SMT-Based Analysis of Biological Computation

Boyan Yordanov, Christoph M. Wintersteiger,
Youssef Hamadi, and Hillel Kugler

Microsoft Research, Cambridge UK
{yordanov,cwinter,youssefh,hkugler}@microsoft.com
http://research.microsoft.com/z3-4biology

Abstract. Synthetic biology focuses on the re-engineering of living organisms for useful purposes while DNA computing targets the construction of therapeutics and computational circuits directly from DNA strands. The complexity of biological systems is a major engineering challenge and their modeling relies on a number of diverse formalisms. Moreover, many applications are "mission-critical" (e.g. as recognized by NASA's Synthetic Biology Initiative) and require robustness which is difficult to obtain. The ability to formally specify desired behavior and perform automated computational analysis of system models can help address these challenges, but today there are no unifying scalable analysis frameworks capable of dealing with this complexity.

In this work, we study pertinent problems and modeling formalisms for DNA computing and synthetic biology and describe how they can be formalized and encoded to allow analysis using Satisfiability Modulo Theories (SMT). This work highlights biological engineering as a domain that can benefit extensively from the application of formal methods. It provides a step towards the use of such methods in computational design frameworks for biology and is part of a more general effort towards the formalization of biology and the study of biological computation.

1 Introduction

Significant progress in molecular and cellular biology and breakthroughs in experimental methods have raised hopes that the engineering of biological systems can serve for technological and medical applications, with a tremendous promise ranging from the sustainable production of biofuels and other materials [31] to the development of "smart" therapeutics [4]. Among the different approaches towards such molecular programming, DNA computation (the construction of computational circuits directly from DNA strands) and synthetic biology (the re-engineering of genetic networks within organisms) have emerged as promising directions with a number of experimental studies demonstrating their feasibility [22,23]. Recently, NASA has acknowledged the importance of this domain by creating the Synthetic Biology Initiative[1] [17] designed to *"harness biology in reliable, robust, engineered systems to support NASA's exploration and science missions, to improve life on Earth, and to help shape NASA's future"*.

[1] http://syntheticbiology.arc.nasa.gov/

G. Brat, N. Rungta, and A. Venet (Eds.): NFM 2013, LNCS 7871, pp. 78–92, 2013.

More generally, the engineering of biological systems can lead to a better understanding of biological computation (the computational processes within living organisms) with the goal of addressing some of the following questions: *What do cells compute? How do they perform such computation?* and *In what ways can the computation be modified or engineered?* which is the focus here.

Biological complexity presents a major engineering challenge, especially since many relevant applications can be considered as "mission-critical", while robustness is hard to engineer. Computational modeling currently focuses on using simulation to help address these challenges by allowing *in silico* experiments, however simulation alone is often not sufficient to uncover design flaws. For such applications, foundational computer-aided design technologies that allow desired behavior to be specified formally and analyzed automatically are needed. However, unifying analysis frameworks capable of dealing with the biological complexity and the diverse modeling formalisms used in the field are currently missing. Inspired by the study and engineering of other computational systems such as computer hardware and software, the application of formal methods has already attracted attention in the context of biology. In this work, we take a Satisfiability Modulo Theories (SMT)-based approach, utilizing transition systems as a uniform representation for biological models, and enabling efficient analysis for important properties of DNA computing and synthetic biology. Using theories richer than Boolean logic as in SMT offers a more natural framework by allowing higher-level problem descriptions and enhanced expressive power, provided that (efficient) automatic reasoning procedures are available. Such decision procedures are being developed actively [1] and are implemented in modern solvers such as Z3 [8] where, for certain applications, SMT-based methods outperform simpler theories [28], while their model-generation capabilities are important for the problems we consider. The richness of SMT accommodates analysis procedures to address a diverse set of biological questions and 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).

The main goals of this paper are (1) to study the pertinent modeling formalisms and problems for DNA computing and synthetic biology as representative biological engineering disciplines, formalize them, and describe how they can be encoded to allow analysis using SMT-based methods; (2) to exploit domain-specific knowledge in order to identify properties of these systems to help improve the scalability of analysis methods; (3) to present results from the application of these methods on challenging examples beyond what was possible using previous analysis approaches; and (4) to explore the utility of a general framework for analyzing biological computation.

2 Preliminaries and Notation

We denote a finite set as $S = \{s_0, \ldots, s_N\}$ where $|S| = N + 1$ is the number of elements in S. We use $S = \{(s_0, n_0), \ldots, (s_N, n_N)\}$ to denote a finite multiset

where each pair (s_i, n_i) denotes an element s_i and its multiplicity n_i with $n_i > 0$. Given a multiset S we use $s \in S$ when the multiplicity of s is not important and $S(s)$ to indicate the multiplicity of s when $s \in S$ and 0 otherwise. The union of multisets $(S \uplus S')$ as well as the multiplication of a multiset by a scalar (nS) are used according to their standard definitions.

3 DNA Computation

In the field of chemistry, mathematical theories such as mass-action kinetics have been developed to describe chemical reaction systems and predict their dynamical properties[12]. In molecular programming, the long term vision is to study the inverse problem where chemical and molecular systems are engineered with the goal of obtaining specific behavior of molecular events. The use of DNA as a chemical substrate has attracted attention, partially due to the availability of experimental techniques, as well as the predictability of chemical properties such as Watson-Crick pairing (the complementarity of the G-C and A-T base pairs which dictates the binding of DNA sequences). These properties have been exploited as early as in [3] where a strategy for computing a Hamiltonian path in a graph using DNA is described. DNA strand displacement (DSD) [25] is a particular DNA computation framework which, in principle, can be used to implement arbitrary computational procedures [14] and allows the use of DNA as a universal substrate for chemical reaction networks [27]. The feasibility of experimentally constructing large DNA computing circuits has been demonstrated recently [23]. Even so, the manual engineering of DNA circuits is challenging due to the parallel interactions of a large number of individual DNA species. To address these challenges, tools enabling the computational design and simulation of complex DNA circuits have been developed [16]. Here, we present an SMT-based approach for the analysis of these systems.

3.1 DNA Strand Displacement (DSD) Circuits

In a *DSD circuit*, a network of chemical reactions is constructed from *DNA species* (see Fig. 1), designed to interact according to DNA base-pairing rules. The DSD language [20] formalizes the notion of DNA species and the possible reactions between them. Briefly, a DNA species consists of a number of *strands* (individual DNA sequences)[2]. For example, in Fig. 1-A species s_0 consists of the single strand $\hat{s}_0$ while species s_3 consists of strands $\hat{s}_1$, $\hat{s}_2$, and $\hat{s}_3$ (all strands are listed in Fig. 1-D).

We are interested in studying the dynamics of a DSD circuit with single-molecule resolution by tracking how the amounts of species change as reactions take place, currently abstracting from the exact reaction kinetics (see Sec. 6 for additional discussion). A *state* of the system therefore describes the amount of molecules from each species present (Fig. 1-B). The initial state defined as part

[2] In the DSD language, strands are further subdivided into *domains* (*e.g.* t and $x0$ in strand $\hat{s}_0$ in Fig. 1-D) but for the current presentation this structure is not important.

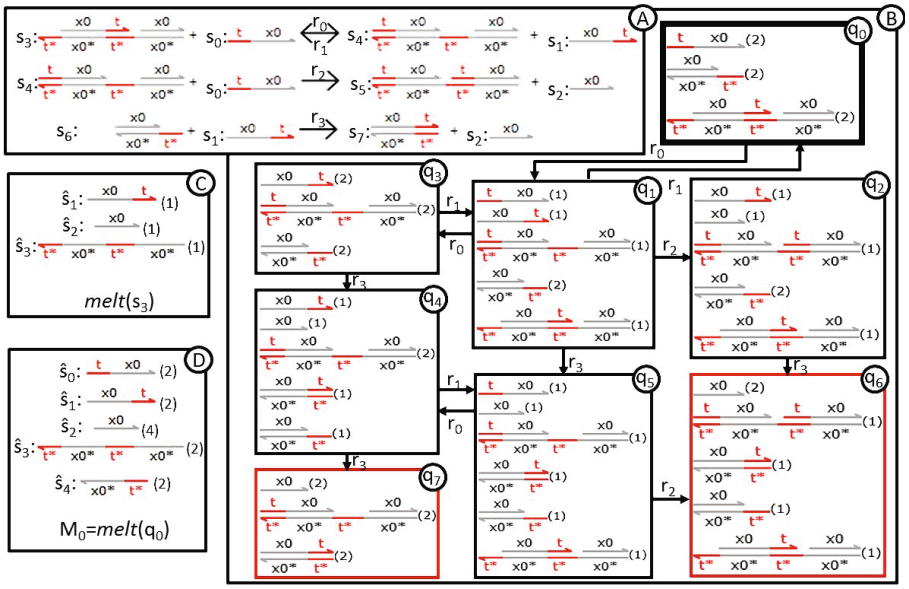

Fig. 1. A DSD circuit consisting of eight DNA species ($S = \{s_0, \ldots, s_7\}$) and four reactions ($R = \{r_0, \ldots, r_3\}$) represented graphically in (A). The state space of the system is shown in (B) where the multiplicity for each species is given in parenthesis and no further reactions are possible in the highlighted states (q_6 and q_7). The melting of a species and all species from a state (as discussed in Sec. 1.2) is illustrated in (C) and (D), respectively. Each strand from (D) represents a single DNA sequence.

of a DSD program (q_0 in Fig. 1-B), together with the rules of the DSD language, allows the automatic generation of possible reactions and species in the system [16]. We treat a DSD circuit as the pair $(\mathcal{S}, \mathcal{R})$ where $\mathcal{S}$ is a set of species and $\mathcal{R}$ is a set of reactions[3]. A reaction $r \in \mathcal{R}$ is a pair of multisets $r = (R_r, P_r)$ describing the reactants and products of r, where for $(s, n) \in R_r$ (resp. $(s, n) \in P_r$), $s \in \mathcal{S}$ is a species and n is the stoichiometry indicating how many molecules of s are consumed (resp. produced) through reaction r. To formalize the behavior of a DSD circuit, we construct the transition system $\mathcal{T} = (Q, q_0, T)$ where Q is the set of states, $q \in Q$ is a multiset of species ($q(s)$ indicates how many molecules of s are available in q), $q_0 \in Q$ is the initial state[4], and $T \subseteq Q \times Q$ is the transition relation. Reaction r is enabled in q, if there are enough molecules of its reactants

$$enabled(r, q) \leftrightarrow \bigwedge_{s \in \mathcal{S}} q(s) \geq R_r(s) \tag{1}$$

The transition relation T is defined as

$$T(q, q') \leftrightarrow \bigvee_{r \in \mathcal{R}} [enabled(r, q) \wedge \bigwedge_{s \in S} q'(s) = q(s) - R_r(s) + P_r(s)].$$

[3] Reversible reactions are treated as two non-reversible ones (*e.g.* r_0, r_1 in Fig. 1-A).
[4] We assume that the system is initialized in a single state (which is the case for DSD circuits we consider in this paper), although the methods can be generalized.

The definition of T aims to capture the firing of a single enabled reaction per transition. Still, in some special cases multiple reactions can fire in a time step (*e.g.* A $\rightarrow$ C and A + B $\rightarrow$ C + B) but this does not affect the structure of $\mathcal{T}$ and its subsequent analysis.

We assume that the set of species is finite and can be generated *a priory* (see discussion in Sec. 6). To enable the SMT-based analysis of DSD circuits, we represent the set of states of a transition system $\mathcal{T}$ using an integer encoding where $Q \subseteq \mathbb{N}^{|\mathcal{S}|}$ and the transition relation as a function $T : \mathbb{N}^{|\mathcal{S}|} \times \mathbb{N}^{|\mathcal{S}|} \rightarrow \mathbb{B}$. In Sec. 3.2 we prove that for a subset of DSD models the number of molecules of each species cannot exceed some upper bound N which can be computed from the species' structures[5]- this allows a finite representation of $\mathcal{T}$ and can help our analysis. Finite transition systems can be encoded naturally as logical formulas [5]. As an alternative to the integer representation, we encode the amount of species $s \in \mathcal{S}$ as a bit-vector of size $\lceil lg(N+1) \rceil$, leading to $Q \subseteq \mathbb{B}^{\hat{N}}$, $\hat{N} = |\mathcal{S}| \lceil lg(N+1) \rceil$ where $q(s) \in \mathbb{B}^{\lceil lg(N+1) \rceil}$ amounts to a bit-vector extraction and the transition relation is encoded as a function $T : \mathbb{B}^{\hat{N}} \times \mathbb{B}^{\hat{N}} \rightarrow \mathbb{B}$. Note that, although $\mathcal{T}$ is finite when species bounds are available, an explicit state-space representation of Q is often unfeasible to compute for realistic DSD circuits.

3.2 Constraints Generation

Naturally occurring chemical reaction networks are often subjected to constraints such as mass-conservation. In this section, we show that the known structure of species in a DSD circuit allows us to directly compute such constraints, which we exploit in our analysis. Intuitively, the individual strands from which all species in the system are composed are preserved and their total amounts remain unchanged. In the following, we exploit this *conservation of strands* property.

Let $\hat{s}$ denote a single strand where it is possible that $\hat{s} \notin \mathcal{S}$. Given a species $s \in \mathcal{S}$, we use the function $melt(s)$ to compute the multiset of strands that s is composed of (Fig. 1-C). The application of *melt* can be thought of as the "melting" of a species by increasing the temperature to dissociate all individual DNA strands. The function $melt()$ can be extended to operate on a multiset of species (Fig. 1-D), such as a state $q \in Q$

$$melt(q) \triangleq \biguplus_{s \in \mathcal{S}} q(s) melt(s) \tag{2}$$

Proposition 1. *The conservation of strands allows us to restrict the set of states reachable in $\mathcal{T}$ to a subset $\hat{Q} \subseteq Q$, for which the strands multiset is an invariant*

$$\forall q, q' \in \hat{Q}, melt(q) = melt(q') \tag{3}$$

Corollary 1. *The invariant multiset M_0 can be computed from the initial state*

$$\forall q \in \hat{Q}, melt(q) = M_0 \text{ where } M_0 = melt(q_0) \tag{4}$$

[5] As an additional optimization, separate bounds for each species can be computed.

The following constraints account for the conservation of strands

$$\forall q \in \hat{Q}, \bigwedge_{\hat{s} \in \hat{\mathcal{S}}} \left[\sum_{s \in S} q(s) M_s(\hat{s}) = M_0(\hat{s}) \right] \tag{5}$$

where $M_s = \mathit{melt}(s)$ denotes the multiset of strands for species s, $M_s(\hat{s})$ denotes the multiplicity of strand $\hat{s}$ in the composition of species s, and $\hat{\mathcal{S}} = \{\hat{s} \mid \exists s \in \mathcal{S}, \hat{s} \in \mathit{melt}(s)\}$ denotes the set of all individual strands in the system. In practice, Eqn. (5) is translated into constraints that might be challenging to solve. However, they can be simplified to obtain upper bounds on the multiplicities of individual species by constructing $\hat{Q}' \subseteq Q$, where in general $\hat{Q} \subseteq \hat{Q}'$ (*i.e.* $\hat{Q}'$ is an over-approximation of the states satisfying the conservation of strands invariant). Let $N_s = \min\{\lfloor M_0(\hat{s})/M_s(\hat{s}) \rfloor \mid \hat{s} \in M_s\}$ denote the maximal number of molecules of species s as restricted by its least abundant strand. Then

$$\forall q \in \hat{Q}', \bigwedge_{s \in \mathcal{S}} q(s) \leq N_s \tag{6}$$

We encode the constraints from Eqns. (5) and (6) using functions $\mathit{invariant} : Q \to \mathbb{B}$ where, for a state $q \in Q$, $\mathit{invariant}(q)$ iff $q \in \hat{Q}$, and $\mathit{bounds} : Q \to \mathbb{B}$ where $\mathit{bounds}(q)$ iff $q \in \hat{Q}'$ (the exact definition of these functions depends on the encoding of Q). In the following section, we will use these functions as constraints that will allow us to study the existence of states with certain properties[6]. The upper bounds from Eqn. (6) can also serve to determine the required bit-vector size for the encoding from Sec. 3.1 (*i.e.* $N = \max\{N_s \mid s \in \mathcal{S}\}$), while using the individual species bounds can lead to smaller encodings.

Example 1. For the DSD circuit from Fig. 1 the following constraints were generated: $s_0 + s_4 + 2s_5 = 2$, $s_2 + s_3 + s_4 + s_6 = 4$, $s_3 + s_4 + s_5 = 2$, $s_1 + s_3 + s_7 = 2$, and $s_6 + s_7 = 2$. From these, the following species bounds were obtained: $s_0 \leq 2$, $s_1 \leq 2$, $s_2 \leq 4$, $s_3 \leq 2$, $s_4 \leq 2$, $s_5 \leq 1$, $s_6 \leq 2$, and $s_7 \leq 2$.

3.3 Analysis of DNA Computation

To illustrate our method and discuss the formalization of properties relevant to DNA computation, we study a family of *transducer* circuits. Here, a transducer 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 through a set of chemical reactions [15]. A number of transducers can be connected *in series* (where the output of one circuit is the input for the next), which allows us to study systems of different size but with similar behavior. For these circuits, computation terminates when a state is reached where no further reactions are possible (this is also the case for the example from Fig. 1). As an additional requirement, certain reactive species denoted by $\mathcal{S}_r \subseteq \mathcal{S}$ must be

[6] For the bit-vector state encoding described in Sec. 3.1, the preclusion of overflows must be included as part of the constraints from Eqn. (5).

fully consumed throughout the computation but for some system designs this is not always the case [15]. We distinguish between "good" and "bad" termination states depending on the presence of reactive species and express the property of interest using standard temporal logic operators [21] as

$$AG(\neg bad) \wedge EF(good).$$

Formally, for a state $q \in Q$, we define

$$good(q) \leftrightarrow \bigwedge_{r \in \mathcal{R}} \neg enabled(r, q) \wedge \bigwedge_{s \in \mathcal{S}_r} s \notin q$$
$$bad(q) \leftrightarrow \bigwedge_{r \in \mathcal{R}} \neg enabled(r, q) \wedge \bigvee_{s \in \mathcal{S}_r} s \in q.$$

Using the constraints derived in Sec. 3.2, the feasibility of "good" and "bad" termination states can be analyzed. We search for a state q_g (resp. q_b) where

$$good(q_g) \wedge invariant(q_g) \tag{7}$$
$$bad(q_b) \wedge invariant(q_b) \tag{8}$$

When an unsatisfiable result is obtained for the formula from Eqn. (7) (resp. Eqn. (8)), the existence of a "good" (resp. "bad") state can be ruled out, and otherwise, a specific termination state q_g (resp. q_b) can be extracted from the model generated by the SMT solver. The constraints derived in Sec. 3.2 over-approximate the reachable states of a DSD circuit, which can only allow us to show that no reachable states with certain properties (*e.g.* "good" or "bad" states) exists. Identifying states q_g or q_b through this procedure, on the other hand, does not guarantee their reachability. To complete the analysis, we test the reachability of "good" and "bad" states using Bounded Model Checking (BMC) [5]. A "bad" state is reachable through K reactions or less if a trace $q_0, \ldots, q_K$ can be identified where q_0 is the initial state and

$$\bigvee_{k=0}^{K} bad(q_k) \wedge \bigwedge_{k=0}^{K-1} [T(q_k, q_{k+1}) \vee bad(q_k)] \tag{9}$$

while a similar procedure is used to search for reachable "good" states. If (9) is 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.

Besides increasing the number of transducers, system complexity can also be controlled by including multiple copies of the circuit [15], which amounts to changing the number of molecules available in the initial state (*e.g.* $q_0' = mq_0$ for a system with m copies), while the set of species and reactions remain the same. This can make analysis more challenging as the length of computation traces increases. Once a reachable "bad" state q_b is identified in a system, we show that such a state is also reachable for multiple copy systems by proving that no reactions are enabled in state $q' = mq_b$. State q' can be reached in a multi-copy system if each sub-system was to independently reach state q_b.

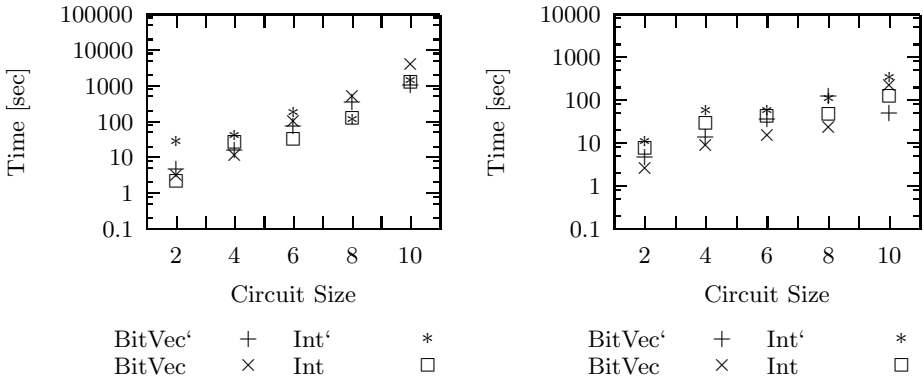

Fig. 2. Computation times for the identification of traces of lengths up to $K = 100$ in the flawed transducer circuits such that a "good" state (left panel) or a "bad" state (right panel) is reached (note the difference in scales). BitVec‘ and BitVec (resp. Int‘ and Int) indicate a bit-vector (resp. integer) encoding with or without the additional constraints from Sec. 3.2 asserted for each state $q_0, \ldots, q_{K-1}$ in (9).

We applied the procedure described in this section to DSD circuits consisting of between $n = 2, \ldots, 10$ transducers in series where all transducers were based on one of two different designs (a flawed and a corrected one). These circuits were found to include $|\mathcal{S}| = 14n + 4$ species and $|\mathcal{R}| = 8n$ reactions and, when the bit-vector encoding was used, a state was encoded as a bit-vector of size 64 (resp. 70) for the flawed (resp. corrected) circuit of size $n = 2$ and 576 (resp. 342) for $n = 10$. For the flawed system design, both "good" and "bad" states were identified using Eqns. (7) and (8) while for the corrected design only "good" termination states were possible. For each of the investigated circuits (encoded either using integers or bit-vectors) computation[7] required under 1 sec.

To confirm the reachability of states we searched for traces with depth up to $K = 100$, which was sufficient to identify computation traces leading to both "good" and "bad" states for flawed transducer circuits of different size (Fig. 2) and "good" states for the corrected one (Fig. 3-left)(the existence of "bad" states for these circuits was already ruled out). For the corrected transducers, we show that the additional constraints from Sec. 3.2 allow us to rule out the possibility of "bad" states, even for systems with many copies of the circuit (Fig. 3-right).

4 Synthetic Gene Circuits

In the field of synthetic biology, engineering principles are applied to redesign genetic networks with the goal of constructing biological systems with specific

[7] All computational results were obtained using the Z3 (version 4.1) theorem prover [8] on 2.5 Ghz Intel L5420 CPU machines with a 2GB memory limit per benchmark.

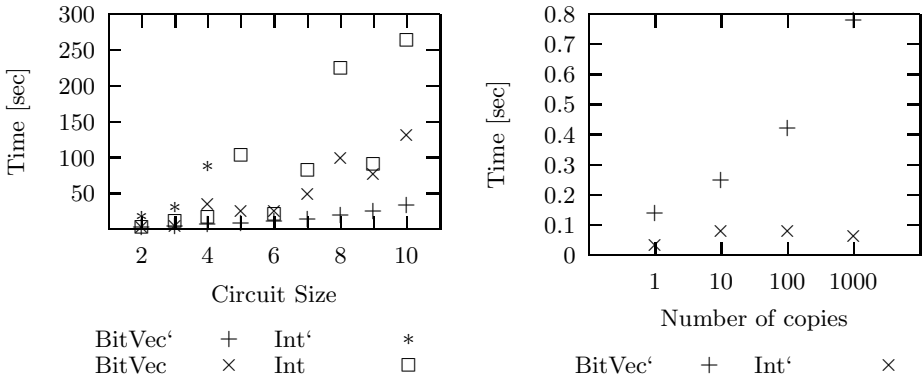

Fig. 3. Computation times for the identification of traces of lengths up to $K = 100$ in the corrected transducer circuits (left panel) and the verification of multiple copies of a circuit with ten transducers based on the corrected design (right panel). BitVec‘ and BitVec (resp. Int‘ and Int) indicate a bit-vector (resp. integer) encoding with or without the additional constraints from Sec. 3.2. For the integer encoding with the additional constraints, the memory limit was reached for circuits of size five and up during the identification of traces (left panel).

behavior (see [22] for a review). The construction of biological *devices* (relatively small gene networks which can serve as basic building-blocks) has been pursued initially and tools and programming languages to support this process have been developed (*e.g.* [19]). The construction of larger-scale systems from devices presents a challenging design problem [22], where chemical species serving as "wires" must be matched to ensure proper function and other constraints must also be satisfied (*e.g.* if the same species is an output of two separate devices in a circuit, cross-talk might occur), while in addition, specific system behavior must be obtained. The development of computational tools enabling the automated design of systems from expressive specification of the desired behavior and capable of handling the complexities of the problem can address these challenges. In the following, we formalize the constraints specific to the synthesis problem of designing a gene network with certain behavior from a library of devices and propose an SMT-based solution.

A device d is a tuple $d = (I_d, S_d, F_d)$ where I_d and S_d are finite sets of input and internally produced (output) species such that $I_d \cap S_d = \emptyset$ and $F_d = \{f_d^s \mid s \in S_d\}$ is a set of update functions (f_d^s is the update function for species s). We capture the dynamics of a device as a Boolean network - a popular formalism for modeling interaction networks [7]. In a Boolean network each species is described as available or not, thus its exact concentration (number of molecules) is abstracted (unlike the DSD formalism we described in Sec. 3). We treat a device d as a transition system $\mathcal{T}_d = (Q_d, Q_{d0}, T_d)$ where $q \in Q_d$ captures which species are available in the system (in the following, we use $q(s) \in \mathbb{B}$ as a Boolean, indicating whether species s is available in state q) and $Q_{d0} = Q_d$ (*i.e.* all states are initial). The dynamics of the system are given by the functions

Table 1. Additional constraints for constructing systems from gene network devices

Constraints	Description
$\bigwedge_{s\in\mathcal{S}} \bigwedge_{d,d'\in D_s, d\neq d'} \neg(D(d) \wedge D(d'))$	To prevent cross-talk, two devices producing the same species are never selected at the same time.
$\bigwedge_{s\in I} \bigvee_{d\in D_s} D(d)$	All species specified as input serve as inputs to a selected device.
$\bigwedge_{s\in O} \bigvee_{d\in D^s} D(d)$	All species specified as output are produced by a selected device.
$\bigwedge_{d\in\mathcal{D}} \left(D(d) \rightarrow \bigwedge_{s\in S_d\setminus O} \bigvee_{d'\in D_s} D(d')\right)$	To prevent the production of species that do not serve any function, all species produced by a selected device are outputs of the circuit or serve as input to another selected device.
$\bigwedge_{d\in\mathcal{D}} \left(D(d) \rightarrow \bigwedge_{s\in I_d\setminus I} \bigvee_{d'\in D^s} D(d')\right)$	All species serving as inputs to a selected device are inputs of the circuit or are produced by another selected device in order to ensure that all device inputs are part of the system.

$f_d^s : Q_d \rightarrow \mathbb{B}$ where, for states $q, q' \in Q_d$, the *synchronous* transition relation (where all species are updated at each time step) is defined as

$$T(q,q') \leftrightarrow \left(\bigwedge_{s\in S_d} q'(s) = f_d^s(q) \right) \tag{10}$$

Note that $\mathcal{T}_d$ is finite and non-deterministic: while each species $s \in S_d$ is updated deterministically, there are no restrictions on the dynamics of species from I_d.

Given a set of devices $\mathcal{D} = \{d_0, \ldots, d_n\}$ we define the set of species $\mathcal{S} = \bigcup_{d\in\mathcal{D}}(I_d \cup S_d)$. A specification of some required system behavior is given over the dynamics of a set of input and output species denoted by $I \subseteq \mathcal{S}$ and $O \subseteq \mathcal{S}$ where $I \cap O = \emptyset$ and $I \cap (\bigcup_{d\in\mathcal{D}} S_d) = \emptyset$. Our goal is to select a subset of devices $D \subseteq \mathcal{D}$ where $D(d) \in \mathbb{B}$ indicates whether device d is used as part of the system.

Let $D_s = \{d \in \mathcal{D} \mid s \in S_d\}$ denote the set of devices producing species s and $D^s = \{d \in \mathcal{D} \mid s \in I_d\}$ denote the set of devices using s as an input. We construct the transition system $\mathcal{T} = (Q, Q_0, T)$ where $q(s) \in \mathbb{B}$ indicates the availability of species $s \in \mathcal{S}$ and $Q_0 = Q$. The following constraints are asserted for all valid system states

$$\forall q \in Q, \bigwedge_{s\in\mathcal{S}\setminus I} \left(\left(\bigwedge_{d\in D_s} \neg D(d) \right) \rightarrow \neg q(s) \right) \tag{11}$$

In other words, a species s that is not produced by any device is never available in valid states of the system, unless $s \in I$. To prevent cross-talk between devices and obtain a smaller solution (*e.g.* where devices that produce unnecessary species are never included), we impose the additional constraints described in Table 1

(note that only the cross-talk constraint is required to avoid contradictions in the following definition). The transition relation of $\mathcal{T}$ is defined as

$$T(q, q') \leftrightarrow \left(\bigwedge_{s \in \mathcal{S}} q'(s) = f_d^s(q_d) \right) \tag{12}$$

where q_d denotes the part of state q relevant for device d (*i.e.* describing species from $I_d \cup S_d$). With the exception of the inputs I, the system is deterministic.

For this problem, a bit-vector encoding is appropriate due to the Boolean structure of the system. For an individual device d we have $Q_d = \mathbb{B}^{|I_d|+|S_d|}$ where each $f_d^s \in F_d$ is a function $f_d^s : \mathbb{B}^{|I_d|+|S_d|} \to \mathbb{B}$. For the overall system, we have $Q = \mathbb{B}^{|\mathcal{S}|}$ where T is a function $\mathbb{B}^{|\mathcal{S}|} \times \mathbb{B}^{|\mathcal{S}|} \to \mathbb{B}$. We use a bit-vector $D \in \mathbb{B}^{|\mathcal{D}|}$ to encode the set of selected devices. Given a device d, q_d can be encoded using appropriate bit-vector extraction and concatenation to select the species from $\mathcal{S} \cap (I_d \cup S_d)$, which allows the application of functions from F_d.

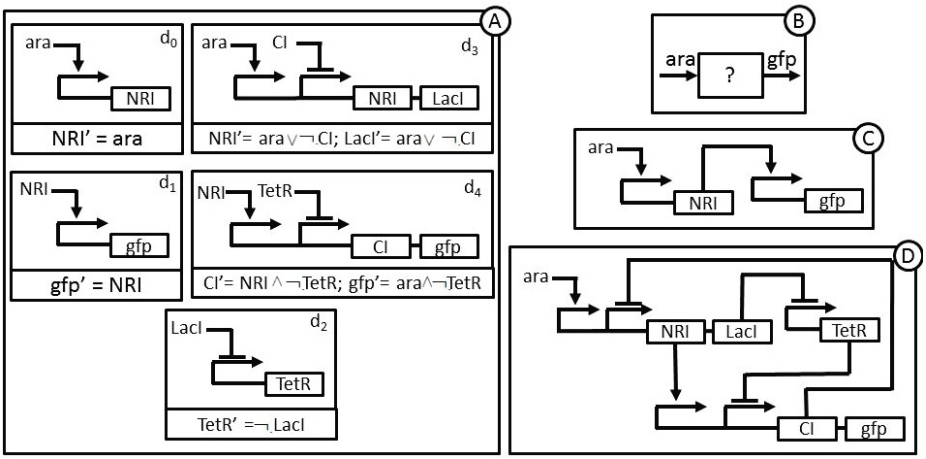

Fig. 4. A library of devices ($\mathcal{D} = \{d_0, \ldots, d_4\}$) is represented graphically in (A). Individual devices are specified by their input and output species (*e.g.* arabinose (*ara*) and the protein *CI* are the inputs of device d_3, while *NRI* and *LacI* are its outputs), together with a Boolean update function for each species. For example, species LacI is available in the next state (indicated by LacI') only if arabinose is available in the current state and the repressor CI is not (positive and negative regulation is represented by pointed or flat arrows). We seek a circuit with an arabinose input and green fluorescent protein (gfp) output (B) which is capable of oscillations and stabilization and satisfies the constraints discussed in Sec. 4. Our procedure identifies the solutions shown in (C, where $D = \{d_0, d_1\}$) and (D, where $D = \{d_2, d_3, d_4\}$) but only the solution from (D) has the desired dynamic behavior.

The characterization of device behavior and the construction of device libraries is currently an ongoing effort in synthetic biology. Therefore, we consider a hypothetical device library (Fig. 4-A) constructed from frequently-used components [22] to demonstrate the proposed approach. The Boolean update rules

we use as an illustration abstract the detailed gene regulation behavior which has been observed and engineered experimentally. Our goal in this example is to design a circuit with the input/output characteristics shown in Fig. 4-B. In addition, we require that it is possible for the output gfp to oscillate for one value of the input *ara* and stabilize for the other. Two possible solutions satisfying all constraints from Table 1 were identified (Fig. 4-C,D). However, this alone does not guarantee that their dynamical properties are consistent with the required behavior. We specify two paths of the system $q_0, \ldots q_K$ and $q'_0, \ldots, q'_K$ with the properties that $\bigvee_{i=1}^{K} q_i(\text{ara}) = q_0(\text{ara})$, $\bigvee_{i=1}^{K} q'_i(\text{ara}) = q'_0(\text{ara})$ and $q_0(\text{ara}) \neq q'_0(\text{ara})$ (*i.e.* a different, constant input signal is applied in each case), $q_{K-1} = q_K$ (in the first case, the circuit stabilizes) and $\bigvee_{i=0}^{K-1}(q'_i = q'_K \wedge \bigvee_{j=i+1}^{K-1} q'_j(\text{gfp}) \neq q'_K(\text{gfp}))$ (in the second case, the circuits oscillates between multiple states where the value of gfp changes). These additional constraints eliminate the device from Fig.4-C as a possible solution, while the device from Fig.4-D is still identified as a candidate (overall, the solution was found in under 1 sec). Besides specifying behavior that the system must be capable of (*i.e.* the existence of trajectories with certain properties), specification of properties that all system trajectories must satisfy are also supported in our approach through the use of quantifiers.

5 Related Work

The application of formal methods to biology has already received attention (*e.g.* [6], among others) but here we focus specifically on biological engineering applications where formal specifications and analysis can supplement computational modeling and simulation to enable the computer-aided design of larger, more reliable systems. DNA circuits have been studied using stochastic simulation, or more recently, using probabilistic model checking [15], which also allows properties regarding the time required for computation or the probability of failure to be expressed. In synthetic biology, computational design platforms exists to target the construction of devices [29,19], while formal specifications have also been considered [30]. Here we focus on the problem of combining devices into systems, while satisfying additional design constraints, including desired system behavior. This problem is related to the synthesis of software programs from components [10] - here we formalize features of the biological design process and address this problem using SMT-based methods.

The Petri-net formalism [11] can naturally describe some properties of the DNA circuits from Sec. 3. The application of formal methods to Petri-nets has been studied extensively, and in the future, relevant analysis procedures can be adapted to the problems we consider. More specifically, the computation of invariants for Petri-nets has been studied as an analysis strategy (*e.g.* in the context of biology [26]). This problem is related to the computation we describe in Sec. 3.2 but here we derive constraints directly from the known composition of DNA species, while for Petri-nets such information is not available and invariants are computed from the network structure (*e.g.* through its incidence matrix [11]). Since the structure of species also determines the possible reactions between them combining these approaches is an interesting future direction.

6 Discussion and Future Work

Our approach is general and suitable for the analysis of biological models beyond the applications discussed in this paper. For example, with the exception of the additional species properties we exploit in Sec. 3.2, DSD circuits can be viewed as general Chemical Reaction Networks (CRNs) and therefore such systems might be analyzed using the proposed methods. Furthermore, the Boolean networks we use in Sec. 4 are a popular modeling formalisms for biological interaction networks (such as gene regulation, and signaling networks) as studied in the field of systems biology. A number of realistic models have been constructed based on this formalism, including large-scale (approaching whole-cell) regulatory and metabolic reconstructions [24] where our methods can help address challenging analysis problems. Besides providing analysis capabilities within tools such as Visual DSD [16], the discussed methods are also available as an online tool at [2].

The expressivity, scalability and extensibility of SMT, together with its model generation capabilities, which served for the identification of (counter)examples in this work but can also allow synthesis applications in the future, were the major consideration during the development of our methods. Our choice of bit-vectors as a specific theory of interest was motivated by the availability of recently developed efficient decision procedures, which allow the use of quantifiers [28]. The investigation of the integer and bit-vector representations we propose on a larger set of benchmarks is a direction of future work.

In Sec. 3 we assume that all species and reactions of a DSD circuit can be generated *a priory* (*e.g.* using Visual DSD [16]), which is often the case for circuits of practical interest with some notable exceptions [14], which might still be approached using SMT (*e.g.* through the use of recursive datatypes [18]). When a sufficient number of molecules is present in chemical and biological systems, species concentrations can be described as continuous values (*e.g.* using (non-linear) ODEs) [7], which is also a common description of the synthetic gene networks studied in Sec. 4. Such systems, as well as other infinite-state, continuous and hybrid models used in biology, can be encoded directly into SMT and analyzed using recently developed decision procedures [13]. As an alternative, (conservative) finite transition system abstractions can be constructed (*e.g.* as in [30]) to enable the SMT-based analysis of such systems.

In the modeling of biological systems, capturing individual molecules numbers (as in Sec. 3) can provide detailed and biologically accurate system descriptions, which are often difficult to analyze. Constructing (finite) transition system representations as in Sec. 3 allows us to express important properties (*e.g.* reachability) with such level of detail, which is important for applications such as DNA computing. Currently, we ignore all probabilistic aspects of these systems (which arise, for example, when reaction rates are considered) but SMT reasoning procedures for probabilistic systems (*e.g.* [9]) can help address some of the current limitations and extend our approach to other model classes (*e.g.* probabilistic Boolean networks). For challenging areas such as probabilistic SMT, it seems natural to explore biological applications for motivation and as a source of benchmarks that can help drive the development of novel methods.

7 Conclusion

As a step towards the development of an SMT-based analysis framework for studying biological computation that is scalable and supports a wide set of models and specifications, in this paper we focus on problems related to the engineering of biological systems, as part of the emerging fields of DNA computation and synthetic biology. We show that for a number of applications in these domains, transition systems capture important behavior and can be analyzed together with relevant specifications and additional constraints through SMT-based methods in an efficient manner, going beyond what was possible using other techniques. Our approach is general and is currently being applied to other biological models and formalisms. This work highlights biological engineering as a domain that can benefit extensively from the application of formal methods, while the biological complexity can also motivate the development of novel analysis methods.

References

1. Satisfiability modulo theories competition, `http://www.smtcomp.org/2012/`
2. `Z34bio` at `rise4fun` - software engineering tools from MSR (2012), `http://rise4fun.com/z34biology`
3. Adleman, L.: Molecular computation of solutions to combinatorial problems. Science 266(5187), 1021–1024 (1994)
4. Benenson, Y., Gil, B., Ben-Dor, U., Adar, R., Shapiro, E.: An autonomous molecular computer for logical control of gene expression. Nature 429(6990), 423–429 (2004)
5. Biere, A., Cimatti, A., Clarke, E., Zhu, Y.: Symbolic model checking without BDDs. In: Cleaveland, W.R. (ed.) TACAS 1999. LNCS, vol. 1579, pp. 193–207. Springer, Heidelberg (1999)
6. Chabrier-Rivier, N., Chiaverini, M., Danos, V., Fages, F., Schächter, V.: Modeling and querying biomolecular interaction networks. Theoretical Computer Science 325(1), 25–44 (2004)
7. de Jong, H.: Modeling and simulation of genetic regulatory systems: A literature review. Journal of Computational Biology 9(1), 67–103 (2002)
8. de Moura, L., Bjørner, N.: Z3: An efficient SMT solver. In: Ramakrishnan, C.R., Rehof, J. (eds.) TACAS 2008. LNCS, vol. 4963, pp. 337–340. Springer, Heidelberg (2008)
9. Fränzle, M., Hermanns, H., Teige, T.: Stochastic satisfiability modulo theory: A novel technique for the analysis of probabilistic hybrid systems. In: Egerstedt, M., Mishra, B. (eds.) HSCC 2008. LNCS, vol. 4981, pp. 172–186. Springer, Heidelberg (2008)
10. Gulwani, S., Jha, S., Tiwari, A., Venkatesan, R.: Synthesis of loop-free programs. SIGPLAN Not. 46, 62–73 (2011)
11. Heiner, M., Gilbert, D., Donaldson, R.: Petri Nets for Systems and Synthetic Biology. In: Bernardo, M., Degano, P., Zavattaro, G. (eds.) SFM 2008. LNCS, vol. 5016, pp. 215–264. Springer, Heidelberg (2008)
12. Horn, F., Jackson, R.: General mass action kinetics. Archive for Rational Mechanics and Analysis 47(2) (1972)

13. Jovanović, D., de Moura, L.: Solving Non-linear Arithmetic. In: Gramlich, B., Miller, D., Sattler, U. (eds.) IJCAR 2012. LNCS, vol. 7364, pp. 339–354. Springer, Heidelberg (2012)
14. Lakin, M.R., Phillips, A.: Modelling, simulating and verifying turing-powerful strand displacement systems. In: Cardelli, L., Shih, W. (eds.) DNA 17. LNCS, vol. 6937, pp. 130–144. Springer, Heidelberg (2011)
15. Lakin, M.R., Parker, D., Cardelli, L., Kwiatkowska, M., Phillips, A.: Design and analysis of DNA strand displacement devices using probabilistic model checking. Journal of the Royal Society, Interface 9(72), 1470–1485 (2012)
16. Lakin, M.R., Youssef, S., Polo, F., Emmott, S., Phillips, A.: Visual DSD: a design and analysis tool for DNA strand displacement systems. Bioinformatics 27(22), 3211–3213 (2011)
17. Langhoff, S., Rothschild, L., Cumbers, J., Paavola, C., Worden, P.: Workshop Report on What are the Potential Roles for Synthetic Biology in NASA's Mission? Technical report (2012)
18. Milicevic, A., Kugler, H.: Model checking using SMT and theory of lists. In: Bobaru, M., Havelund, K., Holzmann, G.J., Joshi, R. (eds.) NFM 2011. LNCS, vol. 6617, pp. 282–297. Springer, Heidelberg (2011)
19. Pedersen, M., Phillips, A.: Towards programming languages for genetic engineering of living cells. J. R. Soc. Interface 6(suppl. 4), S437–S450 (2009)
20. Phillips, A., Cardelli, L.: A programming language for composable DNA circuits. Journal of the Royal Society, Interface 6(suppl. 4), S419–S436 (2009)
21. Pnueli, A.: The temporal logic of programs. In: 18th Annual Symposium on Foundations of Computer Science (FOCS 1977), pp. 46–57. IEEE (1977)
22. Purnick, P.E., Weiss, R.: The second wave of synthetic biology: from modules to systems. Nature Reviews. Molecular Cell Biology 10(6) (2009)
23. Qian, L., Winfree, E.: Scaling up digital circuit computation with DNA strand displacement cascades. Science 332(6034), 1196–1201 (2011)
24. Samal, A., Jain, S.: 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)
25. Seelig, G., Soloveichik, D., Zhang, D.Y., Winfree, E.: Enzyme-free nucleic acid logic circuits. Science 314(5805), 1585–1588 (2006)
26. Soliman, S.: Finding minimal P/T-invariants as a CSP. In: Proceedings of the fourth Workshop on Constraint Based Methods for Bioinformatics WCB, vol. 8 (2008)
27. Soloveichik, D., Seelig, G., Winfree, E.: DNA as a universal substrate for chemical kinetics. Proceedings of the National Academy of Sciences of the United States of America 107(12), 5393–5398 (2010)
28. Wintersteiger, C., Hamadi, Y., de Moura, L.: Efficiently solving quantified bit-vector formulas. In: FMCAD, pp. 239–246 (2010)
29. Yaman, F., Bhatia, S., Adler, A., Densmore, D., Beal, J.: Automated Selection of Synthetic Biology Parts for Genetic Regulatory Networks. ACS Synthetic Biology 1(8), 332–344 (2012)
30. Yordanov, B., Belta, C.: A formal verification approach to the design of synthetic gene networks. In: IEEE Conference on Decision and Control and European Control Conference, pp. 4873–4878. IEEE (2011)
31. Zhang, F., Rodriguez, S., Keasling, J.D.: Metabolic engineering of microbial pathways for advanced biofuels production. Current Opinion in Biotechnology 22(6), 775–783 (2011)