A Property-Driven Methodology for Formal Analysis of Synthetic Biology Systems

Savas Konur and Marian Gheorghe

Abstract—This paper proposes a formal methodology to analyse bio-systems, in particular synthetic biology systems. An integrative analysis perspective combining different model checking approaches based on different property categories is provided. The methodology is applied to the synthetic *pulse generator* system and several verification experiments are carried out to demonstrate the use of our approach to formally analyse various aspects of synthetic biology systems.

Index Terms—formal analysis, verification, model checking, synthetic biology, synthetic pulse generator

1 Introduction

Synthetic biology is an emerging discipline, which provides an engineering perspective to biology. There have been tremendous research efforts over the last few years to develop computational models, engineering methods and tools supporting this discipline. Many software platforms have been introduced, including CAD design tools, DNA construction tools, part libraries, etc. [1] In particular, computational design tools are very useful in this respect, because they enable designing and analysing biological parts, devices and systems in a computer-based environment. Namely, they can mimic *in vivo* experiments as *in silico* experiments. This obviously provides a significant advantage to reduce the costs and duration of wet-lab experiments.

Various tools have been devised in this respect, which integrate different computational methods for analysing synthetic biology systems. Simulation is one of them, which has been predominantly used. Although simulation plays an important role to discover system behaviours, it can only reveal limited information.

Formal verification, a computational method permitting to analyse *all* system behaviours, is an alternative and complementary approach to simulation. This requires an abstract representation of the system as a formal model. The *exhaustive* analysis required by this method allows revealing more comprehensive information than simulation. *Model checking* is an algorithmic verification technique, where a formal representation of a system *property* is verified against a mathematical representation of a model. Various model checking approaches, and hence tools (e.g., NuSMV [2], Prism [3], Plasma [4], and MC2 [5]) have been devised to support automatic verification. Model checking has been applied to systems biology extensively. Recently, a survey paper [6] summarises the research in this area by presenting various bio-systems analysed by different model checkers.

The information inferred via the use of model checking depends on the types of properties querying the model. A model checking approach is utilised to verify certain property types, implying that while one approach supports a certain property type, another approach might not support it. This suggests the use of different approaches when formally analysing different property types – for instance *qualitative* vs *quantitative* behaviour [7]. However, most of the systems biology models have been analysed using a single model checking approach. This limits inferring more information about different aspects of a system.

As far as we know, the vast majority of the biomodels analysed using formal verification methods are within the scope of systems biology. There is very limited effort to apply model checking in the context of synthetic biology (e.g. [8]) or attempts the extend the application of this verification approach in systems biology to synthetic biology problems [7]. This might be due to the fact that synthetic biology requires an engineering approach and adequate tools in this respect.

Another drawback is that model checking is a very daunting and tedious task for non-experts, because it requires a very good understanding of the model checking philosophy and a reasonable familiarity with both modelling and property specification languages.

In this paper, to target these challenges, we will propose a formal approach to analyse bio-systems, in particular synthetic biology systems. This suggests an integrative perspective combining different model checking approaches based on different property types and the use of some natural language patterns to express various properties. This approach makes the use of the model checking methods very effective and easy to use, even for non-experts in for-

• *Department of Computer Science, University of Sheffield, UK*
E-mail: {s.konur,m.gheorghe}@sheffield.ac.uk

mal methods, which is coherent with the engineering viewpoint in synthetic biology [1].

Our Contribution

We can summarise the contribution of the paper as follows: (i) we provide some guidance on how verification can be applied to biological systems; (ii) we describe a platform to analyse several property categories by taking advantage of how they are supported by various model checking approaches; (iii) we devise automatic translations of a system model into the input formats of different model checkers, helping non-experts in avoiding using various modelling languages; (iv) we feature a novel property construction method, which enables an automated property construction process using natural language query patterns, making the property specification process very accessible to non-experts; and (v) we demonstrate our approach on a synthetic biology construct as a step towards the effective use of formal verification in synthetic biology systems.

The paper is organised as follows: Section 2 provides a brief overview of model checking. Section 3 discusses the methodology in detail. Section 4 describes a synthetic biology construct used as a case study. Section 5 addresses the application of the methodology to a pulse-generator case study and shows the corresponding verification experiments carried out. Section 6 draws conclusions and discusses future work.

2 Model Checking

Model checking is an algorithmic verification technique, which analyses the validity of a logically defined property using mathematical techniques. It is a complementary approach to simulation because, unlike simulation, it requires an *exhaustive exploration* of *all* system behaviours. The algorithms devised to explore models' state spaces are implemented in specific tools, called *model checkers*.

A model checker requires a mathematical representation of its input model written in a high-level modeling language and a property to be checked expressed as logical statements, e.g. *temporal logic formulas* [9]. The model checker then exhaustively analyses the property against the model, and returns a 'yes' or 'no' answer depending on the outcome of this analysis. In case of a failure in verifying the property, it returns a counter example to the user, which can then be used to trace back and correct any faulty segments in the model.

Model checking has been widely used in the verification of various systems, e.g. concurrent [10] and distributed systems [11], multi-agent systems [12], pervasive systems [13] and swarm robotics [14]. Recently, it has been also applied to analysis of various biological systems, e.g. ERK/MAPK pathway [15], FGF signalling pathway [16], cell cycle in eukaryotes [17], EGFR pathway [18], T-cell receptor signaling pathway [19], cell cycle control [20], and genetic Boolean gates [21], [22].

Classical model checking techniques are mainly used to infer qualitative information about the system behaviour. However, there are other dimensions/facets required to understand more complex system behaviours [23]. In synthetic biology, as well as the qualitative information, we also need to infer quantitative information to obtain more novel information about system properties.

Probabilistic model checking is a *quantitative* extension of classical model checking, which allows us to analyse the likelihood of an event to occur, such as the concentration of a species exceeding a threshold value, rather than just a 'yes' or 'no' answer. The quantitive approach also permits to draw graphs regarding the evolution of various species, which can be used to search for any trends in their behaviour or to find any anomalies.

Although model checking is very useful to derive vital information about the system's behaviour, it is known to be very resource demanding. The approach is not feasible when it is applied to very large models because a huge state space needs to be constructed and explored. This is especially the case for synthetic biology models because, as we will show later, these models are very large, containing compartments and components, and in most cases they stretch the capabilities of the model checking tools.

Statistical model checking is an alternative model checking technique which, unlike probabilistic model checking, does not require an exhaustive exploration of *all* behaviours. Instead, the method is based on constructing a finite set of system runs (i.e. simulation traces) and then calculating the correctness as an approximation using statistical techniques, such as Monte Carlo. Whereas, in the case of the probabilistic model checking, a precise verification result can be obtained as a result of numerical analysis, in statistical model checking we can only obtain an approximate result. The accuracy is calculated based on specified degree of *confidence*. However, the performance is significantly improved.

2.1 NuSMV

NuSMV [2] is one of the most popular tools, used to model check the correctness of finite state-based systems. NuSMV models are constructed using a high-level modelling language, called *SMV*, which allows a modular and hierarchical system description. NuSMV can verify *temporal logic* statements, expressed in Linear-time Temporal Logic (LTL) and Computation Tree Logic (CTL). For example, the property

Prop. 1. *"`GFP` is produced as a response to the signal `3OC12HSL`"*

is expressed in CTL as

$$\text{AG}\ (\texttt{3OC12HSL} > 0 \Rightarrow \text{EF}\ \texttt{GFP} > 0)$$

where A (E, resp.) is a *path* operator representing that for *all* (for *some*, resp.) executions of the system, and F (G, resp.) is a *temporal* operator representing the *eventually* (*always*, resp.) modality, meaning that a behaviour is *eventually* (*always*, resp.) true.

NuSMV implements *symbolic* methods to have a more compact representation of the state space and hence to reduce the computational resources required to perform the verification.

We note that SPIN [24] is another widely used model checking tool, developed to verify qualitative (temporal) properties. SPIN provides complete support for LTL, but cannot express CTL properties.

2.2 PRISM

The mostly widely used *probabilistic* model checking tool is PRISM [3]. The tool allows model checking of various probabilistic systems, such as discrete time Markov chains (DTMCs), Markov decision processes (MDPs) and continuous time Markov chains (CTMCs). Models are constructed using a state-based high-level language, called *reactive modules*. Quantitative properties are expressed using the probabilistic temporal logic PCTL [25] and Continuous Stochastic Logic (CSL) [26]. Both languages can express quantitative statements, such as the probability of occurrence of an event. CSL is in particular useful to express *steady-state* properties. For example, the property

Prop. 2. *"What is the probability that the signal will be available in the steady-state?"*

This property can be formally expressed as

$$\text{S}_{=?}\ [\texttt{3OC12HSL} > 0]$$

where S denotes the likelihood of a property to hold in the steady-state. PRISM also extends its property language with *reward* formulas. For example, the property

Prop. 3. *"What is the expected* `GFP` *at time t?"*

is specified in PRISM as follows:

$$\text{R}\{\text{“}\texttt{GFP}\text{”}\}_{=?}\ [\text{I} = t]$$

where R is the *reward* operator and I is the *instantaneous* time operator.

PRISM implements numerical methods (along with symbolic methods) to verify PCTL and CSL properties. The tool also permits statistical model checking, which relies on a discrete-event simulator. One particular drawback of the PRISM's statistical model checking component is that it allows only a restricted set of properties to be verified. For example, *steady-state* and complex properties with embedded operators are not supported.

2.3 PLASMA

PLASMA [4] is a *statistical* model checker for probabilistic systems. The tool uses a finite number of simulations to approximate the likelihood that an arbitrary simulation trace satisfies a property. The user either provides the number of the simulation runs, or it provides an error and probability, the tool then calculates the necessary number of simulations to ensure that the estimated result is within the error bound with the given probability [4].

PLASMA supports several models, including DTMCs and CTMCs. Properties are specified using the logic BLTL, a bounded variation of linear temporal logic (LTL) augmented with a probability operator. BLTL allows temporal properties bounded with time. For example, consider the following property:

Prop. 4. *"What is the probability that the* `3OC12HSL` *concentration exceeds m within t seconds?"*

It can be translated to BLTL as follows:

$$\text{P}_{=?}\ \left[\text{true}\ \text{U}^{\leq t}\ \texttt{3OC12HSL} > m\right]$$

where P is the *probability* operator representing the likelihood of a property to hold, and $\text{U}^{\leq t}$ is a *temporal* operator representing the *bounded until*, meaning that until the time point t.

2.4 MC2

MC2 [5] is another statistical model checking tool, which relies on the Monte Carlo approximation. The tool constructs a number of finite simulation traces, used to calculate the likelihood of properties to hold. The tool relies on PLTL_c as the property specification language, a probabilistic extension of Linear Temporal Logic (LTL) with "numerical constraints over real value variables". MC2 allows the full formula set of probabilistic properties, enriched with some additional *functions* such as *maximum/minimum* concentrations of a species and "*derivative* of the concentration of species at each time point" [5]. For example, in MC2, we can express properties such as

Prop. 5. *"It is true with a probability greater than 0.95 that* `GFP` *increases until it reaches the half of its maximum concentration."*

This property can be translated to PLTL_c as follows:

$$\text{P}_{>0.95}\left[\text{d}\,\texttt{[GFP]} > 0\ \text{U}\ \texttt{[GFP]} = \frac{1}{2}\text{max}\,\texttt{[GFP]}\right]$$

where U is a *temporal* operator representing *until*, $\text{d}\,\texttt{[GFP]}$ and $\text{max}\,\texttt{[GFP]}$ represent the *derivative* and *maximum* concentration of `GFP`, respectively. The formula then means that the `GFP` concentration increases *until* a time point, after which it *always* reduces and *eventually* reaches to the half of its *maximum* value.

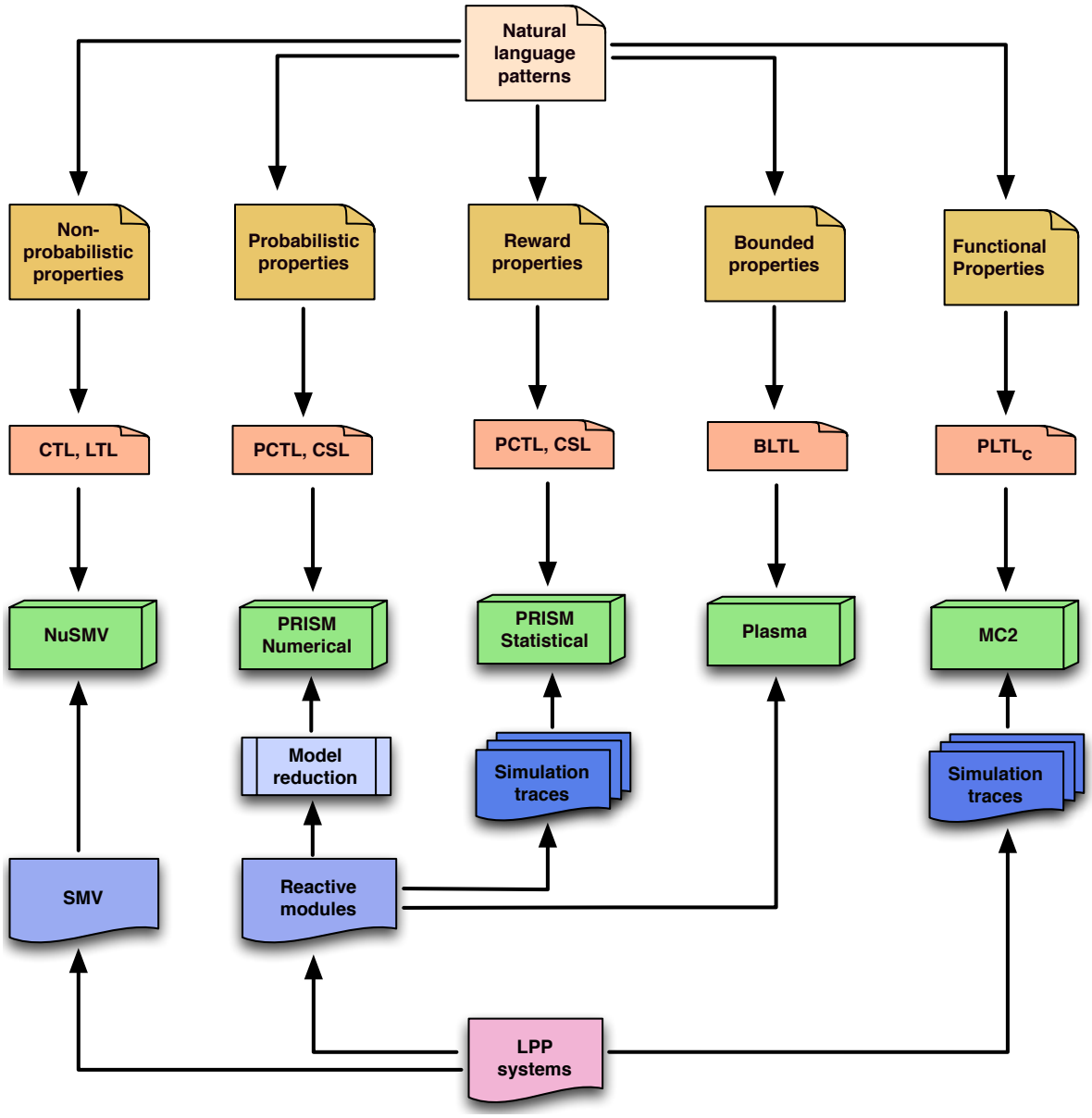


Fig. 1: A property-based methodology for model checking synthetic biology systems.

3 A Property-Driven Methodology

As described above, model checking is used to infer information about the system's behaviour. Depending on the type of analysis, we can ask various types of questions. However, these queries might require specific query languages. For example, while one query requires an exact result, for some other query it is sufficient to provide an approximate result.

In this section, we provide a methodology comprising different model checking approaches. This methodology is novel in the sense that rather than using a single method or tool it integrates different approaches so as to analyse different types of properties. This makes our methodology more flexible and easier to expand when new property types are considered.

The overall model checking methodology, which is described in more detail below, is presented in Figure 1. The individual components illustrated in this figure have been developed, and the third party model checking tools (free and open source) have been employed. As will be reported later, some of these components are already integrated into the current release of the INFOBIOTICS WORKBENCH software platform [8]. However, in this paper, our focus is the entire methodology, we therefore use the individual components standalone, and evaluate the methodology accordingly.

3.1 A Software Suit for Synthetic Biology

The INFOBIOTICS WORKBENCH (IBW) [8] is a software platform to model and analyse synthetic biology systems. This integrates various tools, enabling different

Category	Example
Non-probabilistic	The concentration of FIS decreases and then becomes steady in stationary phase [27]. GFP is preceded by the production of at least one of LacI or TetR [28].
Probabilistic	What is the probability that FGFR relocates and FGF is bound when relocation occurs [29]? What is the probability that the concentration of Raf-1/RKIP/ERK-PP complex will be less than M until the Raf-1/RKIP complex reaches the concentration C [30]?
Reward	What is the expected time taken before FGFR relocates [29]? What is the expected time to reach a state in which all of the gates have finished executing [31]?
Bounded	What is the probability that the monomer is at level i at time T [32]? The fraction of PP-ERK stays below the threshold with a given probability during the first 300 seconds [19].
Functional	The protein rises then falls to less than 100 mMol at 60 minutes [5]. The active RAS peaks within 2 minutes to a maximum of 20% of total RAS [5].

TABLE 1: Some example properties from the literature.

types of *in silico* experiments, e.g. simulation [22], model checking [21] and optimisation as well as graphical visualisation.

The IBW's simulator, MCSS, allows stochastic simulation or deterministic numerical integration of models based on multi-compartmental cells, whose dynamics are governed by a set of kinetic rules. In addition to standard (single compartment) Gillespie stochastic simulation algorithms (SSAs) [33], MCSS also employs a multi-compartmental SSA [34], [22]. IBW allows graphical viewing of simulation results in different formats, e.g. timeseries, histograms, 3D heat/surface maps and animations.

IBW integrates some model checking tools to enable formal analysis of temporal, dynamic, spacial and probabilistic behaviour of stochastic systems. Properties of system models are formulated as probabilistic logic formulas and automatically verified.

IBW also provides a model editor to create and edit models written in a dedicated high-level language. Another feature of IBW is that it translates the outputs of the experiments to various different formats. This permits to analyse the results using different tools outside the workbench.

3.2 Properties

In this paper, we identiy five property categories, *non-probabilistic*, *probabilistic*, *reward*, *bounded* and *functional*, which are extensively recurred in formal analysis of biological models. Some example properties from the literature are presented in Table 1. Each category type requires a different model checking approach.

3.2.1 Property categories

Category 1: Non-probabilistic properties are qualitative properties, used to capture qualitative information, in particular information regarding the reaction pathways and the network topology. Quantitative information, e.g. "concentration levels of species", is not the primary concern for this category. **Prop. 1** is an example of non-probabilistic properties.

Property specification: This type of properties contain only temporal aspect. They can therefore be expressed in LTL or CTL.

Model checking approach: Non-probabilistic properties can be verified using a model checker which supports LTL and CTL. Since the properties concern the network topology, we must query *all* possible pathways, which implies that the *exact* model checking result must be provided. We therefore choose NUSMV, which employs numerical methods to perform exact verification. Since SPIN does not support CTL, we do not consider it in the verification of the Category 1 properties.

Category 2: Probabilistic properties provide information about the *likelihood* of the occurrence of an event, e.g. "the concentration level of a species exceeds a threshold value". Here, we are particularly interested in *steady-state* properties, as they are recurring properties analysed in biological systems. **Prop. 2** is an example of probabilistic properties.

Property specification: This type of properties contain both temporal and probabilistic aspects. Such properties can be specified in PCTL or CSL.

Model checking approach: Probabilistic properties can be verified using a model checker which supports PCTL and CSL. Most properties in this category require *exact* verification results. We choose the PRISM model checker, because it supports both languages and employs numerical methods. We therefore use PRISM's *numerical* approach to verify properties using the corresponding techniques. Estimated results can be calculated using PRISM's statistical approach. However, the statistical approach supports a very restricted subset of PCTL and CSL (for example, the steady-state properties are not supported), we therefore do not consider the statistical approach for the Category 2 properties.

Category 3: Reward properties provide quantitative information regarding the *expected* concentration levels of species. So, instead of a probability value, they rely on precise concentrations or more complex *reward* structures [3]. **Prop. 3** is a typical example of reward properties.

Property specification: Although it is not part of the original language, PRISM extends PCTL and CSL with reward formulas.

Pattern	Example
Existence	The concentration of the signalling molecules exceeds 0.1 μM within 100 seconds.
Absence	The concentration of the signalling molecules never reaches 0.1 μM within the first 100 seconds.
Universality	The concentration of the signalling molecules is always below the threshold with a probability greater than 0.9.
Recurrence	The reporter protein is repeatedly produced when signalling molecules are introduced.
Steady-state	In the steady state, the concentration of the signalling molecules is more than 0.1 μM.
Until	The reporter gene will not be expressed until the concentration of the inducer is greatar than 0.2 μM.
Response	The production of the transcriptional regulator is followed by the production of the reporter protein.
Precedence	The production of the reporter protein is preceded by the production of the transcriptional regulator.
Reward	The expected concentration of the reporter protein at the time instant 100 seconds exceeds 1.0 nM.

TABLE 2: Property patterns that the NLQ tool features.

	Existence	Absence	Universality	Recurrence	Steady-state	Until	Response	Precedence	Reward
non-probabilistic	✓	✓	✓	✓	✓	✓	✓	✓	
probabilistic	✓	✓	✓	✓	✓	✓	✓	✓	
reward									✓
bounded	✓	✓	✓			✓	✓	✓	
functional	✓	✓	✓	✓	✓	✓	✓	✓	

TABLE 3: Property categories vs. property patterns

Model checking approach: This type of properties can be verified both *numerically* and *statistically*. Since we deal with the expected values, the *exact* verification results are not required. In order to benefit from its computational advantage, we therefore employ PRISM's statistical approach to verify reward properties. We note that MC2 and PLASMA do not support reward-based formulas.

Category 4: Bounded properties specify *transient* properties, i.e. properties constrained by time. While *unbounded properties* can express statements such as "an event eventually occurs", *bounded* ones express properties holding within a certain time bound, e.g. "an event occurs within a certain time". **Prop. 4** is an example of this category.

Property specification: Bounded properties contain temporal operators constrained by a time bound. BLTL, a bounded variant of LTL, can express such properties, because it contains bounded temporal operators. BLTL also employs a probability operator, allowing to specify properties such as those similar to Prop. 4.

Model checking approach: To verify bounded properties, we employ the PLASMA model checker. Here, we assume the *exact* verification results are not required. PLASMA can verify BLTL properties using some statistical techniques, and provides an approximate result for verification. We note that PRISM also allows verifying bounded properties. But, because of high computational resources demanded, it should be used only when precise verification results are needed. On the other hand, the tool's statistical approach supports a very restricted set of bounded properties. MC2 allows constraining formulas based on the number of computation steps, but it does not capture 'time bounds' required by these properties.

Category 5: Functional properties allow specification of probabilistic properties, enriched with some *functions* such as *maximum/minimum* and *decrease/increase* of the concentrations of species. As an example, we can consider **Prop. 5**.

Property specification: Such properties can be expressed in PLTL$_c$, a probabilistic extension of LTL with numerical constraints.

Model checking approach: To verify the properties of this category, we employ the MC2 tool. MC2 provides *approximate* results, calculated using the Monte Carlo approximation. Here, we assume the *exact* verification results are not required. We remark that none of the formalisms and tools used in previous categories can be used to verify Category 5 properties.

3.2.2 Property building

Model checking tools require system properties to be specified in a logical formal syntax. Expressing properties in a logical formalism is a cumbersome and error-prone process even for experts. In most cases, logical formulas are not clear and intuitive enough to capture their meanings.

In order to facilitate the property building process, we feature a *natural language query* (NLQ) tool, which constructs properties from a predefined set of property *patterns*. The idea is that using a graphical user interface, the user selects a pattern in the form of a predefined *natural language* statement and provides the necessary values (e.g. concentration amounts and time bounds), then the tool automatically converts the pattern to its formal counterpart based on the

Pattern	Natural language statement
Until	**#expr2** will eventually hold **[within time bound (#t1, #t2)]** until then **#expr1** holds **[with a probability #op #p]**
Response	**#expr1** is always followed by **#expr2 [within time bound (#t1, #t2)] [with a probability #op #p]**
Steady-state	**#expr** will hold in the steady-state **[with a probability #op #p]**
Reward	The expected reward for **#species** at time instant **#t** has the bound **#op #r**

TABLE 4: Natural language statements of a subset of property patterns.

Property	Pattern	Translation	Assignments
Prop. 1	Response	AG (#expr1 ⇒ EF #expr2)	#expr1: `3OC12HSL > 0`, #expr2: `GFP > 0`
Prop. 2	Steady-state	S#op #p [#expr]	#expr: `3OC12HSL > 0`, #op #p: `=?`
Prop. 3	Reward	R{"#species"}#op #p [I=#t]	#species: `GFP`, #op #p: `=?`, #t: `t`
Prop. 4	Until	P#op #p [#expr1 U[#t1, #t2] #expr2]	#expr1: `true`, #expr2: `3OC12HSL > m`, #op #p: `=?`, #t1: `0` #t2: `t`
Prop. 5	Until	P#op #p [#expr1 U #expr2]	#expr1: `d[GFP] > 0`, #expr2: `[GFP] =` $\frac{1}{2}$ `max[GFP]`, #op #p: `> 0.95`

TABLE 5: Property construction using the patterns.

target model checker selected. The tool supports all the logical formalisms and property categories used in our methodology. It is very flexible as it can be expanded easily when a new property category is added or a new target logic is required. The NLQ tool is compatible with the input/output formats of IBW. For example, it can parse an IBW model to extract the model variables, displayed to the user to construct atomic expressions. So, the tool's outputs can be directly used in IBW to perform verification.

Table 2 presents the list of patters, currently employed in the NLQ tool. These patterns have been derived from the most frequently used properties in systems biology. The idea of introducing patterns for recurring properties have been suggested in some previous studies e.g., [35], [36], [37], [38], [39], [40], where several pattern systems have been defined for different property *categories*, e.g. *non-probabilistic*, *probabilistic*, etc. Here, we introduce a novel approach, based on a different clustering mechanism, which associates property *patterns* to property *categories*. So, unlike the previous approaches, viewing property patterns in one dimensional line (for a particular category type), we view property patterns and property categories from the two-dimensional perspective. Table 3 shows how property patterns listed in Table 2 matches the category patterns. Our approach allows us to define different pattern sets for different property categories, which facilitates the use of an appropriate model checker for each property pattern.

In Table 3, some patterns are not defined for some category types. For example, the 'non-probabilistic', 'probabilistic', 'bounded' and 'functional' categories do not employ the reward operators; hence these categories do not cover the 'Reward' pattern. Similarly, the 'bounded' category does not have the operators to express the 'Steady-state' and 'Recurrence' patterns, because these two patterns require unbounded operators.

We now explain the property construction in more detail. The NLQ tool contains a pre-defined set of property patterns. Table 4 presents a subset of these patterns, from which we can construct **Prop. 1–5**. When a user selects a pattern type, the tool automatically displays the corresponding natural language statement. The user then fills in the fields written in bold using the GUI provided. The tool can read and parse an IBW model and extracts all model variables. The expressions (e.g. **#expr1** and **#expr2**), which represent atomic state formulas, can then be constructed using the model variables.

In the table, the patterns presented appear only in the relevant context. For example, the 'Reward' and 'Steady-state' patterns will only appear in the case of PRISM. **[within time bound (#t1, #t2)]** (where #t1 and #t2 are integer values), **[with a probability #op #p]** (where #op $\in \{<, >, =, \leq, \geq\}$ when #p $\in [0, 1]$, or #op #p is =?) are optional fields, which also appear in the relevant context. For example, **[with a probability #op #p]** does not appear for NUSMV. Similarly, the functions 'max' (denoting the maximum), 'min' (denoting the minimum) and 'd' (denoting the derivative) are only available (when constructing expressions) for MC2. In this way, we take care of presenting the correct form of any selected pattern and making the correct translation into the target model checker based on the optionally chosen probability and time bound fields.

Table 5 shows **Prop. 1–5** are obtained by instantiating the patterns presented in Table 4.

3.3 Models

To make the modelling task easy, a synthetic biology system should be modeled in an amenable formalism. In several case studies [41], [34], [8], it has been shown that *Lattice Population P (LPP) systems* are an intuitive and well-structured modelling formalism, suitable for synthetic biology. In this methodology, we therefore take LPP systems as our biomodel language.

3.3.1 LPP Systems

Many biological models are multi-cellular by design, where communication between cells is the key factor to run the overall system. For example, it is the signalling molecule `3OC12HSL` that provides communication between bacteria and triggers quorum sensing

within different cells. It is therefore essential to adapt a multi-cellular approach in the modelling language.

LPP systems [34], [8] are a a ruled-based modelling formalism, based on stochastic and multi-compartmental extension of P systems [42]. An LPP system consists of (i) a finite set of objects representing system entities (genes, proteins, promoters, RNAs, etc.), (ii) a finite set of *labels* naming membranes, (iii) *initial configuration* of the system (initial concentrations, etc.), (iv) *membrane structure* (a set of membranes representing regions) and (v) a set of *multiset writing rules* representing kinetic rules between species/molecules (gene regulation, complexation, degradation, etc.).

The IBW tool suit accepts system models written in a high-level modelling language, defined for LPP systems. For example, the expression of two proteins from a transcription unit containing two genes is written in this language as follows:

```
ProteinExpression({X,Y},{c_1,c_2,c_3},{l}) =
{
rules:
r1: [ gene_X_Y ]_l -c_1-> [ gene_X_Y + mRNA_X_Y ]_l
r2: [ mRNA_X_Y ]_l -c_2-> [ mRNA_X_Y + X ]_l
r3: [ mRNA_X_Y ]_l -c_3-> [ mRNA_X_Y + Y ]_l
}
```

LPP systems also allow a *spacial* representation. Namely, an LPP system embodies a two-dimensional geometric *lattice*. For example, the following lattice representation places the different bacterial strains over the colony by distributing copies of the bacterial strain 1 in the upper and lower boundaries of the lattice, and the strain 2 in the rest of the lattice.

```
spatialDistribution
    positions for bacteria_1
           parameters
             parameter i = 0:1:10
             parameter j = 0:10:10
          endParameters
          coordinates
              x=i
              y=j
           endCoordinates
    endPositions
    positions for bacteria_2
           parameters
             parameter i = 0:1:10
             parameter j = 1:1:9
          endParameters
          coordinates
              x=i
              y=j
           endCoordinates
    endPositions
endSpatialDistribution
```

In addition to the rules governing the kinetics within a cell, the language enables specifying *translocation* rules representing the transmission of objects between cells in the lattice. This makes the communication between cells possible.

The language also allows modularity and reusability by providing libraries of re-usable modules. A library contains a collection of modules (usually generic), and once created the modules within the library can be instantiated with different species and kinetic rates. Libraries can be also used in different models.

Since LPP systems enable specifying multi-compartmental, discrete and stochastic dynamics, it is a suitable modelling approach for synthetic biology.

3.3.2 Model translation

A typical model checker requires a mathematical representation of its input model written in a high-level modeling language. However, this is not an easy task for non-experts because (i) they must be very familiar with the modelling languages of all model checkers employed, and (ii) the representations in all the languages must be the same. Not surprisingly, this should not be expected from a non-expert.

In our methodology, we therefore consider the automatic translation of a synthetic biology model to the modeling languages of the model checkers we employ. Namely, once a model is provided in the form of LPP systems, the necessary inputs to these tools are automatically generated. So, users do not need to know anything about the syntax of the languages. The translators take care of the necessary translations, upon which the tools are called and verification experiments are performed automatically.

We have implemented the translators to generate the necessary inputs for the PRISM, PLASMA, MC2 and NUSMV tools.

3.3.3 Model reduction

Model checking has a well known problem: *state explosion*. Namely, the state space of a model can expand very quickly, which might make model checking very inefficient, and even impossible for some large models (as in the case of synthetic biology models). As discussed above, one solution is to use statistical model checking. However, if the exact verification results are required (as in the case of the Category 1 and Category 2 properties), we need to consider alternative methods.

A good strategy to relieve the effect of the state explosion problem is reducing the size of the state space. In our methodology, we devise three methods to do this:

(i) Reducing variable bounds: The state space is very sensitive to the ranges (i.e. *lower* and *upper* bounds) that model variables can take. In a typical biological system, it is almost not possible to know the precise ranges for products (and some reactants). In the formal model that a model checker receives as input, we therefore must provide estimated ranges for these species. If we use unrealistic bounds for variables, the resulting state space will be very large.

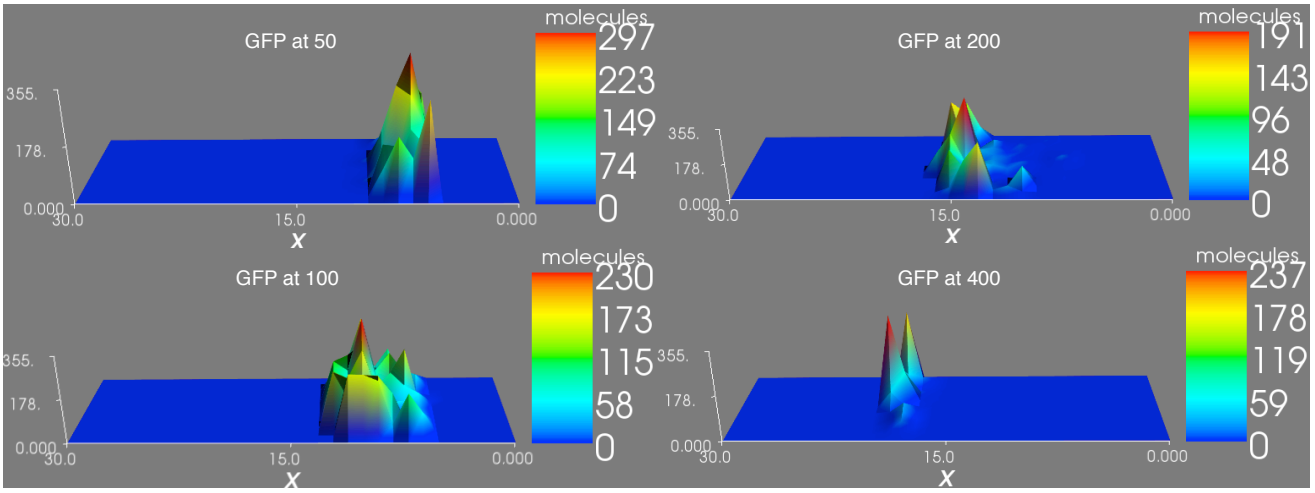


Fig. 2: Propagation of GFP along the bacterial colony (generated in IBW).

To eliminate this problem, we have devised a method that automatically predicts *reasonable* bounds for model variables. The method first generates a set of simulation traces, which are then analysed using an *invariant detector* tool to determine the maximum and minimum values of model variables. These values are then used as the lower and upper bounds. This method (i) saves users from providing bounds for model variables (a very tedious task for models with too many species) and (ii) prevents the increase in the state space caused by unrealistic bounds.

We have implemented this method using the DAIKON tool [43], which finds out mathematical properties (e.g. $x \leq 10$ and $y \leq 2x$) in a program. DAIKON is originally developed to find invariants in high-level programming languages e.g. C, C++ and Java. However, it has been also used to report invariants from simulation traces [44], where it is used to facilitate the specification of temporal properties, and to produce the boundaries of the simulated system together with other properties [45]. In this methodology, we extend the work of [45], [44] by adjusting it to accept the outputs of PRISM's discrete event simulator.

Daikon can report many types of invariants, such as *arithmetic* ($y > 3x$), *non-zero* (e.g. $x \neq 1$), *element of* (e.g. x is one of $\{0, 1, 2\}$) and *interval* (e.g. $0 \leq x \leq 10$). The tool can also find some redundant invariants (e.g. $x == x$). However, it employs a filtering mechanism that permits omitting redundant and unwanted invariants to be returned. In this way, we can only obtain the *upper bound* (e.g. $x \leq 2$) and *lower bound* (e.g. $x \geq 0$) invariants.

We remark that this method is only applied to the Category 2 properties. The Category 1 properties are verified against non-probabilistic models. We can therefore ignore the kinetic aspect and consider Boolean values for model variables [28], [46].

(ii) Resizing concentration levels: Reducing variable bounds prevents increasing the state space as a result of unrealistic bounds. However, the system might still operate on high concentration levels. Modeling the exact concentration levels has a profound effect on the state space.

As a complementary approach to reducing variable bounds, we have devised a method which resizes all concentration levels by the same factor. To keep the kinetic behaviour same, we also slow down reactions by scaling down the kinetic constants by the same factor.

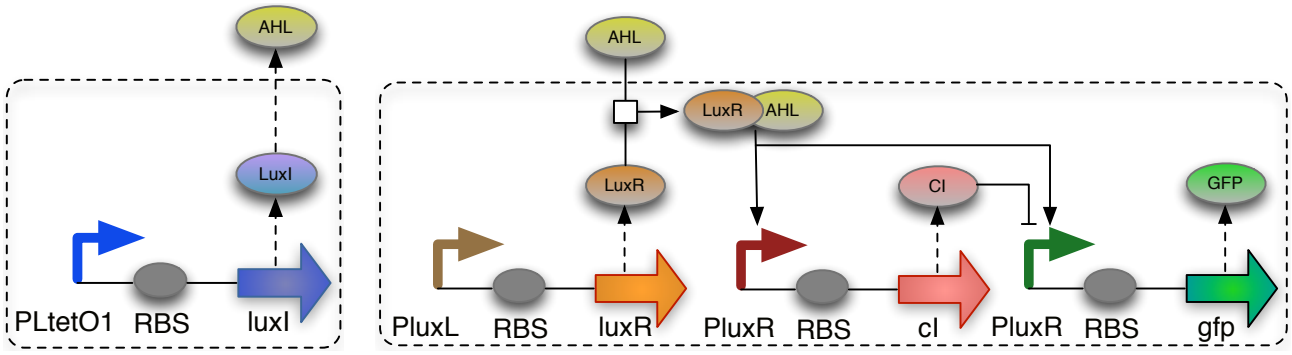
Currently, this method is applied to LPP models manually. An automatic deployment of the method is under construction.

(iii) Simplifying the kinetic rules: Non-probabilistic models can also be simplified by replacing a long chain of reactions by a simpler rule set which will capture the starting and ending parts of this chain, and hence eliminating species that do not appear in the new rule set. With this transformation we can achieve a simplification of the state space, but also of the number of transitions associated with the model [46].

4 A SYNTHETIC BIOLOGY SYSTEM: PULSE-GENERATOR

In this section, we will describe a synthetic biology construct, to which our methodology will be applied. The *pulse-generator* is a synthetic bacterial colony constructed by Weiss et. al [47], [48]. The system first produces a signalling molecule, which triggers the expression of the green fluorescent protein (GFP), and then propagates GFP along the bacterial colony (see Figure 2). The pulse generator consists of two types of cells, *sender* and *pulsing* cells (see Fig. 3), which are described as follows [8]:

> "Sender cells contain the gene luxI from *Vibrio fischeri*. This gene codifies the enzyme LuxI responsible for the synthesis of the



(a) Sender cell. (b) Pulsing cell.

Fig. 3: The sender and pulsing cells of the pulse generator system (reproduced from [8]).

molecular signal 3OC12HSL (AHL). The luxI gene is expressed constitutively under the regulation of the promoter PLtetO1 from the tetracycline resistance transposon."

"Pulsing cells contain the luxR gene from *Vibrio fischeri* that codifies the 3OC12HSL receptor protein LuxR. This gene is under the constitutive expression of the promoter PluxL. It also contains the gene cI from lambda phage codifying the repressor CI under the regulation of the promoter PluxR that is activated upon binding of the transcription factor LuxR_3OC12. Finally, this bacterial strain carries the gene gfp that codifies the green fluorescent protein under the regulation of the synthetic promoter PluxPR combining the Plux promoter (activated by the transcription factor LuxR_3OC12) and the PR promoter from lambda phage (repressed by the transcription factor CI)."

The sender and pulsing cells are placed along a lattice. Sender cells are placed at one end and pulsing cells are distributed at the remaining part of the lattice. We have previously carried out some verification analysis for the pulse generator system [8], but this was only based on PRISM.

5 Applying the Methodology to the Pulse Generator

In this section, we will apply our methodology to the pulse generator construct. Before presenting the experimental results, we will first describe the models to be used in the verification experiments.

Stochastic model. We create an LPP model for two bacterial strains, which represent the stochastic behaviour of each cell. The reaction rules describe the regulation of the corresponding promoters used in the sender and pulsing cells. The sender cells consist of 11 kinetic rules, which model the production of the 3OC12HSL signal, and the pulsing cells comprise 38 kinetic rules, which capture the production of the GFP protein (as a response to 3OC12HSL).

The geometry of the bacterial colony is described by a lattice, where the sender and pulsing cells are distributed over specific regions. The lattice is surrounded by *boundary* cells, which serve as a buffer for molecules emitted from the border cells. In our experiments below, we consider a 5×10 lattice (including the boundary cells), where the sender cells are placed up to the first three columns of the lattice, and the pulsing cells are distributed over the remaining locations.

Non-deterministic model. This model is a qualitative abstraction obtained from the stochastic model by removing the kinetic constants of the reaction rules. The model captures the entire reaction network and all pathways; but quantitative aspects of the system, e.g. the concentration levels of species, are not represented. It describes the presence of molecular species rather than their concentrations.

We note that the complete models and experimental results can be accessed at [49].

5.1 Category 1: Non-probabilistic properties

Non-probabilistic properties are queried (*exhaustively*) against the non-probabilistic model (representing the basic model for qualitative analysis) using the NUSMV model checker.

We now verify **Prop. 1**, representing the dependency between molecular species and the sequence of events occurring on various reaction pathways. The corresponding CTL formula can be automatically built, as described in Section 3.2.2. Table 6 presents the verification result for this property.

5.2 Category 2: Probabilistic properties

In this approach, probabilistic properties are *exhaustively* checked against a reactive modules model, automatically generated from the LPP (stochastic) model. We remark that the approach is not feasible for

Property	Langauge	Model checker	Result
Prop. 1	CTL	NuSMV	True
Prop. 2	CSL	Prism numerical	0.95
Prop. 3	CSL	Prism statistical	Figure 4a
Prop. 4	BLTL	Plasma	Figure 4b
Prop. 5	PLTL_c	MC2	True

TABLE 6: Verification of property categories.

model	state space	constr. time
estimated bounds	3×10^{11}	62 sec.
reduced variable bounds	5×10^{5}	0.1 sec.
reduced variable bounds + resized concentration levels	9×10^{2}	0.006 sec.

TABLE 7: State space and model construction time for three different models.

large models because of the computational resources required. For example, the lattice we have defined contains more than 1000 rules, so model checking is not eligible for such a large model. We therefore consider a single cell type.

Our methodology applies the model reduction methods, presented in Section 3.3.3, to make the numerical model checking more feasible. Table 7 compares the sizes of the state spaces and the construction times for three different models of a *sender* cell: (i) variable bounds are assigned to an estimated value of 200; (ii) realistic upper bounds obtained from the invariant detector are used; and (iii) the reduced upper bounds are used and they are resized by a scale factor 1/10. As shown in Table 7, applying both techniques significantly reduces the state space, making model checking more amenable.

We now verify **Prop. 2**, representing a steady-state property, against the sender cell model described in (iii). As discussed in Section 3.2.1, we assume that the exact verification results are required for this category. We therefore use Prism's numerical engine to verify the property (expressed in CSL), whose result is given in Table 6. The CSL formulas can be automatically generated using the NLQ tool.

5.3 Category 3: Reward properties

To verify reward-based formulas, we automatically generate simulation traces from the reactive modules model, and use Prism's *statistical* model checking engine. We remind that MC2 and Plasma do not support reward-based formulas. The reward formulas (featured in Prism's property language) can be automatically generated using the natural language query patterns, as described in Section 3.2.2.

We now verify **Prop. 3** to obtain the concentration level of GFP. The verification result is illustrated in Figure 4a, which clearly shows that GFP propagates through the pulsing cells. Namely, GFP is produced earlier in the pulsing cells close to the sender cells than those further away.

5.4 Category 4: Bounded properties

Bounded properties are verified using the Plasma model checker. The stochastic model is automatically converted to the input format of Plasma. Also, the BLTL properties are generated using the property patterns in the NLQ tool. Plasma then performs verification using *statistical* methods.

Prop. 4 represents a *transient* property, which is verified using this approach. The verification result for $t_1 = t_2 = T$ and $m_1 = 0, m_2 = 200$ is presented in Figure 4b, which is inline with the propagation behaviour described in Section 4. We remind that Prism's statistical model checking does not support Prop. 4 (in its general form), and MC2 does not capture the time bounds specified in the property.

5.5 Category 5: Functional properties

We use MC2 to verify the properties of this type, as the tool's specification language, PLTL_c, supports some functions to query the quantitative information such as the max/min and decrease/increase of the species concentrations. The simulation traces are automatically generated from an LPP model, and PLTL_c properties are automatically constructed using the NLQ tool.

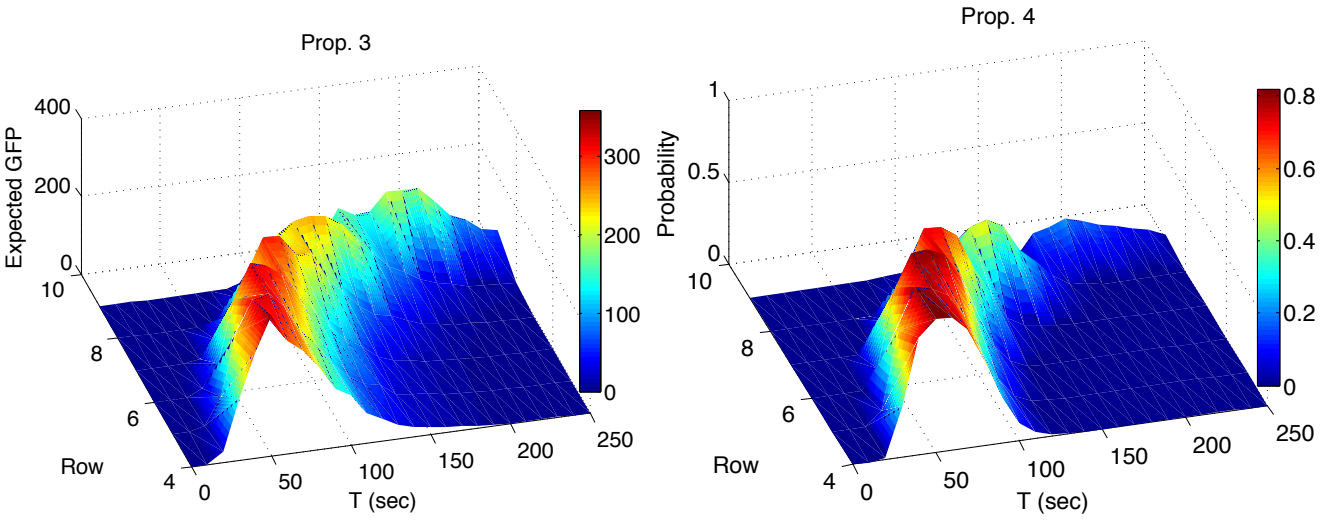
Table 6 illustrates the verification result of **Prop. 5**. The property is satisfied for all pulsing cells in the lattice. We note that this property cannot be verified using the other model checkers.

6 Conclusion

In this paper, we have proposed a methodology to formally analyse bio-systems, in particular synthetic biology systems. The methodology suggests an integrative perspective using different model checking approaches based on the different property types. The methodology has been applied to the synthetic pulse generator. We have carried out several verification experiments to illustrate the use of our approach.

The methodology facilitates the model checking process by automating the model construction and property generation. Also, integrating various model checkers enables us to formally analyse different aspects of a system. This approach makes the model checking process easier and more accessible to non-experts.

The Ibw software platform currently integrates the model translation and property construction components. Some of the model reduction methods, i.e. resizing concentration levels and simplifying the kinetic rules, have not been integrated into the tool, because we currently do not have algorithms, finding optimal solutions for different models. As the current and future work, we will integrate the remaining unintegrated parts to the next release of Ibw, and demonstrate the approach on new state-of-art synthetic biology systems. We are also working on some



(a) Expected GFP (Prop. 3) (b) Transient property (Prop. 4)

Fig. 4: Verification results at the consecutive rows in the lattice, where pulsing cells are distributed.

new features, e.g. high-performance simulators. All these features will be available in the next release of the platform.

ACKNOWLEDGMENTS

This work is supported by the EPSRC's ROADBLOCK project (grant number: EP/I031812/1). M.G. is also partially supported by the MuVet project (CNCS UEFISCDI – grant number PN-II-ID-PCE-2011-3-0688). The authors thank the anonymous reviewers for their valuable comments. The authors also acknowledge helpful discussions with some partners in this project, including Natalio Krasnogor, Sara Kalvala, Harold Fellermann, Christophe Ladroue, Daven Sanassy, Laurentiu Mierla, Jamie Twycross, Jonathan Blakes and Francisco Jose Romero-Campero.

REFERENCES

[1] A. A. Cheng and T. K. Lu, "Synthetic biology: An emerging engineering discipline," *Annual Review of Biomedical Engineering*, vol. 14, no. 1, pp. 155–178, 2012.

[2] A. Cimatti, E. Clarke, E. Giunchiglia, F. Giunchiglia, M. Pistore, M. Roveri, R. Sebastiani, and A. Tacchella, "NuSMV Version 2: An OpenSource Tool for Symbolic Model Checking," in *Proc. International Conference on Computer-Aided Verification (CAV 2002)*, ser. LNCS, vol. 2404. Springer, 2002.

[3] A. Hinton, M. Kwiatkowska, G. Norman, and D. Parker, "PRISM: A tool for automatic verification of prob. sys." in *Proc. TACAS*, ser. LNCS, vol. 3920. Springer, 2006, pp. 441–444.

[4] B. Boyer, K. Corre, A. Legay, and S. Sedwards, "Plasma-lab: A flexible, distributable statistical model checking library," in *Quantitative Evaluation of Systems*, ser. Lecture Notes in Computer Science. Springer Berlin, 2013, vol. 8054, pp. 160–164.

[5] R. Donaldson and D. Gilbert, "A monte carlo model checker for Probabilistic LTL with numerical constraints," Dept. of Computing Science, University of Glasgow, Research Report TR-2008-282, 2008.

[6] J. Fisher and N. Piterman, "Model checking in biology," in *A Systems Theoretic Approach to Systems and Synthetic Biology I: Models and System Characterizations*. Springer, 2014, pp. 255–279.

[7] M. Heiner, D. Gilbert, and R. Donaldson, "Petri Nets for Systems and Synthetic Biology," in *Formal Methods for Computational Systems Biology*, ser. LNCS, vol. 5016. Springer, 2008, pp. 215–264.

[8] J. Blakes, J. Twycross, S. Konur, F. J. Romero-Campero, N. Krasnogor, and M. Gheorghe, "Infobiotics workbench: A P systems based tool for systems and synthetic biology," in *Applications of Membrane Computing in Systems and Synthetic Biology*, ser. Emergence, Complexity and Computation. Springer International Publishing, 2014, vol. 7, pp. 1–41.

[9] S. Konur, "A survey on temporal logics for specifying and verifying real-time systems," *Frontiers of Computer Science*, vol. 7, no. 3, pp. 370–403, 2013.

[10] R. Alur, K. McMillan, and D. Peled, "Model-checking of correctness conditions for concurrent objects," *Information and Computation*, vol. 160, no. 1-2, pp. 167 – 188, 2000.

[11] M. Yabandeh, "Model Checking of Distributed Algorithm Implementations," Ph.D. dissertation, IC, 2011.

[12] S. Konur, M. Fisher, and S. Schewe, "Combined model checking for temporal, probabilistic, and real-time logics," *Theoretical Computer Science*, vol. 503, no. 0, pp. 61 – 88, 2013.

[13] S. Konur, M. Fisher, S. Dobson, and S. Knox, "Formal Verification of a Pervasive Messaging System," *Formal Aspects of Computing*, vol. 26, no. 4, pp. 677–694, 2014.

[14] S. Konur, C. Dixon, and M. Fisher, "Analysing robot swarm behaviour via probabilistic model checking," *Robotics and Autonomous Systems*, vol. 60, no. 2, pp. 199 – 213, 2012.

[15] M. Heiner, D. Gilbert, and R. Donaldson, "Petri Nets for Systems and Synthetic Biology," in *Formal Methods for Computational Systems Biology*, ser. LNCS, vol. 5016. Springer, 2008, pp. 215–264.

[16] J. Heath, M. Kwiatkowska, G. Norman, D. Parker, and O. Tymchyshyn, "Probabilistic model checking of complex biological pathways," *Theoretical Computer Science*, vol. 319, no. 3, pp. 239–257, 2008.

[17] F. J. Romero-Campero, M. Gheorghe, L. Bianco, D. Pescini, M. J. Perez-Jimenez, and R. Ceterchi, "Towards probabilistic model checking on P systems using PRISM," in *Membrane Computing*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2006, vol. 4361, pp. 477–495.

[18] S. Eker, M. Knapp, K. Laderoute, P. Lincoln, J. Meseguer, and K. Sonmez, "Pathway logic: Symbolic analysis of biological signaling," in *Proceedings of the Pacific Symposium on Biocomputing*, 2002, pp. 400–412.

[19] E. M. Clarke, J. R. Faeder, C. J. Langmead, L. A. Harris, S. K. Jha, and A. Legay, "Statistical model checking in BioLab: Applications to the automated analysis of T-cell receptor signaling pathway," in *Computational Methods in Systems Biology*,

ser. Lecture Notes in Computer Science. Springer, 2008, vol. 5307, pp. 231–250.

[20] L. Calzone, N. Chabrier-Rivier, F. Fages, and S. Soliman, "Machine learning biochemical networks from temporal logic properties," in *Transactions on Computational Systems Biology VI*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2006, vol. 4220, pp. 68–94.

[21] S. Konur, M. Gheorghe, C. Dragomir, L. Mierla, F. Ipate, and N. Krasnogor, "Qualitative and quantitative analysis of systems and synthetic biology constructs using P systems," *ACS Synthetic Biology*, 2014.

[22] D. Sanassy, H. Fellermann, N. Krasnogor, S. Konur, L. Mierlă, M. Gheorghe, C. Ladroue, and S. Kalvala, "Modelling and stochastic simulation of synthetic biological boolean gates," in *The 16th IEEE International Conference on High Performance Computing and Communications*, 2014.

[23] S. Konur and M. Fisher, "A roadmap to pervasive systems verification," *The Knowledge Engineering Review*, 2014.

[24] G. J. Holzmann, "The model checker SPIN," *IEEE Transactions on Software Engineering*, vol. 23, no. 5, pp. 275–295, 1997.

[25] H. Hansson and B. Jonsson, "A logic for reasoning about time and reliability," *Formal Aspects of Computing*, vol. 6, pp. 102–111, 1994.

[26] C. Baier, B. Haverkort, H. Hermanns, and J. Katoen, "Model-checking algorithms for continuous-time markov chains," *Software Engineering, IEEE Transactions on*, vol. 29, no. 6, pp. 524–541, 2003.

[27] 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," *Bioinformatics*, vol. 21, no. suppl 1, pp. i19–i28, 2005.

[28] S. Konur, M. Gheorghe, C. Dragomir, F. Ipate, and N. Krasnogor, "Conventional verification for unconventional computing: a genetic XOR gate example," *Fundamenta Informaticae*, 2014.

[29] M. Kwiatkowska, G. Norman, and D. Parker, *Symbolic Systems Biology*. Jones and Bartlett, 2010, ch. Probabilistic Model Checking for Systems Biology, pp. 31–59.

[30] M. Calder, V. Vyshemirsky, D. Gilbert, and R. Orton, "Analysis of signalling pathways using the PRISM model checker," in *Proc. Computational Methods in Systems Biology (CMSB'05)*, 2005, pp. 179–190.

[31] M. Lakin, D. Parker, L. Cardelli, M. Kwiatkowska, and A. Phillips, "Design and analysis of DNA strand displacement devices using probabilistic model checking," *Journal of the Royal Society Interface*, vol. 9, no. 72, pp. 1470–1485, 2012.

[32] F. Ciocchetta and J. Hillston, "Bio-PEPA: A framework for the modelling and analysis of biological systems," *Theoretical Comput. Science*, vol. 410, no. 33-34, pp. 3065–3084, 2009.

[33] D. Gillespie, "A general method for numerically simulating the stochastic time evolution of coupled chemical reactions," *Journal of Computational Physics*, vol. 22, no. 4, pp. 403–434, 1976.

[34] F. J. Romero-Campero, J. Twycross, M. Camara, M. Bennett, M. Gheorghe, and N. Krasnogor, "Modular assembly of cell systems biology models using P systems," *International Journal of Foundations of Computer Science*, vol. 20, no. 03, pp. 427–442, 2009.

[35] M. B. Dwyer, G. S. Avrunin, and J. C. Corbett, "Patterns in property specifications for finite-state verification," in *Proceedings of the 21st international conference on Software engineering*, ser. ICSE '99. ACM, 1999, pp. 411–420.

[36] L. Grunske, "Specification patterns for probabilistic quality properties," in *Proceedings of the 30th international conference on Software engineering*, ser. ICSE '08. ACM, 2008, pp. 31–40.

[37] P. T. Monteiro, D. Ropers, R. Mateescu, A. T. Freitas, and H. de Jong, "Temporal logic patterns for querying dynamic models of cellular interaction networks," *Bioinformatics*, vol. 24, no. 16, pp. i227–i233, 2008.

[38] P. Bellini, P. Nesi, and D. Rogai, "Expressing and organizing real-time specification patterns via temporal logics," *Journal of Systems and Software*, vol. 82, no. 2, pp. 183–196, Feb. 2009.

[39] C. Dragomir, F. Ipate, S. Konur, R. Lefticaru, and L. Mierlă, "Model Checking Kernel P Systems," in *14th International Conference on Membrane Computing*, ser. LNCS, vol. 8340. Springer, 2013, pp. 151–172.

[40] S. Konur, "Towards light-weight probabilistic model checking," *Applied Mathematics*, vol. 2014, p. 15 pages, 2014.

[41] F. J. Romero-Campero, J. Twycross, H. Cao, J. Blakes, and Natalio, "A multiscale modeling framework based on P systems," in *Membrane Computing*, ser. LNCS. Springer, 2009, vol. 5391, pp. 63–77.

[42] G. Păun, "Computing with Membranes," *Journal of Computer and System Sciences*, vol. 61, no. 1, pp. 108–143, 2000.

[43] M. D. Ernst, J. H. Perkins, P. J. Guo, S. McCamant, C. Pacheco, M. S. Tschantz, and C. Xiao, "The Daikon system for dynamic detection of likely invariants," *Science of Computer Programming*, vol. 69, no. 1–3, pp. 35 – 45, 2007.

[44] R. Lefticaru, F. Ipate, L. Valencia-Cabrera, A. Turcanu, C. Tudose, M. Gheorghe, M. J. Pérez-Jiménez, I. M. Niculescu, and C. Dragomir, "Towards an integrated approach for model simulation, property extraction and verification of P systems," *Tenth Brainstorming Week on Membrane Computing*, vol. vol. I, pp. 291–318, 2012.

[45] F. Bernardini, M. Gheorghe, F. Romero-Campero, and N. Walkinshaw, "A hybrid approach to modeling biological systems," in *Membrane Computing*, ser. Lecture Notes in Computer Science. Springer, 2007, vol. 4860, pp. 138–159.

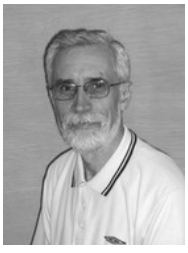
[46] M. E. Bakir, F. Ipate, S. Konur, L. Mierlă, and I. Niculescu, "Extended simulation and verification platform for kernel P systems," in *15th International Conference on Membrane Computing*, 2014.

[47] S. Basu, R. Mehreja, S. Thiberge, M.-T. Chen, and R. Weiss, "Spatiotemporal control of gene expression with pulse-generating networks," *PNAS*, vol. 101, no. 17, pp. 6355–6360, 2004.

[48] S. Basu, Y. Gerchman, C. H. Collins, F. H. Arnold, and R. Weiss, "A synthetic multicellular system for programmed pattern formation," *Nature*, vol. 434, pp. 1130–1134, 2005.

[49] "Pulse generator," http://www.dcs.shef.ac.uk/∼konur/models/pulsegenerator.

Savas Konur Savas Konur received his PhD degree in Computer Science from the University of Manchester (UK) in 2008. He is currently a Research Associate and a member of the Verification and Testing research group in the Department of Computer Science, University of Sheffield (UK). He previously worked in the Department of Computer Science, University of Liverpool (UK). His research interests include temporal reasoning, formal specification and verification, model checking, and application of formal methods to systems and synthetic biology, real-time systems and pervasive systems.



Marian Gheorghe Marian Gheorghe received his PhD from the Department of Computer Science of the University of Bucharest (Romania) in 1991. He is currently a Reader and the Head of the Verification and Testing research group in the Department of Computer Science, University of Sheffield (UK). He has been working for a long time in defining and using computational models, in formal verification and testing and applications of them to systems and synthetic biology. He has published widely in these areas and has been involved in research projects on these topics.