Expressive Statistical Model Checking of Genetic Networks with Delayed Stochastic Dynamics

Paolo Ballarini, Jarno Makela, Andre Ribeiro

Expressive Statistical Model Checking of Genetic Networks with Delayed Stochastic Dynamics

Paolo Ballarini[1], Jarno Mäkelä[2], and Andre S. Ribeiro[2]

[1] Ecole Centrale Paris, France
paolo.ballarini@ecp.fr
[2] Tampere University of Technology, Finland
{andre.ribeiro,jarno.makela}@tut.fi

Abstract. The recently introduced Hybrid Automata Stochastic Logic (HASL) [8] establishes a powerful framework for the analysis of a broad class of stochastic processes, namely Discrete Event Stochastic Processes (DESPs). Here we demonstrate the potential of HASL based verification in the context of genetic circuits. To this aim we consider the analysis of a model of gene expression with delayed stochastic dynamics, a class of systems whose dynamics includes both Markovian and non-Markovian events. We identify a number of relevant properties related to this model, formally express them in HASL terms and, assess them with COSMOS, a statistical model checker for HASL model checking. We demonstrate that this allows assessing the "performances" of a biological system beyond the capability of other stochastic logics.

Keywords: Statistical model checking, genetic networks, stochastic dynamics, stochastic petri nets

1 Introduction

Biological systems are regulated by complex information processing mechanisms which are at the basis of their survival and adaptation to environmental changes. Despite the continuous advancements in experimental methods many of those mechanisms remain little understood. The end goal of computational systems biology [26] is to help filling in such knowledge gap by developing formal methods for rigorously representing and effectively analysing biological systems. Understanding what cells actually compute, how they perform computations and, eventually, how such computations can be modified/engineered are essential tasks which computational modelling aims to. In this context, the ability to "interrogate" a model by posing relevant "questions", referred to as *model checking*, is critical. Model checking approaches have proved effective means to the analysis of biological systems, both in the framework of non-probabilistic models [19, 12] and in that of stochastic models [28, 23].

Our contribution. We consider the application of a recently introduced stochastic logic, namely the Hybrid Automata Stochastic Logic (HASL), to the verification of biological systems. Our contribution is twofold. In the first part we demonstrate the effectiveness of HASL verification by developing a full case study of gene expression, a relevant biological mechanism represented by means of non-Markovian models, and which, therefore, cannot be analysed by means of classical (Markovian) stochastic model checking. In the second part we introduce preliminary results illustrating the effectiveness of HASL verification in dealing with a rather relevant aspect of many biological mechanisms, namely: the analysis of oscillatory trends in stochastic models of biological systems.

Paper organization. In Section 2 we provide some backgrounds which put into context the proposed approach. In Section 3 we introduce the gene expression mechanism with stochastic delays we refer to in the remainder of the paper. In Section 4 we recall the basics of the HASL formalism. In Section 5 we present the formal analysis (by means of HASL) of the previously introduced single-gene model. In section 6 we illustrate the application of HASL to measurements of oscillations. Conclusive remarks are given in Section 7.

2 Background

Model Checking and Systems Biology. Model checking is a technique addressing the formal verification of discrete-event systems. Its success is mainly due to the following points: (1) the ability to express specific properties by formulas of an appropriate logic, (2) the firm mathematical foundations based on automata theory and (3) the simplicity of the verification algorithms which has led to the development of numerous tools. Initially [17] targeted to the verification of functional *qualitative* properties of non-probabilistic models by means of "classical" temporal logics (i.e. LTL, CTL), model checking has progressively been extended toward the performance and dependability analysis realm (i.e. *quantitative* verification) by adaptation of classical temporal logics to express properties of Markov chains [21],[6]. In systems biology [26] two modelling alternatives are typical: (1) the continuous-deterministic framework, whereby dynamics of biological agents are expressed in terms of (a system of) differential equations (e.g. ODE, PLDE) and (2) the discrete-stochastic framework, whereby dynamics are expressed in terms of a stochastic process (most often a continuous-time Markov chain). The application of model checking to systems biology has targeted both modeling frameworks. BIOCHAM [18], GNA [15], BioDIVINE [10] are examples of tools providing LTL/CTL model-checking functionalities for the verification of *qualitative* properties of biological models represented by means of differential equations. Conversely, Prism[31], MARCIE [35] are examples of tools featuring Continuos Stochastic Logic (CSL) [6] model-checking for the verification of *quantitative* properties of continuous-time Markov chains (CTMC) models of biological systems. Recently *linear-time* reasoning (as opposed to CSL *branching-time* reasoning) has been extended to the probabilistic framework as well. Examples are: the addition of LTL properties specifications in PRISM; the introduction of the bounded LTL, i.e. BLTL [24].

3 Genetic Networks with Delayed Stochastic Dynamics

Gene expression is the process by which proteins are synthesized from a sequence in the DNA. It consists of two main phases: *transcription* and *translation.* Transcription is the copying of a sequence in the DNA strand by an RNA polymerase (RNAp) into an RNA molecule. This process takes place in three main stages: initiation, elongation and termination. Initiation consists of the binding of the RNAp to a promoter (Pro) region, unwinding the DNA and promoter escape. Afterwards, elongation takes place, during which the RNA sequence is formed, following the DNA code. Once the termination sequence is reached, both the RNAp and the RNA are released. In prokaryotes, translation, the process by which proteins are synthesized from the (transcribed) RNA sequence, can start as soon as the Ribosome Binding Site (RBS) region of the RNA is formed.

The rate of expression of a gene is usually regulated at the stage of transcription, by activator/repressor molecules that can bind to the operator sites (generally located at the promoter region of the gene) and then promote/inhibit transcription initiation. Evidence suggests that this is a highly stochastic process (see, e.g. [4]), since usually, the number of molecules involved, e.g. transcription factors and promoter regions, is very small, ranging from one to a few at a given moment [37]. Due to that, stochastic modeling approaches were found to be more appropriate than other strategies (e.g. ODE models or Boolean logic).

Stochastic models of gene expression with delayed dynamics. The first stochastic models of gene expression assumed that the process of gene expression, once initialized, is instantaneous [4]. Namely, each step was modeled as a uni- or bi-molecular reaction and its kinetics was driven by the stochastic simulation algorithm (SSA) [20]. These models do not account for one important aspect of the kinetics of gene expression. Namely, that it consists, as mentioned, of a sequential process whose intermediate steps take considerable time to be completed once initiated (see e.g. [25]). This feature can be accounted for by introducing 'time delays' in the appearance of the products modeling the process [13, 34, 32].
Biochemical reaction with stochastic delays can be generally denoted as:

$$\sum n_i R_i \xrightarrow{k} \sum m_j P_j^{nd} + \sum m'_k P_k^d(dist_k)$$

where R_i, P_j^{nd} and P_k^d denote, respectively, the i-th reactant, the j-th non-delayed product and the k-th delayed product (n_i, m_j and m'_k being the stoichiometric coefficients) and $dist_k$ denotes the distribution for the delayed introduction of k-th delayed product. For example, reaction $A + B \xrightarrow{k} A + C(\delta(\tau))$ represents a reaction between molecules A and B, that produces molecule C from B by a process that takes τ seconds to occur once initiated (i.e. $\delta(t)$ denotes a delta dirac distribution centred in t). When this reaction occurs, the number of molecules A is kept constant, a molecule B is immediately removed from the system and a molecule C is introduced in the system τ seconds after the reaction takes place.

To deal with delayed reactions different adaptations of Gillespie's SSA algorithm (referred to as "delayed SSA") have been introduced. Initially two methods [13, 11] were proposed for implementing reactions with delays. Then [34] introduced, a generalization of the method proposed in [13], in that it allows multiple time delays in a single reacting event. This algorithm allows implementing a generalized modeling strategy of gene networks [32] and is the one which the SGNSim [33] tool is based on.

3.1 Single gene expression model

We consider a model of single gene expression that follows the approach proposed in [32]. Our model differs in that transcription is modeled as a 2-step process so as to accurately account for the open complex formation and promoter escape [25]. Each of these processes duration follows an exponential distribution. The gene expression system we refer to consists of the following reactions[3]:

$$R_1: \; Pro + *RNAp \xrightarrow{k_t} Prox \tag{1}$$

$$R_2: \; Prox \xrightarrow{\lambda_1} Pro + RBS + R(\Gamma(G_{len}, 0.09)) \tag{2}$$

$$R_3: \; *Rib + RBS \xrightarrow{k_{tr}} RBS(\delta(\tau_1)) + Rib(\Gamma(G_{len}, 0.06)) + P(\Gamma(\tau_{5_{sh}}, \tau_{5_{sc}})) \tag{3}$$

$$R_4: \; RBS \xrightarrow{rbsd} \emptyset \tag{4}$$

$$R_5: \; Pro + Rep \xrightarrow{k_r} ProRep \tag{5}$$

$$R_6: \; ProRep \xrightarrow{k_{unr}} Pro + Rep \tag{6}$$

Reactions (1) and (2) model transcription. In (1), an RNAp binds to a promoter (Pro), which remains unavailable for more reactions until reaction (2) occurs. Following reaction (2), which models the promoter escape, both the promoter and the RBS become available for reactions. Also from reaction (2), once transcription is completed, at $\Gamma(G_{len}, 0.09)$, a complete RNA (represented by R) is released in the system. R will not be substrate to any reaction, and is only modeled as a means to count the number of RNA molecules produced over a certain period of time. In our model, according to the SSA, the time necessary for any reaction to occur follows an exponential distribution whose mean is determined by the product between the rate constant of the reaction with the number of each of the reacting molecules present in the system at that moment. For simplicity, we assume that the number of RNAPs is constant. In the case of reactions (1) and (2), both k_t and λ_1 are set to 1/400 s [25], following measurements for the lar promoter. Meanwhile, G_{len} is determined by the length of

[3] note that symbol $*$ prefixing a species name in the above reactions means that the reactant is not consumed in the reaction. This is applied for simplicity to those reactants such as ribosomes, which exists in large amounts, and thus fluctuations in their numbers wont be significant in the propensity of reactions.

the gene, here set to 1000 nucleotides, and the time spent by the RNAp at each nucleotide, which follows an exponential distribution with a mean of 0.09 s [29].

In Prokaryotes, translation can begin as soon as the ribosome binding site (RBS) region of the RNA is completed. In reaction (3), a ribosome (Rib) binds to the RBS and translates the RNA. The RBS becomes available for more reactions after τ_1 s. The ribosome is released after $\Gamma(G_{len}, 0.06)$ seconds. The initiation rate, k_{tr} is set to 0.00042 s^{-1} [38]. Following measurements from *E. coli*, we have set $\tau_3 = 2$ s, and $\Gamma(G_{len}, 0.06)$ to follow a gamma distribution dependent on the gene's length, where each codon is added following an exponential distribution with a mean of 0.06 s [29]. Finally, $\Gamma(\tau_{5_{sh}}, \tau_{5_{sc}})$ is such that it accounts for the time that translation elongation takes, as well as the time it takes for a protein to fold and become active. In this case, we used the parameter values measured from GFP mutants commonly used to measure gene expression in *E. coli* [30]. Finally we consider also three additional reactions representing, respectively: RBS decay (equation 4) and promoter repression (equation (5)). Initially, the system has 1 promoter and 100 ribosomes. In the remainder of the paper we illustrate a thorough formal analysis of the above described single gene model by means of HASL model checking.

4 HASL model checking

The Hybrid Automata Stochastic Logic (HASL) [8] is a novel formalism widening the family of model checking approaches for stochastic models. Its main characteristics are as follows: first the class of models it addresses are the so-called Discrete Event Stochastic Processes (DESPs), a broad class of stochastic processes which includes, but (unlike most stochastic logics) is not limited to, CTMCs. Second the HASL logic turns out to be a powerful language through which temporal reasoning is naturally blended with elaborate reward-based analysis. In that respect HASL unifies the expressiveness of CSL[6] and its action-based [5], timed-automata [16, 14] and reward-based [22] extensions, in a single powerful formalism. Third HASL model checking belongs to the family of statistical model checking approaches (i.e. those that employ stochastic simulation as a means to estimate a model's property). More specifically HASL statistical model checking employs confidence-interval methods to estimate the expected value of random variables which may represent either a measure of probability or a generic real-valued measure. In the following we recall the basics of the HASL formalism i.e. the characterization of DESP and of HASL formula. We also quickly outline Cosmos [7] the HASL model checker we employed for analysing the models considered in this paper. For a comprehensive and more formal treatment of HASL we refer the reader to [8].

4.1 DESP

A DESP is a stochastic process consisting of a (possibly infinite) set S of states and whose dynamic is triggered by a (finite) set E of (time-consuming) discrete events. No restrictions are considered on the nature of the delay distribution associated with events, thus any distribution with non-negative support may be considered. For the sake of space in this paper we omit the formal definition of DESP and give an informal description of Generalised Stochastic Petri Nets (GSPNs) [2] the high-level language adopted to characterise DESP in the context of HASL model checking.

DESP in terms of Generalised Stochastic Petri Nets. Accordind to its definition the characterization of a DESP is a rather unpractical one, requiring an explicit listing of all of its elements (i.e. states, transitions, delay distributions, probability distribution governing concurrent events). However several high-level formalisms commonly used for representing Markov chain models (e.g. Stochastic Petri Nets, Stochastic Process Algebras), can straightforwardly be adapted to represent DESPs. In the context of HASL model checking we consider GSPNs as high level formalism for representing DESPs. The choice of GSPNs is due to two factors: (1) they allow a flexible modeling w.r.t. the policies defining the process (choice, service and memory) and (2) allow for efficient path generation (due the simplicity of the *firing rule* which drives their dynamics). We quickly recall the basics about GSPN models pointing out the correspondence with the various parts of a DESP. A GSPN model (e.g. Figure 1) is a bi-partite graph consisting of two classes of nodes, *places* (represented by circles) and *transitions* (represented by bars). Places may contain *tokens* (e.g. representing the number of molecules of a given species) while transitions (i.e. representing to the events) indicate how tokens "flow" within the net. The state of a GSPN consists of a *marking* indicating the distribution of tokens throughout the places. A transition is enabled whenever all of its *input places* contains a number of tokens greater than or equal to the multiplicity of the corresponding (input) arc. An enabled transition may *fire* consuming tokens (in a number indicated by the multiplicity of the corresponding input arcs) from all of its input places and producing tokens (in a number indicated by the multiplicity of the corresponding output arcs) in all of its output places. Transitions can be either *timed* (denoted by empty bars, if exponential, or gray bars if non-exponential) or *immediate* (denoted by black filled-in bars). Generally speaking transitions are characterized by: (1) a distribution which randomly determines the delay before firing it (corresponding to the DESP $delay()$ function); (2) a priority which *deterministically* selects among the transitions scheduled the soonest, the one to be fired; (3) a weight, that is used in the random choice between transitions scheduled the soonest with the same highest priority (corresponding to the DESP $choice()$ function). With the GSPN formalism [2] the delay of timed transitions is assumed *exponentially* distributed, whereas with GSPN-DESP it can be given by any distribution. Thus whether a GSPN timed-transition is characterized simply by its weight $t \equiv w$ ($w \in \mathbb{R}^+$ indicating an $Exp(w)$ distributed delay), a GSPN-DESP timed-transition is characterized by a triple: $t \equiv (\text{Dist-t}, \text{Dist-p}, w)$, where Dist-t indicates the type of distribution (e.g. Unif), dist-p indicates the parameters of the distribution (e.g $[\alpha, \beta]$) and $w \in \mathbb{R}^+$ is used to probabilistically choose between transitions occurring with equal delay[4]

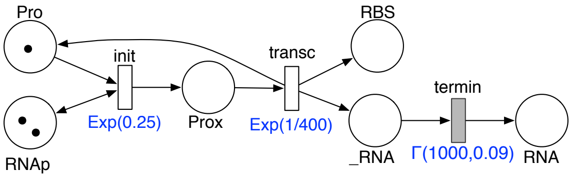


Fig. 1. Example of GSPN-DESP: model of reaction R_1 and R_2 of single-gene system

Example. The GSPN in Figure 1 encodes the transcription phase of the single-gene model (i.e. reaction R_1 and R_2 in Section 3.1). The net has: a place for each species involved in reactions R_1 and R_2 (i.e. Pro, RNAp, Prox, RBS, RNA) plus an extra place (i.e. _RNA) for capturing the intermediate delayed phase of RNA formation; three timed-transition corresponding to the delayed phases of reactions R_1 and R_2. Transitions *init* and *transc* are Exponentially distributed with rate $k_t = 0.25$, respectively $\lambda_1 = 1/400$. Transition *termin* is Gamma distributed (with parameters, $shape = 1000$ and $scale = 0.09$ as from experimental data) and represent the delayed termination of RNA formation as per R_2. In the initial marking $M_0 = (1, 2, 0, 0, 0, 0)$ we assume a molecule of Pro and two molecules of RNAp are available. Thus in state M_1 *init* is the only reaction enabled, and when it fires it will remove one token from both *Pro* and *RNAp* and add a token in each of its output places (i.e. RNAp and Prox), moving the state of the system in marking $M_1 = (0, 2, 1, 0, 0, 0)$ whereby the only enabled transition is *transc*, and so on.

4.2 Hybrid Automata Stochastic Logic

HASL is a logic designed to analyse properties of a DESP $\mathcal{D}$. A HASL formula is a pair $(\mathcal{A}, Z)$ where $\mathcal{A}$ is Linear Hybrid Automaton (i.e. a restriction of hybrid automata [3]) and Z is an expression involving *data variables* of $\mathcal{A}$. The goal of HASL model checking is to estimate the value of Z by synchronisation of the process $\mathcal{D}$ with the automaton $\mathcal{A}$. This is achieved through stochastic simulation of the synchronised process $(\mathcal{D} \times \mathcal{A})$, a procedure by means of which, infinite timed executions of process $\mathcal{D}$ are selected through automaton $\mathcal{A}$ until some final state is reached or the synchronisation fails. During such synchronisation, data variables evolve and the values they assume condition the evolution of the synchronisation. The synchronisation stops as soon as either: a final location of $\mathcal{A}$ is reached (in which case the values of the variables are considered in the estimate of Z), or the considered trace of $\mathcal{D}$ is rejected by $\mathcal{A}$ (in which case variables' values are discarded).

[4] a possible condition in case of non-continuous distributions

Synchronised Linear Hybrid Automata The first component of an HASL formula is an LHA, which is formally defined as follows:

Definition 1. *A* synch. LHA *is a tuple* $\mathcal{A} = \langle E, L, \Lambda, Init, Final, X, flow, \rightarrow\rangle$ *where:*

- $\boldsymbol{E}$*, a finite alphabet of events;*
- $\boldsymbol{L}$*, a finite set of locations;*
- $\boldsymbol{\Lambda : L \longrightarrow Prop}$*, a location labelling function;*
- $\boldsymbol{Init}$*, a subset of L called the initial locations;*
- $\boldsymbol{Final}$*, a subset of L called the final locations;*
- $\boldsymbol{X = (x_1, ...x_n)}$ *a n-tuple of data variables;*
- *flow* $\boldsymbol{: L \mapsto Ind^n}$ *a n-tuple of indicators representing the rate of evolution of each data variable in a location.*
- $\boldsymbol{\rightarrow \subseteq L \times \big((\mathsf{Const} \times 2^E) \uplus (\mathsf{lConst} \times \{\sharp\})\big) \times \mathsf{Up} \times L}$*, a set of edges*

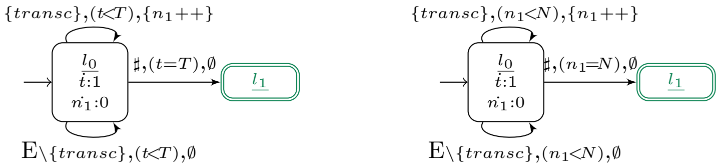
where an edge $(l, \gamma, E', U, l') \in\rightarrow$ (also denoted $l \xrightarrow{\gamma, E', U} l'$), consists of: a constraint γ (i.e. a boolean combination of inequalities of the form $\sum_{1\leq i\leq n} \alpha_i x_i + c \prec 0$ where $\alpha_i, c \in Ind$ are DESP indicators, $\prec \in \{=, <, >, \leq, \geq\}$ and $x_i \in X$; we denote Const the set of such constraints and $\mathsf{lConst} \subset \mathsf{Const}$ the set of left closed constraints, i.e. constraints giving rise to left-closed enabling intervals); a set E' of labels of synchronising events (including the extra label $\sharp$ denoting autonomous edges); a set U of *updates* (i.e. an n-tuple of functions $u_1, ..., u_n$ where each u_k is of the form $x_k = \sum_{1\leq i\leq n} \alpha_i x_i + c$ where the $\alpha_i, c \in Ind$ are DESP indicators; we denote Up the set of updates). by means of which new values are assigned to variables of X on traversing of the edge).

Edges labelled with a set of events in 2^E are called *synchronized* whereas those labelled with $\sharp$ are called *autonomous*. Furthermore we impose the following (informally described[5]) constraints for an automaton $\mathcal{A}$: (c1) only one initial location can be enabled; (c2) the same event cannot lead to different simultaneous synchronisations; (c3) two autonomous transition cannot be fireable simultaneously (c4) infinite loops without synchronisation are not possible. Informally the synchronisation between ($\mathcal{D}$ and $\mathcal{A}$) works as follows: a *synchronised* transition of the product process ($\mathcal{D} \times \mathcal{A}$) is triggered by the occurrence of a corresponding (time-consuming) event of the DESP, whereas an *autonomous* transition occurs (without synchronisation with a DESP event) as soon as the corresponding constraint is enabled (i.e. the variables of the LHA assume values fulfilling the constraint). Note that *autonomous* transitions may not consume time.

Example: Figure 2 depicts two variants of a simple two locations LHA defining measures of the gene-transcription toy model of Figure 1. Location l_0 is the initial location while l_1 the final location. The automaton employs two data-variables: t registering the simulation-time (hence with flow $\dot{t} = 1$ in every location) and n_1, an event counter (hence with flow $\dot{n_1} = 0$ in every location), counting the occurrences of transition *transcr*. The automaton has two synchronising edges (the

[5] for the formal characterisation see [8]

self-loops on l_0) and one autonomous edge (from l_0 to l_1). The top synchronising edge allows to increment the counter n_1 each time a transition *transcr* occurs



(a) time-bounded measures (b) event-bounded measures

Fig. 2. Example of LHA for simple properties of the GSPN-DESP model of Figure 1

whereas the bottom synchronising edge, simply reads in all other transitions occurrence without performing any update. The autonomous edge instead leads to acceptance location as soon as its constraint is fulfilled. The LHA in Figure 2(a) represents time-bounded measures as the constraints on the edges (and notably on the edge leading to the acceptance location) refers to the simulation time t, thus: as soon as $t = T$ the read in path is accepted. On the other hand the LHA in Figure 2(b) represents event-bounded measures accepting paths as soon as transition *transc* have occurred $n_1 = N$ times. In the following we provide few simple examples of relevant measures referred to the LHA in Figure 2 in terms HASL expressions.

HASL expressions The second component of an HASL formula is an expression, denoted Z and defined by the grammar:

$$\begin{aligned} Z &::= E(Y) \mid Z + Z \mid Z \times Z \\ Y &::= c \mid Y + Y \mid Y \times Y \mid Y/Y \mid last(y) \mid min(y) \mid max(y) \mid int(y) \mid avg(y) \\ y &::= c \mid x \mid y + y \mid y \times y \mid y/y \end{aligned} \tag{7}$$

y is an arithmetic expression built on top of LHA data variables (x) and constants (c). Y is a path dependent expression built on top of basic path random variables such as $last(y)$ (i.e. the last value of y along a synchronizing path), $min(y)(max(y))$ the minimum (maximum), value of y along a synchronizing path), $int(y)$ (i.e. the integral over time along a path) and $avg(y)$ (the average value of y along a path). Finally Z, the target measure of an HASL experiment, is an arithmetic expression built on top of the first moment of Y ($E[Y]$), and thus allowing to consider more complex measures including, e.g. $Var(Y) \equiv E[Y^2] - E[Y]^2$, $Covar(Y_1, Y_2) \equiv E[Y_1 \cdot Y_2] - E[Y_1] \cdot E[Y_2]$.

The COSMOS tool Assessment of HASL formulae against a DESP model is performed by means of the Cosmos [1, 7] model checker. Cosmos employs a confidence interval method to estimate the target expression Z. The desired accuracy of the target estimation is then set by the end user in terms of *confidence level* and *interval width*.

5 Analysis of SGN models through HASL

We illustrate the application of HASL model checking to the analysis of a model of gene expression with stochastic delayed dynamics. We (1) present the GSPN-DESP codification of the considered model; (2) introduce a number of relevant properties/measures of a model first describing them informally and then providing their encoding in HASL terms; (3) discuss results obtained by evaluation of the presented properties/measures by means of the Cosmos model checker.

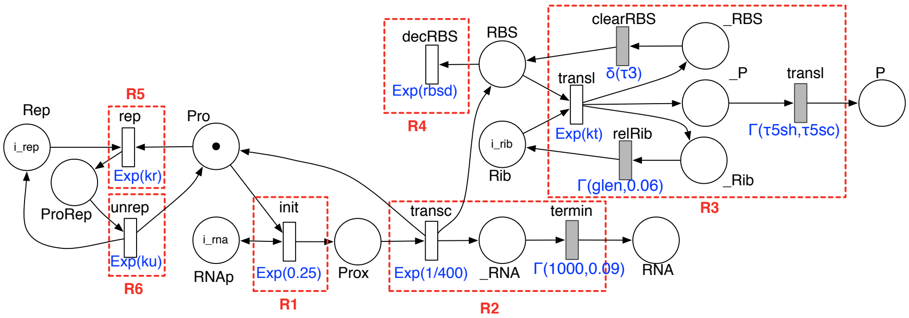


Fig. 3. GSPN model of Single Gene system with delayed stochastic dynamics

5.1 Single Gene model

The single-gene model described by equations (1) to (6) (Section 3.1) is encoded in GSPN-DESP terms by the net depicted in Figure 3. The net includes a place for each species of the model (i.e. Pro, RNAp, Prox, RNA, RBS, Rib P, Rep and ProRep) plus a number of auxiliary places representing intermediate stages of delayed reactions (i.e. _RNA, _RBS, _P, relRib). Initial marking of the net is set by means of parameters *i_rep*, *i_rnap*, *i_rib*, which correspond to the chosen initial population of the model (note that the promoter place, *Pro*, is initialized with one token, as each gene has one promoter region).

Reactions $\{R_1, \dots, R_6\}$ of the single gene model correspond to subnets (enclosed in red-dashed rectangles) in Figure 3. Each such subnet contains either a single exponentially-distributed transition (in case of reactions with non-delayed products i.e. R1, R4, R5, R6) or a combination of exponential and non-exponential transitions (in case of reactions with delayed products, i.e. R1 and R2). For example subnet R3 in Figure 3 represents the encoding of the *translation* reaction. It consists of: the translation-start event (i.e. exponentially distributed transition labeled *s_transl*; the RBS release event (i.e. deterministically distributed transition *clearRBS*); the ribosome releasing event (gamma distributed transition *relRib*); the protein production event (i.e. gamma distributed transition *prodP*). Observe that the effect of repressed gene-expression can be promptly analysed by setting of the repressor initial population (parameter *i_rep*= l_r): unrepressed configurations corresponds to $l_r = 0$, whereas $l_r > 0$ settings correspond to repressed model where the level of repression is proportional to $l_r > 0$.

	performance of TRANSCRIPTION and TRANSLATION mechanisms
ID	**description**
ϕ_{1_a}	average num. of completed-transcriptions (within T)
ϕ_{1_b}	average num. of completed-translations (within T)
ϕ_{2_a}	prob. density of the number of completed-transcriptions (within T)
ϕ_{2_b}	prob. density of the number of completed-translations (within T)
ϕ_{3_a}	cumulative prob. of the number of completed-transcriptions (within T)
ϕ_{3_b}	cumulative prob. of the number of completed-translations (within T)
	efficiency of TRANSLATION wrt TRANSCRIPTION
ID	**description**
ϕ_4	avg. num. of completed translations between two consecutive transcriptions
ϕ_5	prob. of at least N completed translations between two consecutive transcriptions
	REPRESSION related measures
ϕ_6	percentage of time gene is repressed
ϕ_7	how long does it take for translation to stop once a repression starts (i.e. sustainment of translation under repression)

Table 1. Properties of the Single Gene model

Properties of single gene model Table 1 depicts an excerpt of (informally stated) relevant measures of the single-gene model. They are grouped according to different aspects of gene-expression performance. The corresponding HASL encoding is given in Table 2. We briefly illustrate the automata of Table 2 and the associated HASL expressions:

$\boldsymbol{\mathcal{A}_1}$: it is designed for measures concerning the occurrences of *transc* and *transl* events. It accepts all paths of duration T and uses variables, n_1 and n_2 to maintain the number of *transc* and *transl* transitions occurred along a path. Different measures can be assessed through different HASL expressions referred to $\mathcal{A}_1$ including: $\phi_{1_a} = (\mathcal{A}_1, E[last(n_1)])$; $\phi_{1_b} = (\mathcal{A}_1, E[last(n_2)])$ and $\phi_4 = (\mathcal{A}_1, E[last(n_2)/last(n_1)])$ (see Table 1).

$\boldsymbol{\mathcal{A}_2}$: it measures the probability that the number of transcriptions is $n_1 = C$ (within time T). On acceptance (i.e. duration $t = T$), it distinguishes between paths such that $n_1 = C$ (in which case the bernoulli variable OK is set to 1), and paths such that $n_1 \neq C$ (i.e. OK is set to 0). The probability density of n_1 is assessed by re-iterated evaluations of formula $\phi_{2_a} = (\mathcal{A}_2, last(OK))$ corresponding to different values of C[6].

$\boldsymbol{\mathcal{A}_3}$: for measures concerning the amount of time gene is repressed. Apart from the usual global clock t it uses a timer t_r registering the time gene is repressed, hence it grows ($\dot{t}_r = 1$) only in location l_1 (i.e. repression is ON, corresponding to a marking of place $ProRep > 0$), while it is unchanged ($\dot{t}_r = 0$) in location l_0 (i.e. marking of place $ProRep = 0$). Also note that both l_0 and l_1 are initial locations, which is perfectly legal as their constraints make them mutually exclusive (this way $\mathcal{A}_3$ can be used to analyse both *repressed* and *unrepressed* configurations of the model).

$\boldsymbol{\mathcal{A}_4}$: it measures *"how likely it is that within a transcription interval (i.e. the interval between two occurrences of the* transc *event) at least N translations have been completed"*. It uses variables n_1 and n_2 (as above) and n_3 to count how many transcription intervals (along a path) contain $n_2 \geq N$ translations. The result is stored in $p_1 = n_3/n_1$ on acceptance. Note that, in this case, we consider an event-bounded observation window consisting of $n1 = N_1$ transcription events. Measure ϕ_5 (Table 1) in HASL terms is $\phi_5 = (\mathcal{A}_4, last(p_1))$.

$\boldsymbol{\mathcal{A}_5}$: it is designed for measures of *sustainment of translation activity under repression* (i.e. ϕ_7 in Table 1). It uses the following variables: n_o counting the number of repression intervals (interval between two repression events) in which translation arrested; t_o: measuring the *translation time-to-arrest* in a repression interval (given that translation arrested); T_o timer measuring the cumulated t_o. Note that translation arrest corresponds to the absence of tokens in all translation related places of the GSPN model (Figure 3), corresponding to condition: $(RBS{=}0 \wedge \_RBS{=}0 \wedge \_P{=}0 \wedge \_Rib{=}0)$. Locations l_0, l_1 and l_2 are then associated to the following state conditions of the model: *repression is off* (l_0), *repression is on and translation off* (l_1) and *repression is on and translation ongoing* (l_2). All paths of duration $t{=}T$ are accepted and the target measure[7] is be obtained through expression $Z = E[last(T_o)/Last(n_o)]$.

Remark. A formal assessment of HASL expressiveness is beyond the scope of this paper however we make some considerations in that respect. The peculiarity of HASL based reasoning is that any combination of state and/or transition and/or reward conditions may be employed to characterise the paths of interests. This is the main difference with other stochastic logics, ranging from those limited to state-based reasoning (e.g. CSL, BLTL), to those featuring state/action based reasoning but not supporting rewards (e.g. asCSL [5]), up to the timed-automata ones which mix state/action-based reasoning with (multiple) time-bounding [16, 14]. So far reward-based analysis has been added to logics fea-

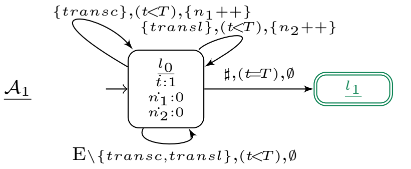
[6] note that simple variants of $\mathcal{A}_2$ can be used to assess the PDF of translations and the CDFs of both transcription and translation.

[7] note that with a time-bounded measurement, as with $\mathcal{A}_5$, measuring may stop in any instant (not necessarily at the end) of a repression interval: this is not a problem as T_o and n_o are updated only when translation arrests, thus if bound T is reached before translation arrests, measure T_o/n_o will correctly refer to the duration of translation sustainment over all completed repression cycles

legend of used variables:

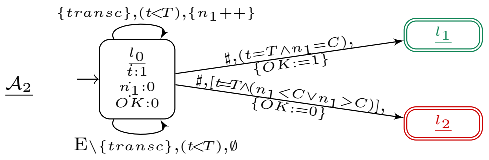
t:total time n_1:completed transcriptions n_2:completed translations
t_r:repressed time n_3:♯ of transcr-intervals such that $n_2 > N$ OK:bernoulli variable
t_o:sustain time (inter-repression) n_o:♯ repress-intervals with died off transl
p_1:% of transcr-intervals such that $n_2 > N$
T_o:total sustain time

Expected completed transcriptions within T


$\{transc\},(t<T),\{n_1{+}{+}\}$
$\{transl\},(t<T),\{n_2{+}{+}\}$
$\mathcal{A}_1$
l_0
$\dot{t}$:1
$\dot{n_1}$:0
$\dot{n_2}$:0
♯,$(t{=}T)$,∅
l_1
E\{transc,transl},$(t<T)$,∅

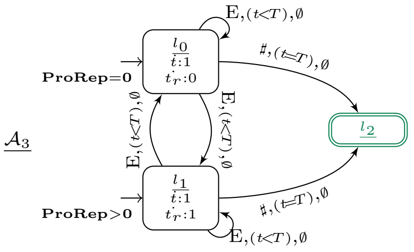
$\phi_1 \equiv (\mathcal{A}_1, E[last(n_1)])$

PDF of completed transcriptions within T


$\{transc\},(t<T),\{n_1{+}{+}\}$
$\mathcal{A}_2$
l_0
$\dot{t}$:1
$\dot{n_1}$:0
$\dot{OK}$:0
♯,$(t{=}T \wedge n_1{=}C)$, $\{OK{:=}1\}$
♯,$[t{=}T \wedge (n_1{<}C \vee n_1{>}C)]$, $\{OK{:=}0\}$
l_1
l_2
E\{transc},$(t<T)$,∅

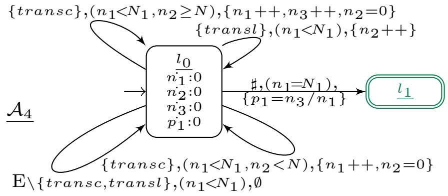
$\phi_{2_a} \equiv (\mathcal{A}_2, E[last(OK)])$

proportion of repressed time within T



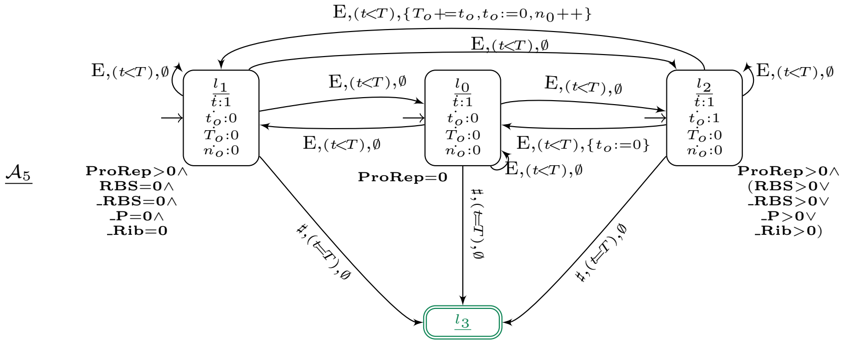
$\phi_6 \equiv (\mathcal{A}_3, E[(100/T) * last(t_r))])$

prob. of at least N translations between 2 consecutive transcriptions for first N_1 transcriptions


$\{transc\},(n_1{<}N_1, n_2{\geq}N),\{n_1{+}{+}, n_3{+}{+}, n_2{=}0\}$
$\{transl\},(n_1{<}N_1),\{n_2{+}{+}\}$
$\mathcal{A}_4$
l_0
$\dot{n_1}$:0
$\dot{n_2}$:0
$\dot{n_3}$:0
$\dot{p_1}$:0
♯,$(n_1{=}N_1)$, $\{p_1{=}n_3/n_1\}$
l_1
$\{transc\},(n_1{<}N_1, n_2{<}N),\{n_1{+}{+}, n_2{=}0\}$
E\{transc,transl},$(n_1{<}N_1)$,∅

$\phi_5 \equiv (\mathcal{A}_4, E[last(p_1)])$

sustainment of translation under repression

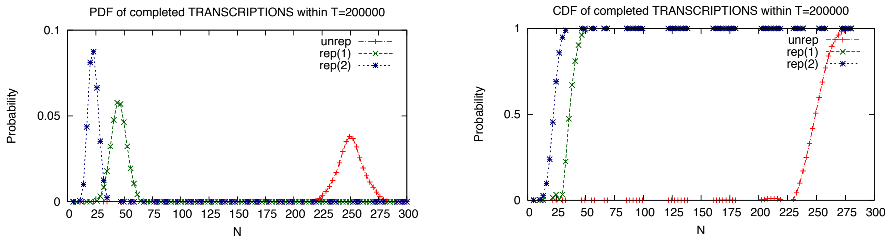


$\phi_7 \equiv (\mathcal{A}_6, E[last(T_o)/last(n_o)])$

Table 2. LHA for various measures of the Single Gene model

turing state-based reasoning. For example the rewards enriched version of CSL supported by PRISM [27] allows for considering multiple (state and transition) reward structures and to assess reward measures wrt paths of a given CTMC model. However, even with the addition of rewards, CSL remains a language limited to state-based temporal reasoning, thus, differently from HASL, reward values do not play an active role in characterising relevant paths. As a consequence several measures that can easily be expressed with HASL, do not always have an equivalent in CSL terms (or if they do they require hard wiring of extra information in the original CTMC). For example, measuring the PDF (and CDF) of an event occurrences, is easily done with HASL (e.g. LHA $\mathcal{A}_2$, of Table 2), whereas cannot be naturally achieved with CSL, unless states of the original CTMC are enriched with variables counting the occurrences of relevant events. Similarly, more complex measures involving combination of elaborate conditions on rewards values (i.e. LHA variables) as the factors characterising the selected paths (e.g. those corresponding to automata $\mathcal{A}_4$ and $\mathcal{A}_5$, in Table 2) seem not to be expressible through CSL rewards.

Experiments. We assessed the previously described HASL measures through experiments executed with the Cosmos model checker. For time-bounded measures we have considered (following [33]) $T = 2{\cdot}10^5$ as time horizon which roughly corresponds to 60 cell cycles, considering an average a cell cycle period of about 55 minutes (i.e. 3300s) in the case of E. coli. All experiments have been run with the following setting concerning confidence interval estimation: confidence-level: 99.99%; interval-width: 0.01.

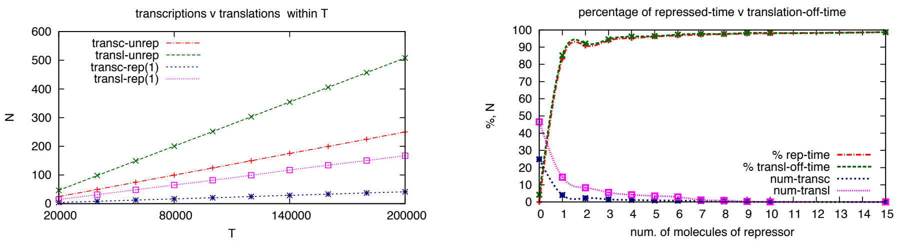


(a) PDF (b) CDF

Fig. 4. PDF and CDF of completed transcriptions within T

Experiment 1. Figure 4 compares plots of the PDF (Figure 4(a)) and CDF (Figure 4(b)) of random variable n_1: *num. of completed transcription within* T (query ϕ_2 and ϕ_3) of *unrepressed* vs *repressed* configurations (i.e. *rep(1)*, corresponding to initial marking $i\_rep = 1$ and *rep(2)*, corresponding to initial marking $i\_rep = 2$). The effect of repression is evident as the bell-shaped probability density of n_1 is shifted toward lower values for increasing level of repression.

Experiment 2. Figure 5(a) compares the *expected number of completed* transcriptions *vs.* translations *within* T in function of time for unrepressed and repressed (*rep(1)*) configurations. Observe that the throughput of *translation* is roughly twice as much as that of *transcriptions*, (both in unrepressed condition, as well as, in presence of repression). This is due to the rates of RNA degradation and translation initiation.



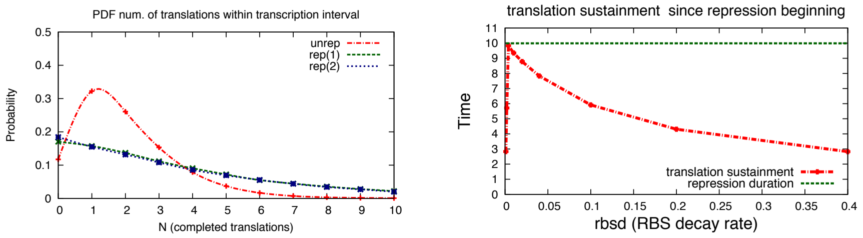
(a) unrep v rep transc and transl

(b) % rep-time v transc and transl

Fig. 5. Exp. transcriptions and translations and percentage of repressed time

Experiment 3. Figure 5(b), plots two measures of timing: *the percentage of time gene is repressed* ($\mathcal{A}_3$) and the *percentage of time no translation activity is going on* (variant of $\mathcal{A}_3$) when system is observed for duration T and in function of the level of repression (num. of repressor molecules on the x-axis). To observe also the trend of transcription and translation activity in function of repression level Figure 5(b) also includes two curves referred to the expected number of *transcriptions*, respectively *translations*, within T. Observe that the presence of a single repressor is sufficient for the gene to remain repressed for 83% of the time and, likewise, for translation to be non-existing for 85% of the observation time (whereas in absence of repressor, translation activity is only non-existing for about 4% of the time).

Experiment 4. Figure 6(a) compares the PDFs of random variable n_2: *num. of completed transcription within a transcription interval* (i.e. within two consecutive transcription completions) (query $\phi_4 : (\mathcal{A}_4, Last(p_1))$. This is computed for the unrepressed model and for two configurations of the repressed model (*rep(1)* and (*rep(2)*). Outcomes indicate that in presence of repression the probability density is more "distributed", than the bell shaped one corresponding to the unrepressed configuration. Furthermore increasing the level of repression seems to have no effect on the probability density (plots *rep(1)* and (*rep(2)* are essentially identical).



(a) PDF num-transl in transcr-interval (b) translation sustainment

Fig. 6. Measure related to translation activity

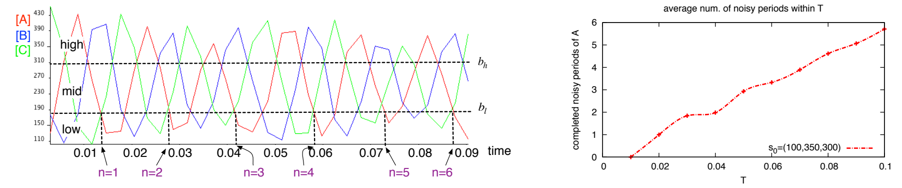
Experiment 5. Figure 6(b) refers to measurement of the *translation sustainment within a repression-interval* (query $\phi_5 : (\mathcal{A}_5, Last(n_{off})/Last(n_{rep}))$ in function of the RBS decay rate (*rbsd*). We varied *rbsd* in the interval $[0.001, 4]$ which includes $rbsd = 0.01$ i.e. the value complying with experimental evidence used in the“standard” model's configuration. Obtained results indicate, quite sensibly, that translation sustainment is inversely proportional to RBS decay. It should be noted that with $rbsd < 0.004$ the translation sustainment is actually increasing with *rbsd* (not very evident in plot of Figure 6(b)). This is because, by definition, query $\phi_5 : (\mathcal{A}_5, Last(n_{off})/Last(n_{rep}))$ measures the sustainment of translation *on condition that sustainment lasts lesser than repression.* With $rbsd < 0.004$, however, decay is so slow that with high probability sustainment lasts longer than repression, while with low probability it lasts less. In this case ($rbsd < 0.004$) it is sensible that the average value of (low-likely) *translation sustainment not exceeding repression duration* increases with *rbsd*.

6 Measuring oscillations with HASL

Oscillatory trends are fundamental aspects of the dynamics of many biological mechanisms, therefore the ability to detect/measure oscillations in biological models is crucial. CSL-based characterisation of oscillations in CTMC models of biochemical reactions have been considered in [9], with limited success, and more comprehensively in [36]. Here we show preliminary results concerning the application of HASL to the analysis of oscillations. Let σ be a (infinitely long) simulation trace of an n-dimensional DESP model whose states' form is $s = (s_1, \ldots s_n) \in \mathbb{N}^n$, with s_i being the value along the ith dimension (e.g the number of molecules of species i). Let $\sigma_i(t)$ denote the i-projection of $\sigma(t)$ the state σ is at time t. Following the characterisation given in [36] σ_i can either be: *convergent* (i.e. tending to a finite value), *divergent* (i.e. tending to infinity) or *oscillating* (i.e. the lack of the previous two). Furthermore σ_i is *periodic* with period δ iff $\forall t, \sigma_i(t) = \sigma_i(t + \delta)$. Thus σ is periodic oscillatory along the i-th dimension iff σ_i is both oscillating and periodic. Here, instead, we focus on a less restrictive characterisation of oscillatory trends namely that of *noisy periodicity* [36]. Given an upper and a lower bound $b_h, b_l \in \mathbb{N}$ ($b_h > b_l$), inducing intervals $low = (-\infty, b_l]$, $mid = (b_l, b_h)$ and $high = [b_h - \infty)$ (i.e. l,m and h), trace σ_i is said *noisy periodic* iff it perpetually switches from low to $high$ (passing through mid) and returning to low.Note that such trends corresponds to the following regular expression: $e_{np} = l(l)^*m(ll^*m)^*h(mm^*h)^*m(mh^*m)^*l$.

We illustrate preliminary results about application of HASL to oscillations analysis by means of a simple example. Reactions (8) represent, the so called, 3-way doped oscillator, a systems consisting of three species A, B and C which oscillate perpetually. Note that A, B and C form a loop of of dependency whereby A is converted into B, which, in turns, is converted into C, which, in turns, is converted into A. D_A, D_B and D_C, are auxiliary species representing doping substances which guarantees the *liveness* of the conversion loop. It can be easily shown, (e.g. by application of stochastic simulation), that species A, B and C oscillate (Figure 7(a)) with amplitude, period and "noisiness" dependent on the initial population (a_0, b_0, c_0) (by default we assume the population of doping species to be 1).

$$\begin{array}{ccc} A + B \xrightarrow{r_A} 2B & B + C \xrightarrow{r_B} 2C & C + A \xrightarrow{r_C} 2A \\ D_A + C \xrightarrow{r_C} A + D_A & D_B + A \xrightarrow{r_A} B + D_B & D_C + B \xrightarrow{r_B} C + D_C \end{array} \tag{8}$$



(a) simulation trace of doped oscillator init-population:$(100, 350, 310)$

(b) number of periods in doped oscillator

Fig. 7. Number of periods in a simulated trace of the 3-ways doped oscillator (left) and corresponding average number of noisy periods calculated (in function of time) through the HASL query $\phi_{np} = (\mathcal{A}_{np}, E(last(n))$ (right) with bounds set to $b_l = 180$ and $b_h = 300$.

An LHA to measure noisy periodic traces: the LHA $\mathcal{A}_{np}$ in Figure 8 is designed to measure the number of noisy periods of amplitude $a \geq (b_h - b_l)$, where b_l and b_h are the bounds inducing the *low*, *mid* and *high* (as above). It consists of three locations l_0, l_1 and l_2, associated to the *low* (i.e. $A \leq b_l$), respectively the *mid* (i.e. $b_l < A < b_h$) and the *high* (i.e. $A \geq b_h$) interval.
It uses three variables: t (total time), n to count the completed noisy periods (i.e. corresponding to regular expression $e_{np} = l(l)^*m(ll^*m)^*h(mm^*h)^*m(mh^*m)^*l$), and *top*, a boolean flag used to condition the increase of n on completion of a noisy period (i.e. *top* is set to 1 entering the *high* interval and n is incremented on entering the *low* interval only if $top = 1$, in which case *top* is reset). The

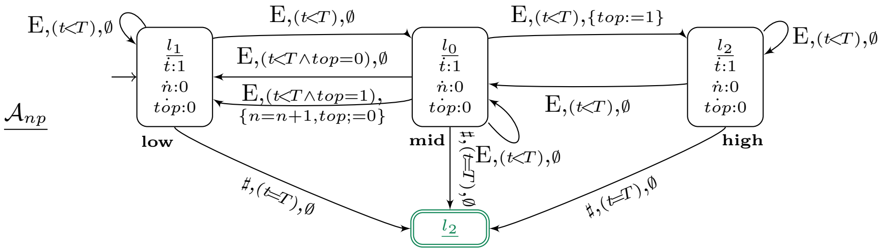


Fig. 8. An LHA for measuring the number of periods of a *periodic noisy* oscillatory trace of the 3-way oscillator

average number of completed noisy periods with time T can be assessed by means of HASL formula $\phi_{np} = (\mathcal{A}_{np}, E(last(n))$. Figure 7(b) depicts the outcome of assessment of ϕ_{np} in function of time bound T. Such results are in good agreement with simulated traces, a sample of which is shown in Figure 7(a).

7 Conclusion

We presented some insights on the application of HASL statistical model checking to the analysis of (non-necessarily Markovian) stochastic models of biological systems. The most important feature of HASL model checking lays in its expressive power: by employing LHA as machinery to characterise relevant trajectories of a model it is possible to identify/assess elaborate measures which may not be naturally accounted for with more popular stochastic logic, i.e. those featuring state-based temporal reasoning, such as, for example, CSL and BLTL. We demonstrated (part of) the potential of the HASL language by developing and assessing a number of properties of a model of single-gene network with delayed non-Markovian dynamic. Although such model is a rather simple, it served well to our aim, which was to show the potential of HASL based analysis. Furthermore we presented preliminary insights on the application of HASL to the analysis of oscillatory trends. Future developments include: 1) the application of HASL approach to more complex systems, such as, for example a model of the P53-Mdm2 feedback loop with stochastic dynamics previously analysed with the GNSim simulator but not yet formally model checked. 2) further development of HASL based oscillation analysis (i.e. measurements of frequency, amplitude of oscillatory trends). The main difficulty of the HASL approach is the technicality of the formalism itself. In particular, specifying an HASL property boils down to specifying an automata, something which may be far from intuitive for non-expert users. Thus, a further direction of development regards working on the definition of a more intuitive property specification language with an associated translator to LHA specifications.

References

1. COSMOS home page. http://www.lsv.ens-cachan.fr/software/cosmos/.
2. M. Ajmone Marsan, G. Balbo, G. Conte, S. Donatelli, and G. Franceschinis. *Modelling with Generalized Stochastic Petri Nets.* John Wiley & Sons, 1995.
3. R. Alur, C. Courcoubetis, T. A. Henzinger, and P.-H. Ho. Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems. In *Hybrid Systems*, LNCS 736, 1992.
4. A. P. Arkin, J. Ross, and H. H. McAdams. Stochastic Kinetic Analysis of a Developmental Pathway Bifurcation in Phage-l Escherichia coli. *Genetics*, 149(4):1633–1648, 1998.
5. C. Baier, L. Cloth, B. Haverkort, M. Kuntz, and M. Siegle. Model checking action- and state-labelled Markov chains. *IEEE Trans. on Software Eng.*, 33(4), 2007.
6. C. Baier, B. Haverkort, H. Hermanns, and J.-P. Katoen. Model-checking algorithms for CTMCs. *IEEE Trans. on Software Eng.*, 29(6), 2003.
7. P. Ballarini, H. Djafri, M. Duflot, S. Haddad, and N. Pekergin. COSMOS: a statistical model checker for the hybrid automata stochastic logic. In *Proceedings of the 8th International Conference on Quantitative Evaluation of Systems (QEST'11)*, pages 143–144. IEEE Computer Society Press, sep. 2011.
8. P. Ballarini, H. Djafri, M. Duflot, S. Haddad, and N. Pekergin. HASL: an expressive language for statistical verification of stochastic models. In *Proc. Valuetools*, 2011.
9. P. Ballarini and M.L. Guerriero. Query-based verification of qualitative trends and oscillations in biochemical systems. *Theoretical Computer Science*, 411(20):2019 – 2036, 2010.
10. J. Barnat, L. Brim, I. Cerná, S. Drazan, J. Fabriková, J. Láník, D. Safránek, and H. Ma. Biodivine: A framework for parallel analysis of biological models. In *COMPMOD*, volume 6 of *EPTCS*, pages 31–45, 2009.
11. Manuel Barrio, Kevin Burrage, Andre Leier, and Tianhai Tian. Oscillatory regulation of hes1: Discrete stochastic delay modelling and simulation. *PLoS Computational Biology*, 2(9):1017–1030, 2006.
12. G. Bernot, J-P Comet, A. Richard, and J. Guespin. Application of formal methods to biological regulatory networks: extending Thomas' asynchronous logical approach with temporal logic. *Journal of Theoretical Biology*, 229(3):339–347, August 2004.
13. D. Bratsun, D. Volfson, L. S. Tsimring, and J. Hasty. Delay-induced stochastic oscillations in gene regulation. *Proc Natl Acad Sci U S A*, 102(41):14593–8, 2005.
14. T. Chen, T. Han, J.-P. Katoen, and A. Mereacre. Quantitative model checking of CTMC against timed automata specifications. In *Proc. LICS'09*, 2009.
15. H. De Jong, J. Geiselmann, C. Hernandez, and M. Page. Genetic network analyzer: qualitative simulation of genetic regulatory networks. *Bioinformatics*, 19(3):336–244, 2003.
16. S. Donatelli, S. Haddad, and J. Sproston. Model checking timed and stochastic properties with CSL^{TA}. *IEEE Trans. on Software Eng.*, 35, 2009.
17. E. A. Emerson and E. M. Clarke. Characterizing correctness properties of parallel programs using fixpoints. In *Proc. of the 7th Coll. on Automata, Languages and Programming*, LNCS 85, 1980.
18. F. Fages and S. Soliman. Formal cell biology in biocham. In *Proc. of 8th Int. Conf. on Formal methods for computational systems biology*, SFM'08, 2008.
19. F. Fages, S. Soliman, and Chabrier N. Rivier. Modelling and querying interaction networks in the biochemical abstract machine BIOCHAM. *Journal of Biological Physics and Chemistry*, 4(2):64–73, 2004.

20. D.T. Gillespie. Exact stochastic simulation of coupled chemical reactions. *Journal of Physical Chemistry*, 81(25):2340–2361, 1977.
21. H. A. Hansson and B. Jonsson. A framework for reasoning about time and reliability. In *Proc. 10th IEEE Real -Time Systems Symposium*, pages 102–111, Santa Monica, Ca., 1989. IEEE Computer Society Press.
22. B.R. Haverkort, L. Cloth, H. Hermanns, J-P. Katoen, and C. Baier. Model checking performability properties. In *Proc. DSN'02*, 2002.
23. J. Heath, M. Kwiatkowska, G. Norman, D. Parker, and O. Tymchyshyn. Probabilistic model checking of complex biological pathways. *Theoretical Computer Science*, 319(3):239–257, 2008.
24. S. K. Jha, E. M. Clarke, C. J. Langmead, A. Legay, A. Platzer, and P. Zuliani. A bayesian approach to model checking biological systems. In *Proc. of the 7th Int. Conference on Computational Methods in Systems Biology*, pages 218–234. Springer-Verlag, 2009.
25. M Kandhavelu, A Hakkinen, O Yli-Harja, and A S Ribeiro. Single-molecule dynamics of transcription of the lar promoter. *Phys Biol*, 9(2), 2012.
26. H. Kitano. *Foundations of Systems Biology.* MIT Press, 2002.
27. M. Kwiatkowska, G. Norman, and D. Parker. Stochastic model checking. In *Formal Methods for the Design of Computer, Communication and Software Systems: Performance Evaluation*, volume 4486 of *LNCS*, pages 220–270. Springer, 2007.
28. 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*, 2011. To appear.
29. J. Makela, J. Lloyd-Price, O. Yli-Harja, and A.S. Ribeiro. Stochastic sequence-level model of coupled transcription and translation in prokaryotes. *BMC Bioinformatics*, 12(1):121, 2011.
30. J.A. Megerle, G. Fritz, U Gerland, K. Jung, and J.O. Rädler. Timing and Dynamics of Single Cell Gene Expression in the Arabinose Utilization System. *Biophysical Journal*, 95:2103–2115, 2008.
31. Prism home page. http://www.prismmodelchecker.org.
32. A. Ribeiro, R. Zhu, and S. A. Kauffman. A general modeling strategy for gene regulatory networks with stochastic dynamics. *Journal of computational biology : a journal of computational molecular cell biology*, 13(9):1630–1639, November 2006.
33. Andre S. Ribeiro and Jason Lloyd-Price. SGN Sim, a stochastic genetic networks simulator. *Bioinformatics (Oxford, England)*, 23(6):777–779, March 2007.
34. Marc R Roussel and Rui Zhu. Validation of an algorithm for delay stochastic simulation of transcription and translation in prokaryotic gene expression. *Physical Biology*, 3(4):274–284, 2006.
35. M Schwarick, C Rohr, and M Heiner. Marcie - model checking and reachability analysis done efficiently. In *Proc. 8th International Conference on Quantitative Evaluation of SysTems (QEST 2011)*, pages 91–100. IEEE CS Press, 2011.
36. D. Spieler. Model checking of oscillatory and noisy periodic behavior in markovian population models. Master's thesis, Saarland University, 2009.
37. Y. Taniguchi, P. J. Choi, G-W Li, H. Chen, M. Babu, J. Hearn, A. Emili, and X.S. Xie. Quantifying E. coli Proteome and Transcriptome with Single-Molecule Sensitivity in Single Cells. *Science*, 329(5991):533–538, July 2010.
38. R. Zhu, A.S. Ribeiro, D. Salahub, and S.A. Kauffman. Studying genetic regulatory networks at the molecular level: Delayed reaction stochastic models. *Journal of Theoretical Biology*, 246(4):725–745, 2007.