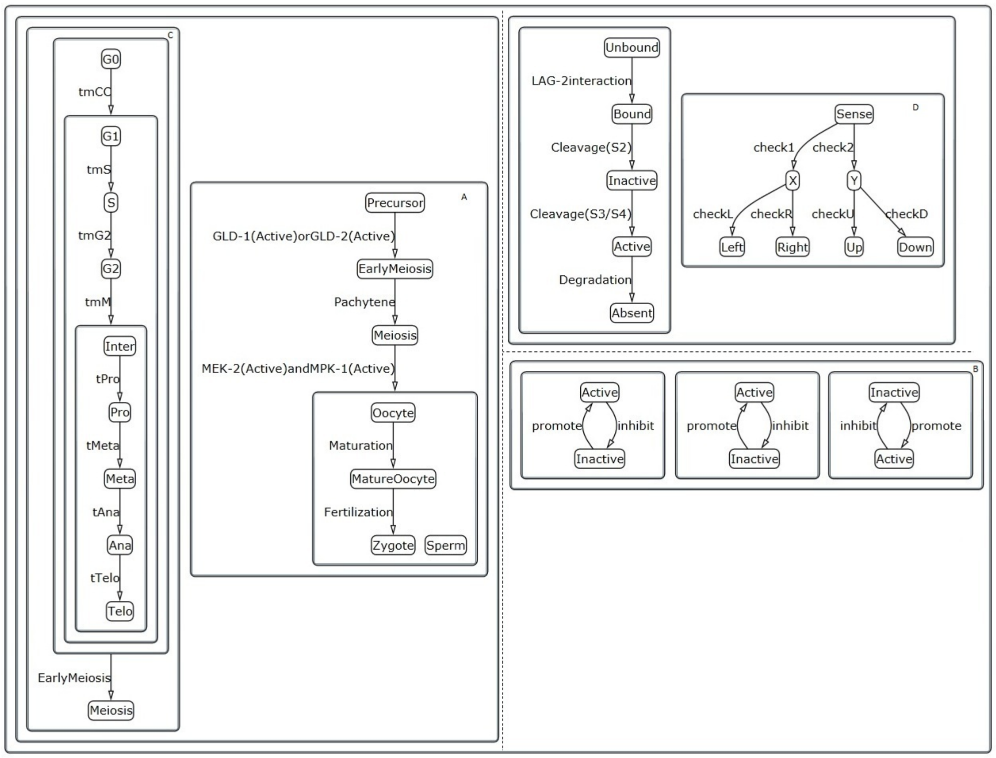

Figure 4: Germ cell statechart model

variables for all orthogonal components and any hierarchal nodes, which allows capturing the overall state of the system (the current active node for each component) and its dynamics.

7 Scenario-based models

Similarly to statecharts, scenario-based approaches offer an intuitive, visual framework for the construction of biological models. We illustrate the use of scenario-based modeling and its encoding within an SMT solver using the well studied system of cell fate specification in the vulva of the "worm" *C. elegans* [72]. This system is of particular interest as it allows studying how a set of six cells, that initially have the same fate and are thus called members of an equivalence group, acquire a specific pattern of fates during development. This system has also been extensively modeled in the past few years, see e.g., [46, 32, 47, 55, 42].

We consider the VPC model shown in Fig. 5 - a more detailed description of this scenario-based model is available in [47]. The model includes 5 *charts* (`Start`, `Levels`, `Primary_Z1_P6`, `Primary_Z1_P5`, ,`Secondary_Z1_P7`, and `Levels`). Each chart includes a number of *objects* (*e.g.* chart `Start` includes objects `Start.Env` and `Start.Worm`), which communicate through certain *methods* (*e.g.* `Start`, `Hatching`, `L1`, etc). The model describes the development of the six vulval precursor cells (VPCs), named $P3.p, P4.p, \cdots P8.p$. Two additional cells, named $Z1.ppp, Z4.aaa$ play a role in pattern formation as one of these cells will signal to the VPCs and contribute to the fate specification process. Figure 5-a shows the chart `Start`, with its two objects, `Env` and `Worm`, describing the following behavior: after the occurrence of method `Start` sent by the `Env` object to `Worm`, the Worm performs the following methods according to this ordering `Hatching`, `L1`, `L2`, `L3`, corresponding to the hatching of the Worm and the different larval developmental stages. In general, the behavior specified in the prechart (in this case `Start`) implies that the behavior specified in the main chart (in this case `Hatching`, `L1`, `L2`, `L3`) eventually occurs. This can be viewed as a liveness property in temporal logic, see [51] for a study of the relationship between LSCs and temporal logic.

Figure 5-e shows a chart `Levels`, specifying that larval stage `L3` implies sending of signals from cell `Z1.ppp` to the VPC cells `P5.p`, `P6.p` and `P7.p`. Cell `P6.p` receives a `High` signal, whereas Cells `P5.p` and `P7.p` receive `Medium` Signals. In Figure 5-b ,using separate charts `PrimaryZ1P6` (Figure 5-b), `SecondaryZ1P5` (Figure 5-c) and `SecondaryZ1P7`(Figure 5-d), cell `P6.p` adapts a Primary fate in response to the `High` signal and cells `P5.p`, `P7.p` adopt a Secondary fate in response to a Medium signal.

The basic translation follows that of [41], which originally used BDD-based solvers to analyze the models. Analysis of scenario-based models using Z3 and the theory of lists is briefly described in [60] - here our use of bit-vectors can potentially lead to a more efficient implementation. An object can be in one of several *locations* (numbered in Figure 5) and its state is represented by a bit-vector of sufficient length. The overall state of the system is represented as a bit-vector containing all objects' states, while methods are not represented explicitly as part of the state. For the model from Figure 5, a state b is a bit-vector of size $N = 26$, subject to the following constraints

$$\begin{array}{lll}
b(\texttt{Start.Env}) & \in 0\ldots 1 & (1\ bit) \\
b(\texttt{Start.Worm}) & \in 0\ldots 5 & (3\ bits) \\
b(\texttt{Levels.Worm}) & \in 0\ldots 1 & (1\ bit) \\
b(\texttt{Levels.Z1.ppp}) & \in 0\ldots 4 & (3\ bits) \\
b(\texttt{Levels.P5.p.ppp}) & \in 0\ldots 2 & (2\ bits) \\
b(\texttt{Levels.P6.p.ppp}) & \in 0\ldots 2 & (2\ bits) \\
b(\texttt{Levels.P6.p.ppp}) & \in 0\ldots 2 & (2\ bits) \\
b(\texttt{Primary\_Z1\_P6.P6.p}) & \in 0\ldots 2 & (2\ bits) \\
b(\texttt{Primary\_Z1\_P6.Z1.ppp}) & \in 0\ldots 1 & (1\ bit) \\
b(\texttt{Primary\_Z1\_P5.P5.p}) & \in 0\ldots 2 & (2\ bits) \\
b(\texttt{Primary\_Z1\_P5.Z1.ppp}) & \in 0\ldots 1 & (1\ bit) \\
b(\texttt{Secondary}_{\texttt{Z}}\texttt{1}_{\texttt{P}}\texttt{7.P7.p}) & \in 0\ldots 2 & (2\ bits) \\
b(\texttt{Secondary}_{\texttt{Z}}\texttt{1}_{\texttt{P}}\texttt{7.Z1.ppp}) & \in 0\ldots 1 & (1\ bit)
\end{array} \tag{17}$$

The transition of an object between locations coincides with the occurrence of methods, which provides a synchronization mechanism and defines the transition relation $T(b, b')$. Although the occurrence of a methods is not represented explicitly as part of the system's state, here we label it to simplify the notation.

$$\begin{array}{llll}
\texttt{Start} & = & b(\texttt{Start.Env}) = 0 & \wedge \quad b'(\texttt{Start.Env}) = 1 \\
\texttt{Hatching} & = & b(\texttt{Start.Worm}) = 1 & \wedge \quad b'(\texttt{Start.Worm}) = 2 \\
\texttt{L1} & = & b(\texttt{Start.Worm}) = 2 & \wedge \quad b'(\texttt{Start.Worm}) = 3 \\
\texttt{L2} & = & b(\texttt{Start.Worm}) = 3 & \wedge \quad b'(\texttt{Start.Worm}) = 4 \\
\texttt{L3} & = & b(\texttt{Start.Worm}) = 4 & \wedge \quad b'(\texttt{Start.Worm}) = 5 \\
\texttt{High} & = & b(\texttt{Levels.Z1.ppp}) = 1 & \wedge \quad b'(\texttt{Levels.Z1.ppp}) = 2 \\
\texttt{Medium1} & = & b(\texttt{Levels.Z1.ppp}) = 2 & \wedge \quad b'(\texttt{Levels.Z1.ppp}) = 3 \\
\texttt{Medium2} & = & b(\texttt{Levels.Z1.ppp}) = 3 & \wedge \quad b'(\texttt{Levels.Z1.ppp}) = 4 \\
\texttt{SetFatePrimary} & = & b(\texttt{Primary\_Z1\_P6.P6.p}) = 1 & \wedge \quad b'(\texttt{Primary\_Z1\_P6.P6.p}) = 2 \\
\texttt{SetFateSecondary1} & = & b(\texttt{Secondary\_Z1\_P5.P5.p}) = 1 & \wedge \quad b'(\texttt{Secondary\_Z1\_P5.P5.p}) = 2 \\
\texttt{SetFateSecondary2} & = & b(\texttt{Secondary\_Z1\_P7.P7.p}) = 1 & \wedge \quad b'(\texttt{Secondary\_Z1\_P7.P7.p}) = 2
\end{array} \tag{18}$$

The overall transition relation $T(b, b')$ is defined as

$$
\begin{array}{lcl}
T_0(b,b') & = & [b(\texttt{Start.Env}) = 0 \wedge (b(\texttt{Start.Env}) = 0 \vee b(\texttt{Start.Env}) = 1)] \vee \\
 & & [b(\texttt{Start.Env}) = 1 \wedge (b(\texttt{Start.Env}) = 0 \vee b(\texttt{Start.Env}) = 1)] \\
T_1(b,b') & = & [\neg\texttt{Start} \wedge b(\texttt{Start.Worm}) = 0 \wedge b'(\texttt{Start.Worm}) = 0] \vee \\
 & & [\texttt{Start} \wedge b(\texttt{Start.Worm}) = 0 \wedge b'(\texttt{Start.Worm}) = 1] \vee \\
 & & [b(\texttt{Start.Worm}) = 1 \wedge (b(\texttt{Start.Worm}) = 1 \vee b(\texttt{Start.Worm}) = 2)] \vee \\
 & & [b(\texttt{Start.Worm}) = 2 \wedge (b(\texttt{Start.Worm}) = 2 \vee b(\texttt{Start.Worm}) = 3)] \vee \\
 & & [b(\texttt{Start.Worm}) = 3 \wedge (b(\texttt{Start.Worm}) = 3 \vee b(\texttt{Start.Worm}) = 4)] \vee \\
 & & [b(\texttt{Start.Worm}) = 4 \wedge b(\texttt{Start.Worm}) = 4] \\
T_2(b,b') & = & [\neg\texttt{L3} \wedge b(\texttt{Levels.Worm}) = 0 \wedge b'(\texttt{Levels.Worm}) = 0] \vee \\
 & & [\texttt{L3} \wedge b(\texttt{Levels.Worm}) = 0 \wedge b'(\texttt{Levels.Worm}) = 1] \\
T_3(b,b') & = & [\neg\texttt{L3} \wedge b(\texttt{Levels.Z1.ppp}) = 0 \wedge b'(\texttt{Levels.Z1.ppp}) = 0] \vee \\
 & & [\texttt{L3} \wedge b(\texttt{Levels.Z1.ppp}) = 0 \wedge b'(\texttt{Levels.Z1.ppp}) = 1] \vee \\
 & & [b(\texttt{Levels.Z1.ppp}) = 1 \wedge (b(\texttt{Levels.Z1.ppp}) = 1 \vee b(\texttt{Levels.Z1.ppp}) = 2)] \vee \\
 & & [b(\texttt{Levels.Z1.ppp}) = 2 \wedge (b(\texttt{Levels.Z1.ppp}) = 2 \vee b(\texttt{Levels.Z1.ppp}) = 3)] \vee \\
 & & [b(\texttt{Levels.Z1.ppp}) = 3 \wedge (b(\texttt{Levels.Z1.ppp}) = 3 \vee b(\texttt{Levels.Z1.ppp}) = 4)] \vee \\
 & & [b(\texttt{Levels.Z1.ppp}) = 4 \wedge b(\texttt{Levels.Z1.ppp}) = 4] \\
T_4(b,b') & = & [\neg\texttt{L3} \wedge b(\texttt{Levels.P5.p}) = 0 \wedge b'(\texttt{Levels.P5.p}) = 0] \vee \\
 & & [\texttt{L3} \wedge b(\texttt{Levels.P5.p}) = 0 \wedge b'(\texttt{Levels.P5.p}) = 1] \vee \\
 & & [\neg\texttt{Medium1} \wedge b(\texttt{Levels.P5.p}) = 1 \wedge b'(\texttt{Levels.P5.p}) = 1] \vee \\
 & & [\texttt{Medium1} \wedge b(\texttt{Levels.P5.p}) = 1 \wedge b'(\texttt{Levels.P5.p}) = 2] \vee \\
 & & [b(\texttt{Levels.P5.p}) = 2 \wedge b'(\texttt{Levels.P5.p}) = 2] \\
T_5(b,b') & = & [\neg\texttt{L3} \wedge b(\texttt{Levels.P6.p}) = 0 \wedge b'(\texttt{Levels.P6.p}) = 0] \vee \\
 & & [\texttt{L3} \wedge b(\texttt{Levels.P6.p}) = 0 \wedge b'(\texttt{Levels.P6.p}) = 1] \vee \\
 & & [\neg\texttt{High} \wedge b(\texttt{Levels.P6.p}) = 1 \wedge b'(\texttt{Levels.P6.p}) = 1] \vee \\
 & & [\texttt{High} \wedge b(\texttt{Levels.P6.p}) = 1 \wedge b'(\texttt{Levels.P6.p}) = 2] \vee \\
 & & [b(\texttt{Levels.P6.p}) = 2 \wedge b'(\texttt{Levels.P6.p}) = 2] \\
T_6(b,b') & = & [\neg\texttt{L3} \wedge b(\texttt{Levels.P7.p}) = 0 \wedge b'(\texttt{Levels.P7.p}) = 0] \vee \\
 & & [\texttt{L3} \wedge b(\texttt{Levels.P7.p}) = 0 \wedge b'(\texttt{Levels.P7.p}) = 1] \vee \\
 & & [\neg\texttt{Medium2} \wedge b(\texttt{Levels.P7.p}) = 1 \wedge b'(\texttt{Levels.P7.p}) = 1] \vee \\
 & & [\texttt{Medium2} \wedge b(\texttt{Levels.P7.p}) = 1 \wedge b'(\texttt{Levels.P7.p}) = 2] \vee \\
 & & [b(\texttt{Levels.P7.p}) = 2 \wedge b'(\texttt{Levels.P7.p}) = 2] \\
T_7(b,b') & = & [\neg\texttt{High} \wedge b(\texttt{Primary_Z1_P6.Z1.ppp}) = 0 \wedge b'(\texttt{Primary_Z1_P6.Z1.ppp}) = 0] \vee \\
 & & [\texttt{High} \wedge b(\texttt{Primary_Z1_P6.Z1.ppp}) = 0 \wedge b'(\texttt{Primary_Z1_P6.Z1.ppp}) = 1] \\
T_8(b,b') & = & [\neg\texttt{High} \wedge b(\texttt{Primary_Z1_P6.P6.p}) = 0 \wedge b'(\texttt{Primary_Z1_P6.P6.p}) = 0] \vee \\
 & & [\texttt{High} \wedge b(\texttt{Primary_Z1_P6.P6.p}) = 0 \wedge b'(\texttt{Primary_Z1_P6.P6.p}) = 1] \vee \\
 & & [b(\texttt{Primary_Z1_P6.P6.p}) = 1 \wedge (b'(\texttt{Primary_Z1_P6.P6.p}) = 1 \vee b'(\texttt{Primary_Z1_P6.P6.p}) = 2)] \\
T_9(b,b') & = & [\neg\texttt{Medium1} \wedge b(\texttt{Secondary_Z1_P5.Z1.ppp}) = 0 \wedge b'(\texttt{Secondary_Z1_P5.Z1.ppp}) = 0] \vee \\
 & & [\texttt{Medium1} \wedge b(\texttt{Secondary_Z1_P5.Z1.ppp}) = 0 \wedge b'(\texttt{Secondary_Z1_P5.Z1.ppp}) = 1] \\
T_{10}(b,b') & = & [\neg\texttt{Medium1} \wedge b(\texttt{Secondary_Z1_P5.P5.p}) = 0 \wedge b'(\texttt{Secondary_Z1_P5.P5.p}) = 0] \vee \\
 & & [\texttt{Medium1} \wedge b(\texttt{Secondary_Z1_P5.P5.p}) = 0 \wedge b'(\texttt{Secondary_Z1_P5.P5.p}) = 1] \vee \\
 & & [b(\texttt{Secondary_Z1_P5.P5.p}) = 1 \wedge (b'(\texttt{Secondary_Z1_P5.P5.p}) = 1 \vee b'(\texttt{Secondary_Z1_P5.P5.p}) = 2)] \\
T_{11}(b,b') & = & [\neg\texttt{Medium2} \wedge b(\texttt{Secondary_Z1_P7.Z1.ppp}) = 0 \wedge b'(\texttt{Secondary_Z1_P7.Z1.ppp}) = 0] \vee \\
 & & [\texttt{Medium2} \wedge b(\texttt{Secondary_Z1_P7.Z1.ppp}) = 0 \wedge b'(\texttt{Secondary_Z1_P7.Z1.ppp}) = 1] \\
T_{12}(b,b') & = & [\neg\texttt{Medium2} \wedge b(\texttt{Secondary_Z1_P7.P7.p}) = 0 \wedge b'(\texttt{Secondary_Z1_P7.P7.p}) = 0] \vee \\
 & & [\texttt{Medium2} \wedge b(\texttt{Secondary_Z1_P7.P7.p}) = 0 \wedge b'(\texttt{Secondary_Z1_P7.P7.p}) = 1] \vee \\
 & & [b(\texttt{Secondary_Z1_P7.P7.p}) = 1 \wedge (b'(\texttt{Secondary_Z1_P7.P7.p}) = 1 \vee b'(\texttt{Secondary_Z1_P7.P7.p}) = 2)] \\
T(b,b') & \leftrightarrow & \bigwedge_{i=1}^{i=9} T_i(b,b')
\end{array}
\tag{19}
$$

The system is always initialized with all objects in location 0 and all objects reach a location other than 0 if the prechart behavior occurs.

8 Synthetic Gene Circuits

In recent years, the application of engineering principles to biological systems has been pursued extensively with the goal of taming biology as a technology. In the field of synthetic biology, novel biological systems have been designed rationally for specific functions (see [5, 65] for reviews), outlining a variety of possible applications areas from therapeutics to biofuel production. In particular, the design and re-design of gene networks is a major focus of the field with the dream of automated compilation of such *gene circuits* from high level designs or functional specifications to DNA sequences. Efforts along this direction include the development of tools for checking the syntactic correctness of circuits [11], graphical tools for circuit design [58], and programming languages for genetic engineering [63].

While research in synthetic biology has focused mostly on the construction of devices (relatively small gene circuits) from parts (DNA segments with specific function), the construction of larger-scale systems from devices can also be pursued [65]. This task is related to software engineering, where complicated programs are constructed using well-characterized software components from libraries, possibly through the use of automated methods. For example, a constraint-based synthesis approach employing an SMT solver was proposed in [37], but all components were assumed to have the same input/output type. The design of gene networks also resembles the circuit layout problem in electrical engineering, where SAT-based solutions have been developed [25]. However, while components within electrical circuits can be connected through a common signal (electricity), in gene circuits various different species (chemicals, proteins, etc) might serve as chemical "wires", which must be matched to ensure proper function. In addition, due to diffusion specific connections cannot be isolated and cross-talk between components might occur, although experimental approaches based on the construction of separate, isolated compartments have been developed [73].

In the following, we consider the constraints specific to gene circuits (including circuits with multiple compartments) and propose an encoding that enables the use of SMT methods to solve design problems. While currently only basic connectivity and exclusivity constraints are handled, the SMT framework allows more complicated design requirements to be incorporated. Such procedures can enable component-based SMT synthesis as in [37] for synthetic biology, where gene circuits that are both structurally sound and functionally correct are designed automatically.

For the design of gene circuits from separate, well-characterized components, we consider a *device* to be a gene regulation network (see Sec. 4) where certain species (*e.g.* proteins or other chemicals) are designated as inputs and outputs. When gene circuits are constructed by incorporating several devices, these input/output species act as chemical wires to connect the separate components and must be matched properly, while other constraints are also satisfied (*e.g.* if the same species is an output of two separate devices in a circuit, cross-talk might occur). While the dynamic behavior of devices might also be characterized, here we focus only on defining input/output connectivity and exclusivity constraints - considering functional specifications is a direction of future work.

We assume that a library of previously constructed devices, denoted by D is available. Each device $d \in D$ is affected by certain proteins or chemicals, which act as its inputs and expresses certain genes to produce proteins, which act as its outputs and can interact with other devices. We denote the set of all species (*i.e.* the inputs and outputs of all devices) by S. We denote the inputs and outputs of a particular device using $in \subseteq S \times D$ and $out \subseteq S \times D$, respectively. As part of the design specification, we seek to construct a system that is able to respond to given inputs denoted by $I \subseteq S$ by producing certain outputs denoted by $O \subset S$. We describe a set of states $Q \subseteq \mathbb{B}^{|D|+|S|}$ where each $q \in Q$ corresponds to a particular system design - for a device $d \in D$, $q(d) = 1$ indicates that d is used as part of this system and for a species $s \in S$, $q(s)$ indicates that s is available (*i.e.* it is either produced by one of the enabled devices or is supplied as an input). This restricts the set of valid states where, for $q \in Q$ we have

$$\bigwedge_{s \in S} q(s) \leftrightarrow \left(s \in I \vee \bigvee_{d \in D} (q(d) \wedge out(s,d)) \right). \tag{20}$$

In the following, we impose additional constraints on valid system states to capture various properties of the design problem. For a device to be enabled, all its inputs must be available

$$\bigwedge_{d \in D} b(d) \rightarrow \left(\bigwedge_{s \in S} in(s,d) \rightarrow q(s) \right). \tag{21}$$

while, in addition, all required system outputs must be available (*i.e.* $\bigwedge_{s \in O} q(s)$). To prevent device cross-talk, we specify that no two devices that share the same output species can be enabled

$$\forall d, d' \in D, \bigvee_{s \in S} (out(s,d) \wedge out(s,d')) \rightarrow \neg(q(d) \wedge q(d')) \tag{22}$$

Finally, to prevent the inclusion of devices that produce unnecessary species, we specify that a species is available only if it is the output of the circuit or an input of an enabled device

$$\bigwedge_{s \in S} q(s) \rightarrow \left(s \in O \vee \bigvee_{d \in D} (q(d) \wedge in(s,d)) \right) \tag{23}$$

To overcome design limitations imposed by the constraints described above, recent experimental methods propose distributing circuit components across different cells (compartments) [73]. We capture such functionality by considering a transition relation $T(q, q')$ where q, q' represent the enabled devices and available species at two separate compartments and satisfy all constraints outlined so far, while additional constraints can be introduced to capture the connections between compartments (*e.g.* the outputs of the q compartment should match the inputs of the q' one). In contrast to the encodings discussed in previous sections, here we use T to encodes a transition between compartments in space rather than the progress of time.

To illustrate the proposed approach to the design of gene circuits, we consider a simple device library consisting of devices $1 \dots 5$ shown in Fig. 6, where the input and output species of each devices are also specified (*e.g.* arabinose (*ara*) and the transcription factor *NRI* are the inputs of device 4 and its outputs are the transcription factors *CI* and *LacI*). This sample library was constructed from components used frequently in experimental projects in synthetic biology [65]. Our goal is to design a circuit that has *ara* as its input and produces the green fluorescent protein (*gfp*) as an output signal (device 6). The only two possible solutions satisfying all constraints described above are shown as devices 7 (constructed from devices 1 and 2) and device 8 (constructed from devices 3,4 and 5).

9 Discussion and Related Literature

In this report, we used finite transition systems as a unifying representation for several biological modeling formalisms, which allowed the development of analysis techniques that are more general and apply across model classes. We showed that, for a number of applications, finite transition systems capture the behavior that is relevant for analysis but there are certain modeling aspects which were restricted. In the following, we outline these limitations and highlight related work.

While the formalization and analysis of biological systems has already received some attention (*e.g.* [13], among others), a framework where various formalisms can be combined, analyzed and compared, together with the set of benchmarks that can help drive future developments, has been missing. The application of formal methods in biology has also been considered (*e.g.* [6, 9, 12]) but such studies are usually focused on a specific technique (*e.g.* model checking) or the analysis of a specific system class (*e.g.* gene regulation networks). To facilitate the development and integration of biological models and SMT-based analysis procedures, in this report we considered several different formalisms, which led to a more expressive and extensible framework.

In the modeling of biological systems, capturing individual molecules numbers can provide detailed and biologically accurate system descriptions, which are often difficult to analyze. For such models, the rates at which reactions occur can also be specified, which leads to models that can be studied through stochastic simulation or analyzed using probabilistic model checking as in [53]. By constructing finite transition system representations as in Sec. 5, we capture molecule numbers and reachability properties, which are crucial in certain applications (*e.g.* DNA computing as in Sec. 5.3), but ignore all probabilistic aspects. At this cost, we are able to apply SMT-based methods and analyze systems that were beyond the scope of previous approaches. SMT reasoning procedures for probabilistic systems (*e.g.* [34]) can help address some of the current limitations and extend our approach to other model classes (*e.g.* probabilistic Boolean networks).

When a sufficient number of molecules is present, species concentrations can be described as continuous values ((*e.g.* using (non-linear) ODEs) [22]. Such systems, as well as other infinite-state, continuous and hybrid models used in biology, can be encoded into SMT directly but might require expensive (or incomplete) decision procedures. As an alternative, (conservative) finite transition system abstractions can be constructed (*e.g.* as in [78]), enabling the analysis and integration of infinite state systems within the framework described here. The application of formal methods to Petri Nets [15], which also describe chemical reaction networks, has been studied extensively and can provide useful analysis procedures, which can then be extended to all the formalisms we consider through their common representation. Chemical reaction networks have also been studied at steady state using flux balance analysis (FBA) [62] but such methods cannot address certain questions (*e.g.* as in DNA computation).

In the report, we illustrate a range of analysis problems important for the study of biological models.

The DNA computing circuits we consider in Sec. 5.3 resemble more traditional computational systems and the questions we pose are encountered often in the verification of hardware and software (*e.g.* reachability of deadlock states). The identification of gene knockouts from Sec. 4.3 is closer to biology but is related to the modification (synthesis) of systems to achieve certain behavior. The related problems of revising models of GRNs and metabolic networks were considered in [18, 67].

The stability questions we study in Sec. 4.2 are inspired by dynamical systems theory but have also attracted attention from other communities [17]. Compared to [17], we are interested in distinguishing stability from oscillations rather then determining if all system executions converge to a single state - these questions are related to local vs. global stability properties. The problem of identifying attractors in Boolean networks, for which several approaches have been proposed, is closely related to studying the system's stability. A tool described in [3] focuses on simulating synchronous and asynchronous Boolean networks but also offers some steady-state and cycle detection capabilities. In [27] a SAT-based algorithm is proposed for the identification of all attractors in a synchronous Boolean network. The approach can be applied to multi-valued (generalized Boolean) networks [28] but, due to non-determinism, cannot be extended easily to asynchronous systems. A BDD-based algorithm capable of dealing with both synchronous and asynchronous execution is proposed in [36, 35], while a BDD-based randomized traversal method is studied in [7]. For more general formalisms, the search for steady-states only is formulated as a constraint satisfaction problem (CSP) in [26], while a SAT-based steady-state identification method is proposed in [21] for piecewise linear differential equations (PLDEs) through the construction of finite abstractions. In the context of metabolic or chemical reaction networks, a SAT-based approach for the identification of steady-state network configurations is discussed in [76]. This approach ignores exact species concentrations, as in the abstraction strategy and encoding we outlined in Sec. 5.2. Exploiting the modularity of biological models to allow the analysis of larger systems, a BDD-based approach for the identification of all recurrent (infinitely visited) states is described in [69], while in [17] a method for the construction of modular stabilization proofs is proposed. The application of modular analysis techniques to the formalisms discussed in this report (*e.g.* chemical reactions in compartments) is a future research direction. For many biological applications, the detection of all the system's attractors is unnecessary but instead checking whether certain specifications holds in these attractors is required - the SMT-based approaches we discuss in this report are particularly suitable for the development of such analysis procedures.

The procedures from Sec. 4 rely on the computation of system diameters - a hard problem studied in the context of bounded model checking [10, 50] which can also help uncover characteristics of biological systems (*e.g* such systems might need to switch between states quickly, resulting in shorter diameters). When identified, such biological properties can lead to specialized methods or improve existing procedures. In Sec. 5.3 we used similar insight about DNA circuits to restrict the system state space and to extend our analysis results directly to more complex models in a modular fashion. A similar strategy of combining biological intuition with powerful modular reasoning has also been pursued in [17] to address challenging analysis problems for biological models in a spatial context. Considering the different formalisms we studied in this report in a spatial context can lead to the development of more detailed models of biological systems (*e.g.* by capturing multiple cells or compartments), which can be analyzed by exploiting modularity in [69, 17].

References

[1] Satisfiability modulo theories competition (smt-comp), 2011, online at http://www.smtcomp.org/2011/, 2011.

[2] `z3-4bio` at `rise4fun` software engineering tools from Microsoft Research. online at http://rise4fun.com/z34biology, 2012.

[3] Istvan Albert, Juilee Thakar, Song Li, Ranran Zhang, and Reka Albert. Boolean network simulations for life scientists. *Source Code for Biology and Medicine*, 3(1):16, 2008.

[4] Rka 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.

[5] Andrianantoandro, S Basu, D K Karig, and R Weiss. Synthetic biology: new engineering rules for an emerging discipline. *Mol Syst Biol*, 2:2006–2006, 2006.

[6] M. Antoniotti, F. Park, A. Policriti, N. Ugel, and B. Mishra. Foundations of a query and simulation system for the modeling of biochemical and biological processes. In *Pacific Symposium on Biocomputing (PSB)*, pages 116–127, 2003.

[7] Ferhat Ay, Fei Xu, and Tamer Kahveci. Scalable steady state analysis of boolean biological regulatory networks. *PLoS ONE*, 4(12):e7992, 12 2009.

[8] Clark Barrett, Aaron Stump, and Cesare Tinelli. The SMT-LIB Standard: Version 2.0. In A. Gupta and D. Kroening, editors, *Proceedings of the 8th International Workshop on Satisfiability Modulo Theories (Edinburgh, UK)*, 2010.

[9] G. Batt, D. Ropers, H.de Jong, J. Geiselmann, R. Mateescu, M. Page, and D. Schneider. Validation of qualitative models of genetic regulatory networks by model checking: analysis of the nutritional stress response in *escherichia coli*. *ISMB (Supplement of Bioinformatics)*, pages 19–28, 2005.

[10] Armin Biere, Alessandro Cimatti, Edmund Clarke, and Yunshan Zhu. Symbolic model checking without bdds. In W. Cleaveland, editor, *Tools and Algorithms for the Construction and Analysis of Systems*, volume 1579 of *Lecture Notes in Computer Science*, pages 193–207. Springer Berlin / Heidelberg, 1999.

[11] Yizhi Cai, Brian Hartnett, Claes Gustafsson, and Jean Peccoud. A syntactic model to design and verify synthetic genetic constructs derived from standard biological parts. *Bioinformatics*, 23(20):2760–2767, 2007.

[12] Nathalie Chabrier and François Fages. Symbolic model checking of biochemical networks. In *Proceedings of the First International Workshop on Computational Methods in Systems Biology*, CMSB '03, pages 149–162, London, UK, UK, 2003. Springer-Verlag.

[13] N. Chabrier-Rivier, M. Chiaverini, V. Danos, F. Fages, and V. Schächter. Modeling and querying biomolecular interaction networks. *Theoretical Computer Science*, 325(1):25–44, 2004.

[14] lvaro Chaos, Max Aldana, Carlos Espinosa-Soto, Berenice de Len, Adriana Arroyo, and Elena Alvarez-Buylla. From genes to flower patterns and evolution: Dynamic models of gene regulatory networks. *Journal of Plant Growth Regulation*, 25:278–289, 2006. 10.1007/s00344-006-0068-8.

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

[16] Edmund M. Clarke, Daniel Kroening, Jol Ouaknine, and Ofer Strichman. Completeness and complexity of bounded model checking. In *VMCAI'04*, pages 85–96, 2004.

[17] Byron Cook, Jasmin Fisher, Elzbieta Krepska, and Nir Piterman. Proving stabilization of biological systems. In *Proceedings of the 12th international conference on Verification, model checking, and abstract interpretation*, VMCAI'11, pages 134–149, Berlin, Heidelberg, 2011. Springer-Verlag.

[18] Fabien Corblin, Eric Fanchon, and Laurent Trilling. Applications of a formal approach to decipher discrete genetic networks. *BMC Bioinformatics*, 11(1):385, 2010.

[19] Markus W. Covert, Eric M. Knight, Jennifer L. Reed, Markus J. Herrgard, and Bernhard O. Palsson. Integrating high-throughput and computational data elucidates bacterial networks. *Nature*, 429(6987):92–96, May 2004.

[20] Maria I. Davidich and Stefan Bornholdt. Boolean network model predicts cell cycle sequence of fission yeast. *PLoS ONE*, 3(2):e1672, 02 2008.

[21] H. de Jong and M. Page. Search for steady states of piecewise-linear differential equation models of genetic regulatory networks. *Computational Biology and Bioinformatics, IEEE/ACM Transactions on*, 5(2):208 –222, april-june 2008.

[22] Hidde de Jong. Modeling and simulation of genetic regulatory systems: A literature review. *Journal of Computational Biology*, 9(1):67–103, 2002.

[23] Leonardo de Moura and Nikolaj Bjørner. Z3: An efficient smt solver. In C. Ramakrishnan and Jakob Rehof, editors, *Tools and Algorithms for the Construction and Analysis of Systems*, volume 4963 of *Lecture Notes in Computer Science*, pages 337–340. Springer Berlin / Heidelberg, 2008.

[24] Nachum Dershowitz, Ziyad Hanna, and Jacob Katz. Bounded model checking with qbf. In Fahiem Bacchus and Toby Walsh, editors, *Theory and Applications of Satisfiability Testing*, volume 3569 of *Lecture Notes in Computer Science*, pages 100–101. Springer Berlin / Heidelberg, 2005.

[25] S. Devadas. Optimal layout via boolean satisfiability. In *Computer-Aided Design, 1989. ICCAD-89. Digest of Technical Papers., 1989 IEEE International Conference on*, pages 294 –297, nov 1989.

[26] Vincent Devloo, Pierre Hansen, and Martine Labb. Identification of all steady states in large networks by logical analysis. *Bulletin of Mathematical Biology*, 65:1025–1051, 2003. 10.1016/S0092-8240(03)00061-2.

[27] Elena Dubrova and Maxim Teslenko. A SAT-based algorithm for finding attractors in synchronous boolean networks. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 8:1393–1399, 2011.

[28] Elena Dubrova, Maxim Teslenko, and Liu Ming. Finding attractors in synchronous multiple-valued networks using SAT-based bounded model checking. In *Multiple-Valued Logic (ISMVL), 2010 40th IEEE International Symposium on*, pages 144 –149, may 2010.

[29] S. Efroni, D. Harel, and I.R. Cohen. Emergent Dynamics of Thymocyte Development and Lineage Determination. *PLoS Computational Biology*, 3(1):3–13, 2007.

[30] Carlos Espinosa-Soto, Pablo Padilla-Longoria, and Elena R. Alvarez-Buylla. A gene regulatory network model for cell-fate determination during arabidopsis thaliana flower development that is robust and recovers experimental gene expression profiles. *The Plant Cell Online*, 16(11):2923–2939, 2004.

[31] Adrien Fauré, Aurélien Naldi, Claudine Chaouiya, and Denis Thieffry. Dynamical analysis of a generic boolean model for the control of the mammalian cell cycle. *Bioinformatics*, 22(14):e124–e131, 2006.

[32] J. Fisher, N. Piterman, E.J.A. Hubbard, M.J. Stern, and D. Harel. Computational Insights into *C. elegans* Vulval Development. *Proceedings of the National Academy of Sciences*, 102(6):1951–1956, 2005.

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

[34] Martin Frnzle, Holger Hermanns, and Tino Teige. Stochastic satisfiability modulo theory: A novel technique for the analysis of probabilistic hybrid systems. In Magnus Egerstedt and Bud Mishra, editors, *Hybrid Systems: Computation and Control*, volume 4981 of *Lecture Notes in Computer Science*, pages 172–186. Springer Berlin / Heidelberg, 2008.

[35] Abhishek Garg, Alessandro Di Cara, Ioannis Xenarios, Luis Mendoza, and Giovanni De Micheli. Synchronous versus asynchronous modeling of gene regulatory networks. *Bioinformatics*, 24(17):1917–1925, 2008.

[36] Abhishek Garg, Ioannis Xenarios, Luis Mendoza, and Giovanni DeMicheli. An efficient method for dynamic analysis of gene regulatory networks and in silico gene perturbation experiments. In T Speed and H Huang, editors, *Research in Computational Molecular Biology, Proceedings*, volume 4453 of *Lecture Notes In Computer Science*, pages 62–76, Heidelberger Platz 3, D-14197 Berlin, Germany, 2007. Springer-Verlag Berlin. 11th Annual International Conference on Research in Computational Molecular Biology, Oakland, CA, APR 21-25, 2007.

[37] Sumit Gulwani, Susmit Jha, Ashish Tiwari, and Ramarathnam Venkatesan. Synthesis of loop-free programs. *SIGPLAN Not.*, 46:62–73, June 2011.

[38] D. Harel. Statecharts: A visual formalism for complex systems. *Science of Computer Programming*, 8:231–274, 1987. (Preliminary version: Technical Report CS84-05, The Weizmann Institute of Science, Rehovot, Israel, February 1984.).

[39] D. Harel and E. Gery. Executable object modeling with statecharts. *Computer*, 30(7):31–42, July 1997. Also in *Proc. 18th Int. Conf. Soft. Eng.*, Berlin, IEEE Press, March, 1996, pp. 246–257.

[40] D. Harel and H. Kugler. The RHAPSODY Semantics of Statecharts (or, On the Executable Core of the UML). In *Integration of Software Specification Techniques for Application in Engineering*, volume 3147 of *Lect. Notes in Comp. Sci.*, pages 325–354. Springer-Verlag, 2004.

[41] D. Harel, H. Kugler, R. Marelly, and A. Pnueli. Smart play-out of behavioral requirements. In *Proc. 4^{th} Intl. Conference on Formal Methods in Computer-Aided Design (FMCAD'02), Portland, Oregon*, volume 2517 of *Lect. Notes in Comp. Sci.*, pages 378–398, 2002. Also available as Tech. Report MCS02-08, The Weizmann Institute of Science.

[42] E. Hoyos, K. Kim, J. Milloz, M. Barkoulas, J.B. Penigault, E. Munro, and M.A. Felix. Quantitative Variation in Autocrine Signaling and Pathway Crosstalk in the Caenorhabditis Vulval Network. *Current Biology*, 21(7):527–538, 2011.

[43] Darrel C. Ince, Leslie Hatton, and John Graham-Cumming. The case for open computer programs. *Nature*, 482(7386):485–488, February 2012.

[44] Toni Jussila and Armin Biere. Compressing bmc encodings with qbf. *Electron. Notes Theor. Comput. Sci.*, 174:45–56, May 2007.

[45] N. Kam, I.R. Cohen, and D. Harel. The immune system as a reactive system: Modeling t cell activation with statecharts. In *IEEE International Symposium on Human-Centric Computing Languages and Environments (HCC 2001)*, pages 15–22. IEEE Computer Society, 2001.

[46] N. Kam, D. Harel, H. Kugler, R. Marelly, A. Pnueli, E.J.A. Hubbard, and M.J. Stern. Formal Modeling of C. elegans Development: A Scenario-Based Approach. In Corrado Priami, editor, *Proc. Int. Workshop on Computational Methods in Systems Biology (CMSB 2003)*, volume 2602 of *Lect. Notes in Comp. Sci.*, pages 4–20. Springer-Verlag, 2003. Extended version appeared in Modeling in Molecular Biology, G.Ciobanu (Ed.), Natural Computing Series, Springer, 2004 .

[47] N. Kam, H. Kugler, R. Marelly, L. Appleby, J. Fisher, A. Pnueli, D. Harel, M.J. Stern, and E.J.A. Hubbard. A scenario-based approach to modeling development: A prototype model of C. elegans vulval fate specification. *Developmental Biology*, 323(1):1–5, 2008.

[48] S.A. Kauffman. Metabolic stability and epigenesis in randomly constructed genetic nets. *Journal of Theoretical Biology*, 22(3):437 – 467, 1969.

[49] Steffen Klamt, Julio Saez-Rodriguez, Jonathan Lindquist, Luca Simeoni, and Ernst Gilles. A methodology for the structural and functional analysis of signaling and regulatory networks. *BMC Bioinformatics*, 7(1):56, 2006.

[50] Daniel Kroening and Ofer Strichman. Efficient computation of recurrence diameters. In *Proceedings of the 4th International Conference on Verification, Model Checking, and Abstract Interpretation*, VMCAI 2003, pages 298–309, London, UK, UK, 2003. Springer-Verlag.

[51] H. Kugler, D. Harel, A. Pnueli, Y. Lu, and Y. Bontemps. Temporal Logic for Scenario-Based Specifications. In N. Halbwachs and L.D. Zuck, editor, *Proc. 11^{th} Intl. Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'05)*, volume 3440 of *Lect. Notes in Comp. Sci.*, pages 445–460. Springer-Verlag, 2005.

[52] H. Kugler, A. Larjo, and D. Harel. Biocharts: A Visual Formalism for Complex Biological Systems. *J. R. Soc. Interface*, 7(48):1015–1024, 2010.

[53] Matthew R. Lakin, David Parker, Luca Cardelli, Marta Kwiatkowska, and Andrew Phillips. Design and analysis of dna strand displacement devices using probabilistic model checking. *Journal of The Royal Society Interface*, 2012.

[54] Matthew R. Lakin and Andrew Phillips. Modelling, simulating and verifying turing-powerful strand displacement systems. In *Proceedings of the 17th international conference on DNA computing and molecular programming*, DNA'11, pages 130–144, Berlin, Heidelberg, 2011. Springer-Verlag.

[55] C. Li, M. Nagasaki, K. Ueno, and S. Miyano. Simulation-based model checking approach to cell fate specification during Caenorhabditis elegans vulval development by hybrid functional Petri net with extension. *BMC Systems Biology*, 3(1), 2009.

[56] 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.

[57] Song Li, Sarah M Assmann, and Rka Albert. Predicting essential components of signal transduction networks: A dynamic model of guard cell abscisic acid signaling. *PLoS Biol*, 4(10):e312, 09 2006.

[58] M.A. Marchisio and J. Stelling. Computational design of synthetic gene circuits with composable parts. *Bioinformatics*, 24(17):1903–1910, 2008.

[59] Luis Mendoza and Ioannis Xenarios. A method for the generation of standardized qualitative dynamical systems of regulatory networks. *Theoretical Biology and Medical Modelling*, 3(1):13, 2006.

[60] A. Milicevic and H. Kugler. Model Checking Using SMT and Theory of Lists. In M. Bobaru, K. Havelund, G. Holzmann, and Gerard R. Joshi, editors, *Proc. 3^{rd} NASA Formal Methods Symposium (NFM'11)*, volume 6617 of *Lect. Notes in Comp. Sci.*, pages 282–297. Springer-Verlag, 2011.

[61] Maher Mneimneh and Karem Sakallah. Sat-based sequential depth computation. In *Proceedings of the 2003 Asia and South Pacific Design Automation Conference*, ASP-DAC '03, pages 87–92, New York, NY, USA, 2003. ACM.

[62] Jeffrey D Orth, Ines Thiele, and Bernhard O Palsson. What is flux balance analysis? *Nat Biotech*, 28, 2010.

[63] Michael Pedersen and Andrew Phillips. Towards programming languages for genetic engineering of living cells. *Journal of The Royal Society Interface*, 6(Suppl 4):S437–S450, 2009.

[64] Andrew Phillips and Luca Cardelli. A programming language for composable dna circuits. *Journal of The Royal Society Interface*, 6(Suppl 4):S419–S436, 2009.

[65] Priscilla E. Purnick and Ron Weiss. The second wave of synthetic biology: from modules to systems. *Nature reviews. Molecular cell biology*, 10(6):410–422, June 2009.

[66] Lulu Qian and Erik Winfree. Scaling up digital circuit computation with dna strand displacement cascades. *Science*, 332(6034):1196–1201, 2011.

[67] Oliver Ray, Ken Whelan, and Ross King. Logic-based steady-state analysis and revision of metabolic networks with inhibition. In *Proceedings of the 2010 International Conference on Complex, Intelligent and Software Intensive Systems*, CISIS '10, pages 661–666, Washington, DC, USA, 2010. IEEE Computer Society.

[68] Areejit Samal and Sanjay Jain. The regulatory network of e. coli metabolism as a boolean dynamical system exhibits both homeostasis and flexibility of response. *BMC Systems Biology*, 2(1):21, 2008.

[69] Marc Schaub, Thomas Henzinger, and Jasmin Fisher. Qualitative networks: a symbolic approach to analyze biological signaling networks. *BMC Systems Biology*, 1(1):4, 2007.

[70] Y. Setty, I.R. Cohen, Y. Dor, and D. Harel. Four-Dimensional Realistic Modeling of Pancreatic Organogenesis. *Proceedings of the National Academy of Sciences*, 2008.

[71] Y. Setty, D. Dalfo, D.Z. Korta, E.J.A. Hubbard, and H. Kugler. A model of stem cell population dynamics: in-silico analysis and in-vivo validation. *Development*, 139(1):47–56, 2012.

[72] P.W. Sternberg and H.R. Horvitz. Pattern formation during vulval development in *C. elegans*. *Cell*, 44:761–772, 1986.

[73] Alvin Tamsir, Jeffrey J. Tabor, and Christopher A. Voigt. Robust multicellular computing using genetically encoded NOR gates and chemical /'wires/'. *Nature*, 469(7329):212–215, January 2011.

[74] Juilee Thakar, Mylisa Pilione, Girish Kirimanjeswara, Eric T Harvill, and Rka Albert. Modeling systems-level regulation of host immune responses. *PLoS Comput Biol*, 3(6):e109, 06 2007.

[75] René Thomas. Boolean formalization of genetic control circuits. *Journal of Theoretical Biology*, 42(3):563 – 585, 1973.

[76] Ashish Tiwari, Carolyn Talcott, Merrill Knapp, Patrick Lincoln, and Keith Laderoute. Analyzing pathways using SAT-based approaches. In *Proceedings of the 2nd international conference on Algebraic biology*, AB'07, pages 155–169, Berlin, Heidelberg, 2007. Springer-Verlag.

[77] C.M. Wintersteiger, Y. Hamadi, and L. de Moura. Efficiently solving quantified bit-vector formulas. In *Formal Methods in Computer-Aided Design (FMCAD), 2010*, pages 239–246, oct. 2010.

[78] B. Yordanov and C. Belta. Formal analysis of discrete-time piecewise affine systems. *IEEE Transactions on Automatic Control*, 55(12):2834 –2840, 2010.

[79] Ranran Zhang, Mithun Vinod Shah, Jun Yang, Susan B. Nyland, Xin Liu, Jong K. Yun, Rka Albert, and Thomas P. Loughran. Network model of survival signaling in large granular lymphocyte leukemia. *Proceedings of the National Academy of Sciences*, 105(42):16308–16313, 2008.

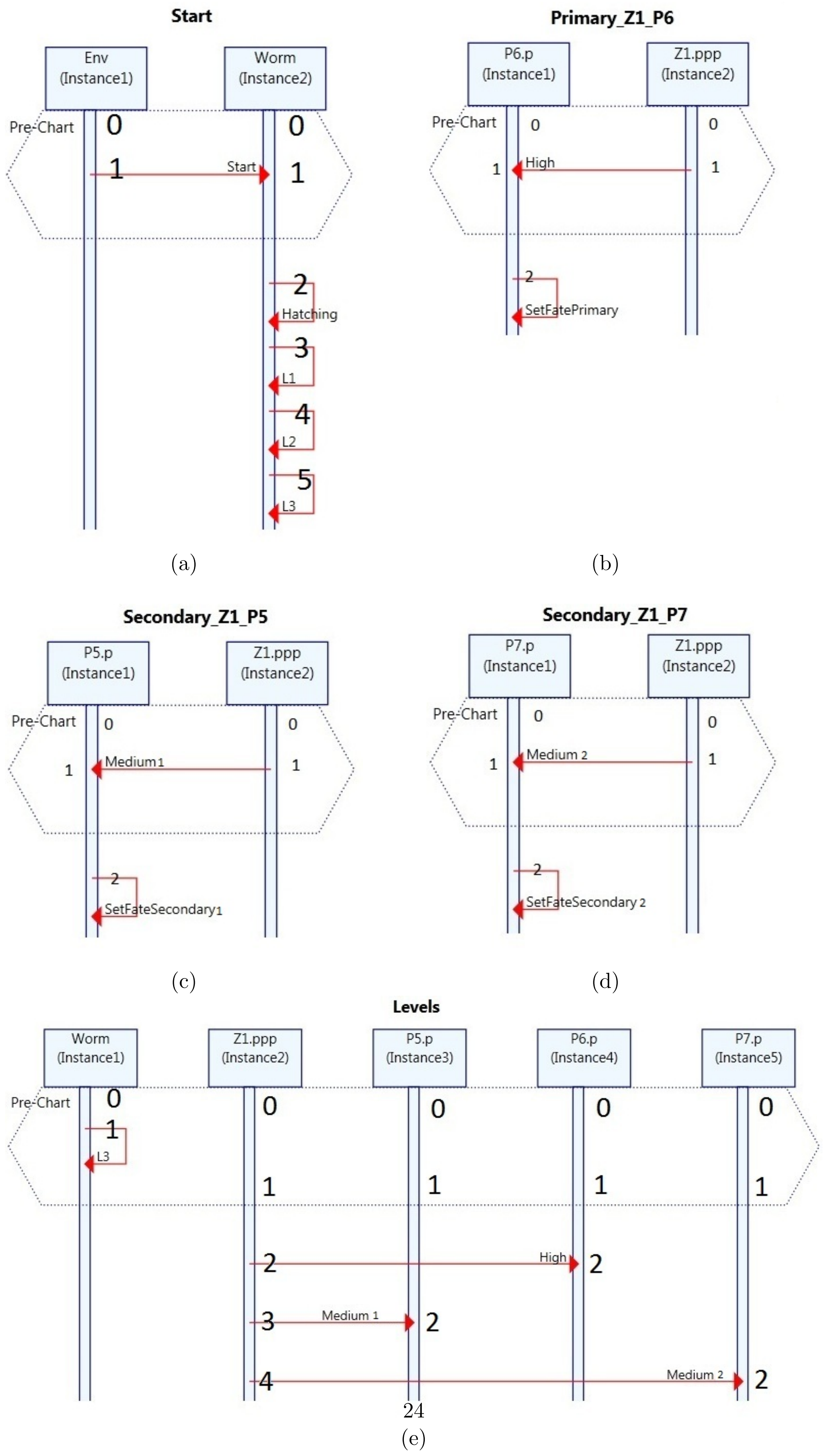

Figure 5: VPC system scenario-based model.

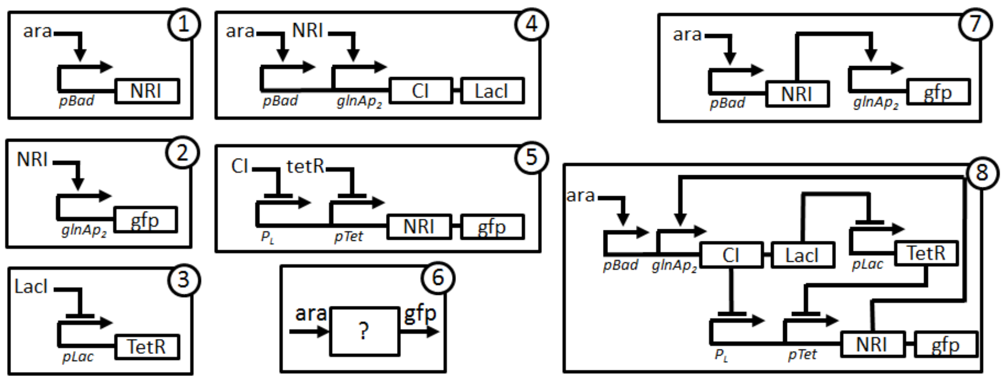

Figure 6: Several available devices ($D = \{1 \ldots 5\}$) are specified by their input and output species (*e.g.* arabinose (*ara*) and the protein *NRI* are the inputs of device 4, while *CI* and *LacI* are its outputs). We want to design a gene circuit with an arabinose input and green fluorescent protein (gfp) output (as in 6), subject to the constraints discussed in Sec. 8. Our procedure identifies solutions 7 (constructed from devices 1 and 2) and 8 (from 3,4, and 5), where all device inputs and outputs are matched.