Statistical Model Checking in *BioLab* : Applications to the automated analysis of T-Cell Receptor Signaling Pathway *

Edmund M. Clarke[1], James R. Faeder[2], Christopher J. Langmead[1], Leonard A. Harris[2], Sumit Kumar Jha[1], and Axel Legay[1]

[1] Computer Science Department, Carnegie Mellon University, Pittsburgh PA
[2] Department of Computational Biology , University of Pittsburgh School of Medicine, Pittsburgh PA

Abstract. We present an algorithm, called BioLab, for verifying temporal properties of rule-based models of cellular signalling networks. BioLab models are encoded in the BioNetGen language, and properties are expressed as formulae in probabilistic bounded linear temporal logic. Temporal logic is a formalism for representing and reasoning about propositions qualified in terms of time. Properties are then verified using sequential hypothesis testing on executions generated using stochastic simulation. BioLab is optimal, in the sense that it generates the minimum number of executions necessary to verify the given property. BioLab also provides guarantees on the probability of it generating Type-I (i.e., false-positive) and Type-II (i.e., false-negative) errors. Moreover, these error bounds are pre-specified by the user. We demonstrate BioLab by verifying stochastic effects and bistability in the dynamics of the T-cell receptor signaling network.

1 Introduction

Computational modeling is an effective means for gaining insights into the dynamics of complex biological systems. However, there are times when the nature of the model itself presents a barrier to such discovery. Models with stochastic dynamics, for example, can be difficult to interpret because they are inherently non-deterministic. In the presence of non-deterministic behavior, it becomes non-trivial to determine whether a behavior observed in a simulation is typical, or an anomaly. In this paper, we introduce a new tool, called BioLab, for *formally* reasoning about the behavior of stochastic dynamic models by integrating techniques from the field of *Model Checking* [8] into the BioNetGen [12, 13] framework for rule-based modeling. We then use BioLab to verify the stochastic bistability of T-cell signalling.

* This research was sponsored by the Gigascale Systems Research Center (GSRC), the Semiconductor Research Corporation (SRC), the Office of Naval Research (ONR), the Naval Research Laboratory (NRL), the Army Research Office (ARO), and the General Motors Lab at CMU awards to E.M.C., as well as the U.S. Department of Energy Career Award (DE-FG02-05ER25696), a Pittsburgh Life-Sciences Greenhouse Young Pioneer Award to C.J.L., National Institutes of Health grant GM76570 and a B.A.E.F grant.

The term "model checking" refers to a family of automated techniques for formally verifying properties of complex systems. Since its inception in 1981, the field of Model Checking has made substantial contributions in industrial settings, where it is the preferred method for formal verification of circuit designs. Briefly, the system is first encoded as a model in a formal description language. Next, properties of interest (e.g., absence of deadlock) are expressed as formulae in temporal logic. Temporal logic is a formalism for representing and reasoning about propositions qualified in terms of time. Given a model, $\mathcal{M}$, a set of initial states, S_0, and a property, ϕ, a model checking algorithm automatically determines whether the model satisfies the formula.

Historically, model checking has most often been applied to engineered systems, and thus the majority of Model Checking algorithms are designed for such systems. Recently, however, there has been growing interest in the application of model checking to biology (e.g.,[5, 6, 19, 21, 22]). Biological systems present new challenges in the context of formal verification. In particular, biological systems tend to give rise to highly parameterized models with stochastic dynamics. Biologists are generally interested in determining whether a given property is (or is not) sensitive to a plausible set of initial conditions and parameter values. Model checking algorithms targeting biological applications must therefore apply to stochastic, multi-parameter models.

BioLab models stochastic biochemical systems using the BioNetGen modeling language. The set of initial states (i.e., S_0) comprise a user-specified set of initial conditions and parameter values. Properties are expressed in probabilistic bounded linear temporal logic. BioLab then statistically verifies the property using sequential hypothesis testing on executions sampled from the model. These samples are generated using variants of Gillespie's algorithm [15, 11, 31], which ensures that the executions are drawn from the "correct" underlying probability distribution. This, combined with the use of sequential hypothesis testing provides several guarantees. First, BioLab can bound the probability of Type-I (i.e., false-positive) and Type-II (i.e., false-negative) errors, with regard to the predictions it makes. These error bounds are specified by the user. Second, BioLab is optimal in the sense that it generates the minimum number of executions necessary to determine whether a given property is satisfied. The number of required executions varies depending on the behavior of the model and is determined dynamically, as the program is running.

The contributions of this paper are as follows: (i) Our method is the first application of statistical model checking to rule-based modeling of biochemical systems. (ii) Our algorithm provides guarantees in terms of optimality, as well as bounds on the probability of generating Type-I and Type-II errors. (iii) We verify that a stochastic model of T-cell receptor signaling exhibits behaviors that are qualitatively different from those seen in an ordinary differential equation model of the same system [23]. In particular we verify that stochastic effects induce switching between two stable steady states of the system.

2 BioNetGen

Proteins in cellular regulatory systems, because of their multicomponent composition, can interact in a combinatorial number of ways to generate myriad protein complexes, which are highly dynamic [17]. Protein-protein interactions and other types of interactions that occur in biochemical systems can be modeled by formulating rules for each type of chemical transformation mediated by the interactions [18]. The rules can be viewed as definitions of reaction classes and used as generators of reactions, which describe the transformations of molecules in the system possessing particular properties. The assumption underlying this modeling approach, which is consistent with the modularity of regulatory proteins [27], is that interactions are governed by local context that can be captured in simple rules. Rules can be used to generate reaction networks that account comprehensively for the consequences of protein-protein interactions. Examples of rule-based models of specific systems can be found in [16, 3, 1, 26].

BioNetGen is a software package that provides tools and a language for rule-based modeling of biochemical systems [2, 13]. A formal description of the language and underlying graph theory is provided in [4]. BioNetGen is similar to the κ-calculus, which has also been developed as a language for rule-based modeling of biochemical systems [9]. Other tools for rule-based modeling are reviewed in [18].

The syntax and semantics of BioNetGen have been thoroughly described in [13]. Briefly, a BioNetGen model is comprised of six basic elements that are defined in separate blocks in the input file: *parameters*, *molecule types*, *seed species*, *reaction rules*, *observables*, and *actions*. Molecules are the basic building blocks of a BioNetGen model, and are used to represent proteins and other structured biological molecules, such as metabolites, genes, or lipids. The optional *molecule types* block is used to defined the composition and allowable states of molecules. Molecules may contain components, which represent the functional elements of molecules, and may bind other components, either in the same molecule or another molecule. Components may be associated with state variables, which take on a finite set of possible values that may represent conformational or chemical states of a component, e.g., tyrosine phosphorylation. An example of a molecule type declaration is

```
TCR(ab,ITAM~U~P~PP,lck,shp)
```

which is used to define the structure of the T cell receptor in the model presented in Sec. 5.2. The name of the molecule type is given first, followed by a comma-separated list of its components in parenthesis. Any declared component may participate in a bond. In addition, the allowed values of the state variable associated with a component are indicated with $\sim$ followed by a name. In the above example, a molecule of type TCR has four components, three of which (ab, lck, and shp) may be used only for binding and one of which (ITAM) has an associated state variable that takes on the values U, P, or PP—representing the unphosphorylated, phosphorylated, and doubly phosphorylated forms respectively.

The *seed species* block defines the molecules and molecular complexes that are initially present in the system with an optional quantifier. Depending on the semantics used in the simulation of the model (see below) the value of the quantifier may be either continuous or restricted to discrete values. For example, the line

```
Lck(tcr,Y~U,S~U)  LCK
```

in the seed species block specifies that the initial amount of the species comprised of a molecule of Lck with both its Y (tyrosine residue 394) and S (serine residue 453) components in the U (unphosphorylated) state is given by the parameter LCK, which is defined in the *parameters* block. Only species with a non-zero initial amount as declared in the *seed species* block are present in the system at the beginning of simulation.

The *reaction rules* block contains rules that define how molecules in the system can interact. A rule is comprised, in order of appearance, of a set of reactant patterns, a transformation arrow, a set of product patterns, and a rate law. A pattern is a set of molecules that select a set of species through a mapping operation [4]. The match of a molecule in a pattern to a molecule in a particular species depends only on the components that are specified in the pattern (which may include wildcards), so that one pattern may select many different species. Three basic types of operations are carried out by the rules in the T cell model: binding (unbinding) of two molecules through a specified pair of components and changing the state variable of a component. An example of a binding rule is

```
TCR(ab,shp)+pMHC(p~ag) -> TCR(ab!1,shp).pMHC(p~ag!1) b1
```

which specifies that *any* TCR molecule containing unbound ab and shp components may bind through its ab component to a p component in the ag state of a pMHC molecule. In this example, the first reactant pattern, `TCR(ab,shp)`, matches any TCR-containing species with free ab and shp components, independent of the state of the remaining two components. The `+` operator separates two reactant patterns that must map to distinct species. The transformation arrow may be either unidirectional (`->`), as in the above rule, or bidirectional (`<->`), indicating that the rule is to be applied in both the forward and reverse directions (i.e., switching the reactant and product patterns). The product patterns define the configuration of the selected reactant molecules following the application of the rule. Here, the ab component of TCR is bound to the p component of pMHC by the addition of an edge labeled `1`, as indicated by the two bond labels (`!1`) in the products. The parameter b1 specifies the rate constant to be used in determining the rate of the reaction, which is computed as a product of the rate constant and each of the reactant amounts.

The *observables* block contains definitions of model outputs, which are defined as sums over the amounts of species matched by a set of patterns. The output of the TCR model is the level of doubly-phosphorylated ERK, which is specified by the following line in the BNGL file

```
Molecules ppERK ERK(S~PP)
```

where the first item is a keyword defining the type of observable, the second item is the name of the observable, and the final item is a list of patterns that determines the matching species.

The *actions* block specify the operations that are to be carried out either to generate or simulate a network. As we now discuss, the choice of operations to perform also defines the semantics under which the model elements are interpreted. BioNetGen uses three basic methods to simulate the time course of observables for a rule-based network: generate-first (GF), on-the-fly (OTF), and network-free (NF). These methods are described in detail in [13]; we provide a brief overview in this section. In GF, rules are iteratively applied to the initial set of seed species until all reachable species and reactions are generated or some other stopping criterion is satisfied. The resulting network can be simulated either by solving a set of ODE's for the average concentration of each species in the system under the influence of the mass action reactions (GF-ODE) or by Gillespie's stochastic simulation algorithm (SSA) [15] to sample the exact solution to the chemical master equations governing the species probabilities (GF-SSA). Both methods generate traces[1] of the species concentrations as a function of time, but the GF-ODE algorithm is deterministic for a given initial state and set of system parameters, whereas each simulation run of GF-SSA from a given initial state represents a stochastic process and may generate a different trace. Like GF-SSA, OTF uses the Gillespie algorithm to generate traces but only generates species and reactions that are reachable within a small number of specified time steps [24, 11]. OTF was originally proposed as a way to maintain computational efficiency for large reaction networks, but is not practical for rule-based models that include oligomerization or attempt a comprehensive description of reaction networks [18, 10]. The NF method [10, 31] avoids explicit generation of species and reactions by simulating molecules as agents and has been shown to have per event cost that is independent of the number of possible species or reactions [10]. NF also relies on the SSA to sample reaction events that govern the evolution of the molecular agents. Because species are not explicitly tracked, the NF method generates traces over observables rather than individual species. This restriction is not an issue for applications to biology because the concentrations of individual species are typically not observable in biological experiments.

3 Model Checking for Stochastic Systems

The following section introduces the concept of *statistical model checking*. We assume the reader is familiar with basic concepts in probability theory.

3.1 The Problem

We use $Pr(E)$ to denote the probability of the event E to occur. We consider a system $\mathcal{M}$ whose executions (sequences of states of the system) are *observable* and a property ϕ that is defined as a set of executions. We assume that one can decide whether an execution trace of $\mathcal{M}$ satisfies ϕ, i.e. whether the execution

[1] The term "trace" is equivalent to the term "execution". From now, we will use "trace" when we want to emphasize that we are talking about a BioNetGen Model

belongs to ϕ. In this paper, the *probabilistic model checking problem* consists in deciding whether the executions of $\mathcal{M}$ satisfy ϕ with a probability greater than or equal to a given threshold θ. The latter is denoted by $\mathcal{M} \models Pr_{\geq\theta}(\phi)$. This statement only makes sense if one can define a probability space on the executions of the system as well as on the set of executions that do satisfy ϕ.

The probabilistic model checking problem can be solved with a *probabilistic model checking algorithm*. Such an algorithm is *numerical* in the sense that it computes the exact probability for the system to satisfy ϕ and then compares it with the value of θ. Successful probabilistic model checking algorithms [7, 20]) have been proposed for various classes of systems, including (continuous time) Markov chains and Markov Decision Processes. The drawback with those approaches is that they compute the probability for all the executions of the system, which may not scale up for systems of large size.

Another way to solve the probabilistic model checking problem is to use a *statistical model checking algorithm*. In the rest of this section, we recap the statistical model checking algorithmic scheme proposed by Younes in [32].

3.2 Statistical Approach

The approach in [32] is based on hypothesis testing. The idea is to check the property ϕ on a sample set of simulations and to decide whether the system satisfies $Pr_{\geq\theta}(\phi)$ based on the number of executions for which ϕ holds compared to the total number of executions in the sample set. With such an approach, we do not need to consider all the executions of the system. To determine whether $\mathcal{M}$ satisfies ϕ with a probability $p \geq \theta$, we can test the hypothesis $H : p \geq \theta$ against $K : p < \theta$. A test-based solution does not guarantee a correct result but it is possible to bound the probability of making an error. The *strength* (α, β) of a test is determined by two parameters, α and β, such that the probability of accepting K (respectively, H) when H (respectively, K) holds, called a Type-I error (respectively, a Type-II error) is less or equal to α (respectively, β).

A test has *ideal performance* if the probability of the Type-I error (respectively, Type-II error) is exactly α (respectively, β). However, these requirements make it impossible to ensure a low probability for both types of errors simultaneously (see [32] for details). A solution to this problem is to relax the test by working with an *indifference region* (p_1, p_0) with $p_0 \geq p_1$ ($p_0 - p_1$ is the *size of the region*). In this context, we test the hypothesis $H_0 : p \geq p_0$ against $H_1 : p \leq p_1$ instead of H against K. If both the values of p and θ are between p_1 and p_0 (the indifference region), then we say that the probability is sufficiently close to θ so that we are indifferent with respect to which of the two hypotheses K or H is accepted.

3.3 An Algorithmic Scheme

Younes proposed a procedure to test $H_0 : p \geq p_0$ against $H_1 : p \leq p_1$ that is based on the *sequential probability ratio test* proposed by Wald [30]. The approach is briefly described below.

Let B_i be a discrete random variable with a Bernoulli distribution. Such a variable can only take 2 values 0 and 1 with $Pr[B_i = 1] = p$ and $Pr[B_i = 0] =$

$1-p$. In our context, each variable B_i is associated with one simulation of the system. The outcome for B_i, denoted b_i, is 1 if the simulation satisfies ϕ and 0 otherwise. In the sequential probability ratio test, one has to choose two values A and B, with $A > B$. These two values should be chosen to ensure that the strength of the test is respected. Let m be the number of observations that have been made so far. The test is based on the following quotient:

$$\frac{p_{1m}}{p_{0m}} = \prod_{i=1}^{m} \frac{Pr(B_i = b_i \mid p = p_1)}{Pr(B_i = b_i \mid p = p_0)} = \frac{p_1^{d_m}(1-p_1)^{m-d_m}}{p_0^{d_m}(1-p_0)^{m-d_m}}, \tag{1}$$

where $d_m = \sum_{i=1}^{m} b_i$. The idea behind the test is to accept H_0 if $\frac{p_{1m}}{p_{0m}} \geq A$, and H_1 if $\frac{p_{1m}}{p_{0m}} \leq B$. An algorithm for sequential ratio testing consists of computing $\frac{p_{1m}}{p_{0m}}$ for successive values of m until either H_0 or H_1 is satisfied. This has the advantage of minimizing the number of simulations. In each step i, the algorithm has to check the property on a single execution of the system, which is handled with a new Bernoulli variable B_i whose realization is b_i. In his thesis [32], Younes proposed a logarithmic based algorithm (Algorithm 2.3 page 27) SPRT that given p_0, p_1, α and β implements the sequential ratio testing procedure. Computing ideal values A_{id} and B_{id} for A and B in order to make sure that we are working with a test of strength (α, β) is a laborious procedure (see Section 3.4 of [30]). In his seminal paper [30], Wald showed that if one defines $A_{id} \geq A = \frac{(1-\beta)}{\alpha}$ and $B_{id} \leq B = \frac{\beta}{(1-\alpha)}$, then we obtain a new test whose strength is (α', β'), but such that $\alpha' + \beta' \leq \alpha + \beta$, meaning that either $\alpha' \leq \alpha$ or $\beta' \leq \beta$. In practice, we often find that both inequalities hold.

The SPRT algorithm can be extended to handle Boolean combinations of probabilistic properties as well as much more complicated probabilistic Model checking problems than the one considered in this paper [32].

Statistical Model Checker The SPRT algorithm can be implemented in order to solve the probabilistic model checking problem for a specific class of systems and a specific class of properties. For this, we have to implement :

- *A simulator* that is able to simulate the system and produce observable executions without necessarily constructing its entire state-space.
- *An execution verifier* that is a procedure to decide whether an execution satisfies a given property.

In section 5, we propose BioLab, which is an implementation of the SPRT algorithm for models encoded and simulated using BioNetGen.

4 Statistical Model Checking for Continuous-time Markov Chains

A BioNetGen model can be interpreted as Continuous-time Markov Chain (CTMC), which may be simulated using the stochastic simulation methods described in Sec. 2. In this section, we review CTMCs and then introduce the

probabilistic bounded linear temporal logic, which will is used in BioLab to define properties over CTMCs.

4.1 Continuous-time Markov Chains

Let $\mathbb{R}$ (resp. $\mathbb{N}$) denote the set of real (resp. natural) numbers and let $\mathbb{R}_{\geq 0}$ and $\mathbb{R}_{>0}$ denote the set of non-negative and strictly-positive real numbers, respectively. $\mathbb{N}$ is the set of natural numbers, and $\mathbb{N}_{\geq 0}$ is the set of strictly positive natural numbers. We now recall the definition of *Structured Continuous-time Markov Chains*.

Definition 1. *A Structured Continuous-time Markov Chain is a tuple* $\mathcal{M} = (S, S_0, R, SV, V)$*, where*

- S *is a finite set of states;*
- $s_0 \in S$ *is the initial state;*
- $R : S \times S \to R_{\geq 0}$ *is the rate matrix.*
- SV *is a* finite set of state variables *defined over* $\mathbb{R}_{\geq 0}$*. These variables represent the concentration of each molecular species in the model.*
- $V : S \times SV \to \mathbb{R}_{\geq 0}$ *is a* value assignment function *providing the value of* $x \in SV$ *in state* s*.*

Let $\mathcal{M} = (S, S_0, R)$ be a structured continuous-time Markov chain. Let $t \in \mathbb{R}_{>0}$ and $s_1, s_2 \in S$, the probability to go from s_1 to s_2 within t time unit is defined as follows

$$P(s_1, s_2, t) = \frac{R(s_1, s_2)}{E(s_1)}(1 - e^{E(s_1)t}), \tag{2}$$

where $E(s_1) = \sum_{s' \in S} R(s, s')$.

An *execution*, also called *trace*, of $\mathcal{M}$ is a possibly infinite sequence $\sigma = (s_0, t_0)$ $(s_1, t_1)(s_2, t_2) \ldots$ such that for each $i \geq 0$, (1) $p(s_i, s_{i+1}, t_i) > 0$, and (2) $t_i \in R_{>0}$. Given (s_i, t_i), t_i is the time that is spent in state s_i. Given s_i, $\sum_{j<i} t_j$ is the number of time units spent before reaching s_i. We use $\sigma(i)$ (with $i \geq 0$) and σ^i to reference the $i-$th state of the execution and the suffix of the execution starting from the pair (s_i, t_i), respectively. Given a set $S' \in S$, we will use $\mathrm{Path}(S')$ to denote the set of all the executions whose initial states are in S'.

4.2 Probabilistic Bounded Linear Temporal Logic

BioLab is intended to be used as a tool for verifying properties of executions of CTMCs. Users specify properties of interest in the probabilistic bounded linear temporal logic. We now give the syntax and the semantics of bounded linear temporal logic (BLTL).

Let SV be a set of non negative real variables and $\sim \in \{\geq, \leq, = \}$. A Boolean predicate over SV is a constraint of the form $x \sim v$, where $x \in SV$ and $v \in \mathbb{R}_{\geq 0}$. A BLTL property is built on a finite set of Boolean predicates over SV using Boolean connectives and temporal operators. The syntax of the logic is given by the following grammar :

$\phi ::= x \sim v \,|\, (\phi_1 \vee \phi_2) \,|\, (\phi_1 \wedge \phi_2) \,|\, \neg\phi \,|\, \mathbf{X}(\phi) \,|\, (\phi_\mathbf{1} \vee \phi_\mathbf{2}) \,|\, (\phi_1 \wedge \phi_2) \,|\, (\phi_1 \mathbf{U^t} \phi_\mathbf{2}) \,|\, (\phi_1 \widetilde{\mathbf{U}}^t \phi_2) \,|\, \mathbf{D_t}(\phi)$.

The operators $\neg$, $\vee$, and $\wedge$ are the normal *logic operators*, which are read "not", "or", and "and", respectively. The operators $\mathbf{X}$, $\mathbf{U^t}$, $\widetilde{\mathbf{U}}$, and $\mathbf{D}$ are the *temporal operators*. The operator $\mathbf{X}$ is read "next", and corresponds to the notion of "in the next state". The operator $\mathbf{U^t}$ is read "until t time units have passed", and requires that its first argument be true *until* its second argument is true, which is required to happen within t time units. The operator $\widetilde{\mathbf{U}}$ is read "release", and requires that its second argument is true during the first t time units unless this obligation has been released by its first argument becoming true. The operator $\mathbf{D_t}$ is read "dwell", and requires that each time the argument becomes true, it is falsified in within t time units.

Two additional temporal operators are in very common use. The first of them is $\mathbf{F^t}$, where $\mathbf{F}$ is read "eventually". The eventually operator requires that its argument becomes true within t units of time. Formally, we have $\mathbf{F^t}\psi = \mathbf{True}\,\mathbf{U^t}\,\psi$. The second operator is $\mathbf{G^t}$, where $\mathbf{G}$ is read "always". This operator requires that its argument stays true during at least t units of time. Formally, we have $\mathbf{G^t}\psi = \mathbf{False}\,\widetilde{\mathbf{U}}^\mathbf{t}\,\psi$.

The semantics of BLTL was informally described above. We now present its formal semantics. The fact that the execution $\sigma = (s_0, t_0)(s_1, t_1), \ldots$ satisfies the BLTL property ϕ is denoted by $\sigma \models \phi$. We have the following:

- $\sigma \models x \sim v$ if and only if $V(\sigma(0), x) \sim v$;
- $\sigma \models \phi_1 \vee \phi_2$ if and only if $\sigma \models \phi_1$ or $\sigma \models \phi_2$;
- $\sigma \models \phi_1 \wedge \phi_2$ if and only if $\sigma \models \phi_1$ and $\sigma \models \phi_2$;
- $\sigma \models \neg\phi$ if and only if $\sigma \not\models \phi$.
- $\sigma \models \mathbf{X}\phi$ if and only if $\sigma^1 \models \phi$.
- $\sigma \models \phi_1 \mathbf{U}^t \phi_2$ if and only if there exists $i \in \mathbb{N}$ such that (1) $\sigma^i \models \phi_2$ and (2) $\sum_{j<i} t_j \leq t$, and for each $0 \leq j < i$ $\sigma^j \models \phi_1$.
- $\sigma \models \phi_1 \widetilde{\mathbf{U}}^t \phi_2$ if and only if for each i such that $\sigma^i \not\models \phi_2$ and $\sum_{m<i} t_m \leq t$, there exists $0 \leq j < i$ such that $\sigma^j \models \phi_1$.
- $\sigma \models \mathbf{D_t}(\phi)$ if and only if for each state $\sigma(i)$ such that $\sigma(i) \models \phi$, there exists $j > i$ such that $\sigma(j) \not\models \phi$ and $\sum_{m=i}^{m=j-1} t_m \leq t$.

Remark 1. It should be noted that we can decide whether an infinite execution satisfies a BLTL property by observing one of its finite prefixes.

We assume that properties of Structured Continuous-time Markov Chains are specified with *Probabilistic Bounded Linear Temporal Logic* (BTL).

Definition 2. *A BTL property is a property of the form $\psi = Pr_{\geq\theta}(\phi)$, where ϕ is a BLTL property.*

We say that the Continuous-time Markov Chain $\mathcal{M}$ satisfies ψ, denoted by $\mathcal{M} \models \psi$, if and only if the probability for an execution of $\mathcal{M}$ to satisfy ϕ is greater than θ. The problem is well-defined since, as it is shown with the following theorem, one can always assign a unique probability measure to the set of executions that satisfy an BLTL property.

Theorem 1. *Let $\mathcal{M}$ be a Continuous-time Markov Chain and ϕ be a BLTL formula. One can always associate a unique probability measure to the set of executions of $\mathcal{M}$ that satisfy ϕ.*

5 On the use of Statistical Model Checking to Analyze a T Cell Model

5.1 The BioLab Algorithm

The BioLab algorithm is a statistical model checker that implements the SPRT algorithm introduced in Section 3.3 for checking BTL properties against BioNetGen models. BioLab uses the BioNetGen simulation engine described in Section 2 to generate traces by randomly simulating biological models, and then uses a *Bounded Linear Temporal Logic trace verifier* to validate the generated traces against the BLTL part of the given BTL property. Depending on the result of the validation of the generated tracess, the BioLab tool decides whether the BTL formula is satisfied/falsified or if more samples are needed in order to make this decision. The structure of the BioLab algorithm is outlined in Figure 5.1. The BioNetGen simulator is used to generate stochastic traces and the trace verifier verifies each of them against the BLTL property. Our *trace verifier* is based on the translation from BLTL to alternating automata [29, 14]. The statistical model checker continues to simulate the BioNetGen model until a decision about the property has been made.

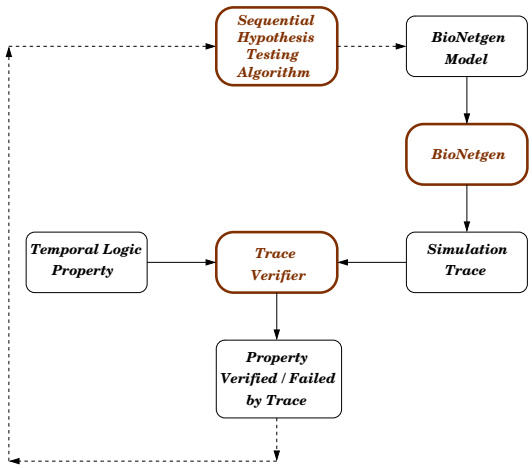

Fig. 1. Architecture of BioLab.

5.2 The T Cell Receptor Model

T lymphocytes, also known as T cells, play a central role in the immune system by detecting foreign substances, known as antigens, and coordinating the immune response. T cells detect the presence of antigen through surface receptors, called T cell receptors (TCRs), which bind to specific polypeptide fragments that are displayed on the surface of neighboring cells by a protein called the major histocompatibility complex (MHC). Variable regions of the immunoglobulin chains that comprise the TCR give rise to a broad range of TCR binding specificities. Individual T cells (or clonal populations derived from the same precursor) express a unique form of TCR. Processes of positive and negative selection during maturation of T cells in the thymus select T cells possessing TCRs with a weak but nonzero affinity for binding MHC molecules carrying peptides derived from host proteins. High-affinity binding between TCR and peptide-MHC (pMHC) complexes induces a cascade of biochemical events that leads to activation of the T cell and initiation of an immune response. To be effective in detecting antigens while avoiding autoimmunity, T cells must generate strong responses to the presence of minute quantities of antigen—as low as a few peptide fragments per antigen-presenting cell—while not responding to the large quantities of endogenous (host) pMHC expressed on all cells. The T cell appears to maintain this delicate balancie between sensitivity and selectivity through a combination of mechanisms that include kinetic proofreading, which discriminates against pMHC-receptor interactions that are too short, positive feedback, which amplifies the response and makes it more switch-like, and negative feedback, which acts in concert with kinetic proofreading to dampen responses to weak stimulation and with positive feedback to enhance the stability of the inactive state.

A computational model incorporating all three of these mechanisms has recently been developed by Lipniacki et al. [23], and serves as the basis for the experiments we conduct here using BioLab. This model extends previous simplified models of kinetic proofreading [25] and feedback regulation [28] by incorporating mechanistic detail about the involvement of specific signaling molecules. A schematic illustration of the model is presented in Fig. 5.2. Binding of pMHC to the TCR initiates a series of binding and phosphorylation events at the receptor that can lead either to activation or inhibition of the receptor depending on the strength of the stimulus, which is indicated along the kinetic proofreading axis. The rectangular box in the figure represents the TCR complex, which requires three components to make its passage to the activated form. These components are pMHC (P), doubly phosphorylated receptor (T_{pp}), and singly-phosphorylated LCK (L_p). In its active form, the LCK kinase can phosphorylate SHP (S) to produce S_p, which acts as a negative feedback by reversing TCR activation events and blocking TCR activation. LCK also acts through a series of intermediate layers to activate the MAP kinase ERK, a potent activator of transcription, whose active form (E_{pp}) is taken as the final readout of T cell activation. As shown in the figure, activated ERK also provides positive feedback by blocking the activity of S_p.

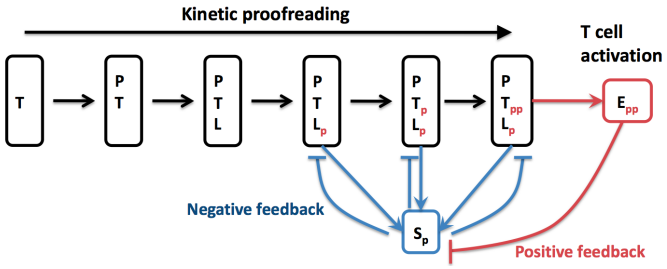

Fig. 2. Overview of the TCR signaling model of Ref. [23]. Lines terminated by flat heads indicate an inhibitory interaction.

This model captures three important properties of T cell activation, which are sensitivity to small numbers of pMHC with high binding affinity, high selectivity between pMHCs of different affinity, and antagonism, the inhibition of response by pMHC of intermediate affinity. Because only small numbers of high-affinity pMHC ligands are displayed on cell surfaces, stochastic effects have a major influence on the dynamics both of the model and of the initiation of signaling through the TCR. The model also exhibits bistable ERK responses over a broad range of pMHC number and binding affinity. This bistable regime has the interesting property that stochastic trajectories may exhibit completely different dynamics from the deterministic trajectory from the same initial state, and even the average behavior of stochastic trajectories may differ qualitatively from the deterministic behavior (see Fig. 7B of [23] for an example). This divergence between the stochastic and deterministic dynamics was the motivation for using this model of TCR as the basis for the current study, which aims to show that formal verification methods can be useful for the characterization of rule-based biochemical models.

The TCR model has been encoded in the BioNetGen language (available at `http://bionetgen.org/index.php/Tcr_tomek`) and serves as the basis for the current experiments. The BioNetGen model is comprised of seven molecule types and 30 rules, which generate a biochemical network of 37 species and 97 reactions. The main output of the model is fraction of ERK that is doubly phosphorylated, denoted by the variable f, which is taken as a measure of T cell activation. For $f < 0.10$ cell is considered inactive, for $f > 0.5$ cell is considered active. The response is observed to be switch-like with respect to stimulation strength, measured by the number of agonist (high affinity) pMHC per cell, given by N_1 (see Fig. 2 of [23]). The system also exhibits bistability with respect to f over a wide range of N_1 values (see Fig. 7A of [23]). As shown in Fig. 5.2, under many input conditions traces from stochastic simulations may sample both stable steady states and thus diverge from deterministic traces starting from the same initial conditions, which sample only a single steady state.

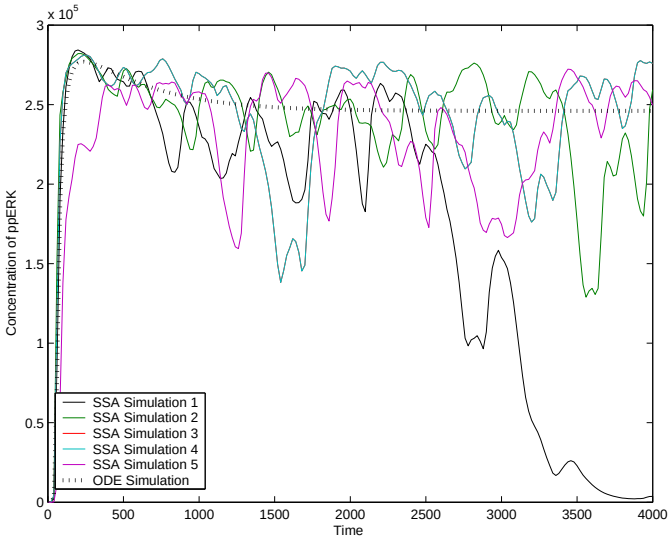

Fig. 3. Traces from deterministic (ODE) and stochastic simulation of the TCR signaling model. $N_1 : 100$, $N_2 : 3000$.

5.3 Experiments

We performed several *in-silico* BioLab experiments on the T Cell Receptor model. Each of our experiments was performed on a cluster of 40 3GHz computational nodes communicating using the Message Passing Interface.

Property 1 In our first experiment, we were interested in the truth of the hypothesis that the fraction f of doubly phosphorylated ERK stays below a given threshold value with a given probability during the first 300 seconds of simulation. We verified the following property with various values of the probability p and the threshold value γ.

$$Pr_{\geq p} \; (\; \mathbf{G^{300}}(\; ppERK \; / \; totalERK < \gamma \;) \;)$$

The first model we analyzed started with 100 molecules of agonist pMHC ($N_1 = 100$) while antagonist pMHC was absent ($N_2 = 0$). We also set the dissociation constant of agonist pMHC as 1/20 per second. The results of our experiment are shown in Table 1.

In our second experimental setup, our system started with 100 molecules of agonist pMHC while there were 3000 molecules of antagonist pMHC. We also set the dissociation constant of agonist pMHC as 1/20 per second and that of antagonist pMHC as 1/3. The results of our experiment are shown in Table 2.

In the third experiment, there were 100 molecules of agonist pMHC and 1000 molecules of antagonist pMHC. We set the dissociation constant of agonist

Sl. No.	p_1	p_0	γ	Result	Total Number of Samples	Number of Successful Samples	Time
1	0.90	0.95	0.1	No	40	0	232.73
2	0.90	0.95	0.5	No	40	0	221.38
3	0.90	0.95	0.7	No	40	2	221.41
4	0.90	0.95	0.9	No	40	2	233.45
5	0.90	0.95	0.95	Yes	240	236	1162.87

Table 1. N_1 : 100, N_2 : 0 , Type-I and Type II error : 0.001

Sl. No.	p_1	p_0	γ	Result	Total Number of Samples	Number of Successful Samples	Time
1	0.90	0.95	0.1	No	40	24	54.25
2	0.90	0.95	0.5	Yes	120	97	168.22
3	0.90	0.95	0.7	Yes	240	237	320.30
4	0.90	0.95	0.9	Yes	200	199	263.87
4	0.90	0.95	0.95	Yes	400	385	533.39

Table 2. N_1 : 100, N_2 : 3000 , Type-I and Type II error : 0.001

Sl. No.	p_1	p_0	γ	Result	Total Number of Samples	Number of Successful Samples	Time
1	0.90	0.95	0.1	No	40	4	41.56
2	0.90	0.95	0.5	No	40	14	67.40
3	0.90	0.95	0.7	Yes	200	199	317.90
4	0.90	0.95	0.9	Yes	200	200	278.08
5	0.90	0.95	0.95	Yes	480	459	777.17

Table 3. N_1 : 100, N_2 : 1000 , Type-I and Type II error : 0.001

pMHC as 1/20 per second and that of antagonist pMHC as 1/3. We summarize the results in Table 3.

The fourth experiment started with 100 molecules of agonist pMHC and 300 molecules of antagonist pMHC. We used the same dissociation constants as in previous experiment. The results are presented in Table 4.

Sl. No.	p_1	p_0	γ	Result	Total Number of Samples	Number of Successful Samples	Time
1	0.90	0.95	0.1	No	40	0	96.02
2	0.90	0.95	0.5	No	40	4	108.65
3	0.90	0.95	0.7	No	40	13	89.89
4	0.90	0.95	0.9	No	160	130	322.50
5	0.90	0.95	0.95	Yes	320	312	866.65

Table 4. N_1 : 100, N_2 : 300 , Type-I and Type II error : 0.001

The fraction of phosphorylated ERK in the first and the fourth experiments exceeded 0.9 within the first 300 seconds with at least 90% probability. This phenomenon was not observed in the second and the third experiments.

Property 2 In our second experiment, we were interested in the truth of the hypothesis that the system can go from the inactive state to the active state. We verified the following property with various values of the probability p.

$$Pr_{\geq p}(\mathbf{F^{300}}(ppERK/totalERK{<}0.1 \wedge \mathbf{F^{300}}\ (ppERK/totalERK{>}0.5)))$$

Our first model started with 100 molecules of agonist pMHC (with dissociation constant 1/20 per second) while antagonist pMHC was assumed to be absent in the initial state. The results are presented in Table 5.

Sl.	p_1	p_0	Result	Total Number of Samples	Number of Successful Samples	Time
1	0.90	0.95	Yes	160	160	412.25
2	0.70	0.75	Yes	120	120	309.58
3	0.50	0.55	Yes	80	80	214.74
4	0.20	0.25	Yes	40	40	88.32
5	0.10	0.15	Yes	40	40	98.84

Table 5. N_1 : 100, N_2 : 0 , Type-I and Type II error : 0.001

We note that the number of samples needed to decide the property depends both upon the fraction of samples that satisfied the property and the probability with which we want the property to be satisfied.

In our second experimental setup, our system started with 100 molecules of agonist pMHC while there were 3000 molecules of antagonist pMHC. We also set the dissociation constant of agonist pMHC as 1/20 per second and that of antagonist pMHC as 1/3. We present the results in Table 6.

Sl.	p_1	p_0	Result	Total Number of Samples	Number of Successful Samples	Time
1	0.90	0.95	No	40	0	24.92
2	0.70	0.75	No	40	0	27.05
3	0.50	0.55	No	80	0	52.19
4	0.20	0.25	No	120	0	86.30
5	0.10	0.15	No	160	0	108.25

Table 6. N_1 : 100, N_2 : 3000 , Type-I and Type II error : 0.001

In the third experiment, there were 100 molecules of agonist pMHC and 1000 molecules of antagonist pMHC. We also set the dissociation constant of agonist pMHC as 1/20 per second and that of antagonist pMHC as 1/3. The results are illustrated in Table 7.

Sl.	p_1	p_0	Result	Total Number of Samples	Number of Successful Samples	Time
1	0.90	0.95	No	40	27	35.16
2	0.70	0.75	Yes	40	34	34.34
3	0.50	0.55	Yes	120	109	111.30
4	0.20	0.25	Yes	40	37	44.57
5	0.10	0.15	Yes	40	36	45.54

Table 7. N_1 : 100, N_2 : 1000 , Type-I and Type II error : 0.001

The fourth experiment started with 100 molecules of agonist pMHC and 300 molecules of antagonist pMHC. We used the same dissociation constants as in previous experiment. The outcome of the experiments are shown in Table 8.

The second model showed a qualitative difference in behavior from the other three models while quantitative differences in behavior can be seen among all the four models. We verified our hypothesis that the stochastic model of the T Cell

Sl.	p_1	p_0	Result	Total Number of Samples	Number of Successful Samples	Time
1	0.90	0.95	Yes	160	160	346.92
2	0.70	0.75	Yes	120	116	226.08
3	0.50	0.55	Yes	80	80	168.87
4	0.20	0.25	Yes	40	40	81.10
5	0.10	0.15	Yes	40	40	73.11

Table 8. N_1 : 100, N_2 : 300 , Type-I and Type II error : 0.001

Receptor pathway can go from the inactive to the active state with a non-zero probability.

Property 3 In our third set of experiments, we were interested in the truth of the hypothesis that the system can go from the active state to the inactive state. We verified the following property with various values of the probability p.

$$Pr_{\geq p}(\mathbf{F^{300}}(ppERK/totalERK{>}0.5 \wedge \mathbf{F^{300}}\ (ppERK/totalERK{<}0.1)))$$

Our model started with 100 molecules of agonist pMHC (with dissociation constant 1/20 per second) while there was no antagonist pMHC. The results of our experiments are illustrated in Table 9.

Sl.	p_1	p_0	Result	Total Number of Samples	Number of Successful Samples	Time
1	0.90	0.95	No	40	0	107.25
2	0.70	0.75	No	40	0	106.95
3	0.50	0.55	No	80	0	218.42
4	0.20	0.25	No	120	0	168.98
5	0.10	0.15	No	160	0	330.80

Table 9. N_1 : 100, N_2 : 0 , Type-I and Type II error : 0.001

Our second model started with 100 molecules of agonist pMHC (with dissociation constant 1/20 per second) while there were 1000 antagonist pMHC (with dissociation constant 1/3 per second). The results of our experiments are illustrated in Table 10.

Property 4 In our fourth set of experiments, we were interested in asking the question if the system spent more than a certain threshold of time in a given state before leaving that state. We verified the following property with various values of the probability p.

Sl.	p_1	p_0	Result	Total Number of Samples	Number of Successful Samples	Time
1	0.90	0.95	No	120	79	57.97
2	0.70	0.75	No	280	160	114.62
3	0.50	0.55	No	160	51	66.04
4	0.20	0.25	Yes	120	73	50.06
5	0.10	0.15	Yes	40	21	19.53

Table 10. N_1 : 100, N_2 : 1000 , Type-I and Type II error : 0.001

$$Pr_{\geq p} \left(\mathbf{D_{100}} \left(ppERK \ / \ totalERK > 0.5 \right) \right)$$

The model we analyzed started with 100 molecules of agonist pMHC (with dissociation constant 1/20 per second) while antagonist pMHC was absent. The results of our analysis are presented in Table 11.

Sl.	p_1	p_0	Result	Total Number of Samples	Number of Successful Samples	Time
1	0.90	0.95	Yes	160	160	216.21
2	0.70	0.75	Yes	120	120	160.32
3	0.50	0.55	Yes	80	80	109.11
4	0.20	0.25	Yes	40	40	54.33

Table 11. N_1 : 100, N_2 : 0 , Type-I and Type II error : 0.0001

6 Discussion and Conclusion

In this paper, we have introduced an algorithm, called BioLab, for formally verifying properties of stochastic models of biochemical processes. BioLab represents the first application of statistical model checking to a rule-based model of signaling, which is specified here using the BioNetGen modeling framework. BioLab is (i) an optimal trace-based method for statistical model checking, which generates the minimum number of traces necessary to verify a property and (ii) BioLab provides user-specified bounds on Type-I and Type-II errors.

We demonstrated BioLab on a recently-developed BioNetGen model of the T-cell receptor signaling pathway [23] with two stable states. We verified that both steady states are reachable on a single stochastic trajectory, whereas only a single steady state is reached on a deterministic ODE-based trajectory

starting from the same initial conditions. Moreover, we verified that the system will alternate between these two states with high probability. These findings are relevant for understanding the TCR signaling pathway, which, under physiological conditions, must generate a robust response to a handful of stimulatory input molecules.

There are a number of areas for future research in BioLab. First, the T-cell receptor signaling model has a number of parameters. We verified properties of the pathway over a range of possible parameter values. In some contexts, it may be preferable to first (re)estimate parameter values for a given model. This can be accomplished by using standard parameter estimation techniques from the fields of Statistics and Machine Learning. One might even incorporate model checking into the parameter estimation phase by formally verifying that the parameter estimates reproduce known data, with high probability. Second, our method is presently limited to probabilistic bounded linear temporal logic formulas; we do not allow nested operators. This restriction can be relaxed through the use of different model checking algorithms. Pursuit of these two goals is ongoing.

References

1. D. Barua, J. R. Faeder, and J. M. Haugh. Structure-based kinetic models of modular signaling protein function: Focus on Shp2. *Biophys. J.*, 92:2290–2300, 2007.
2. M. L. Blinov, J. R. Faeder, B. Goldstein, and W. S. Hlavacek. BioNetGen: software for rule-based modeling of signal transduction based on the interactions of molecular domains. *Bioinformatics*, 20(17):3289–3291, 2004.
3. M. L. Blinov, J. R. Faeder, B. Goldstein, and W. S. Hlavacek. A network model of early events in epidermal growth factor receptor signaling that accounts for combinatorial complexity. *BioSyst.*, 83:136–151, 2006.
4. M. L. Blinov, J. Yang, J. R. Faeder, and W. S. Hlavacek. Graph theory for rule-based modeling of biochemical networks. *Lect. Notes Comput. Sci.*, 4230:89–106, 2006.
5. L. Calzone, N. Chabrier-Rivier, F. Fages, and S. Soliman. Machine learning biochemical networks from temporal logic properties. 4220:68–94, 2006.
6. N. Chabrier and F. Fages. Symbolic Model Checking of Biochemical Networks. *Proc 1st Internl Workshop on Computational Methods in Systems Biology*, pages 149–162, 2003.
7. F. Ciesinski and M. Größer. On probabilistic computation tree logic. In *Validation of Stochastic Systems*, LNCS, 2925, pages 147–188. Springer, 2004.
8. E. Clarke, O. Grumberg, and D. A. Peled. *Model Checking*. MIT Press, Cambridge, MA, 1999.
9. V. Danos, J. Feret, W. Fontana, R. Harmer, and J. Krivine. Rule-based modelling of cellular signalling. *Lect. Notes Comput. Sci*, 4703:17–41, 2007.
10. V. Danos, J. Feret, W. Fontana, and J. Krivine. Scalable simulation of cellular signalling networks. *Lect. Notes Comput. Sci*, 4807:139, 2007.
11. J. R. Faeder, M. L. Blinov, B. Goldstein, and W. S. Hlavacek. Rule-based modeling of biochemical networks. *Complexity*, 10:22–41, 2005.
12. J. R. Faeder, M. L. Blinov, and W. S. Hlavacek. Graphical rule-based representation of signal-transduction networks. In *SAC '05: Proceedings of the 2005 ACM symposium on Applied computing*, pages 133–140, New York, NY, USA, 2005. ACM.

13. J. R. Faeder, M. L. Blinov, and W. S. Hlavacek. Rule-based modeling of biochemical systems with BioNetGen. In I. V. Maly, editor, *Systems Biology*, Methods in Molecular Biology. Humana Press, Totowa, NJ, 2008.
14. B. Finkbeiner and H. Sipma. Checking Finite Traces Using Alternating Automata. *Formal Methods in System Design*, 24(2):101–127, 2004.
15. D. T. Gillespie. A general method for numerically simulating the stochastic time evolution of coupled chemical reactions. *J. Comp. Phys.*, 22:403–434, 1976.
16. B. Goldstein, J. R. Faeder, and W. S. Hlavacek. Mathematical and computational models of immune-receptor signaling. *Nat. Rev. Immunol.*, 4:445–456, 2004.
17. W. S. Hlavacek, J. R. Faeder, M. L. Blinov, A. S. Perelson, and B. Goldstein. The complexity of complexes in signal transduction. *Biotechnol. Bioeng.*, 84:783–794, 2003.
18. W. S. Hlavacek, J. R. Faeder, M. L. Blinov, R. G. Posner, M. Hucka, and W. Fontana. Rules for modeling signal-transduction systems. *Science STKE*, 2006:re6, 2006.
19. M. Kwiatkowska, G. Norman, D. Parker, O. Tymchyshyn, J. Heath, and E. Gaffney. Simulation and verification for computational modelling of signalling pathways. *WSC '06: Proceedings of the 38th conference on Winter simulation*, pages 1666–1674, 2006.
20. M. Z. Kwiatkowska, G. Norman, and D. Parker. Prism 2.0: A tool for probabilistic model checking. In *QEST*, pages 322–323. IEEE, 2004.
21. C. Langmead and S. K. Jha. Predicting protein folding kinetics via model checking. *Lecture Notes in Bioinformatics: The 7th Workshop on Algorithms in Bioinformatics (WABI)*, pages 252–264, 2007.
22. C. Langmead and S. K. Jha. Symbolic approaches to finding control strategies in boolean networks. *Proceedings of The Sixth Asia-Pacific Bioinformatics Conference, (APBC)*, pages 307–319, 2008.
23. T. Lipniacki, B. Hat, J. R. Faeder, and W. S. Hlavacek. Stochastic effects and bistability in T cell receptor signaling. *J. Theor. Biol.*, page in press, 2008.
24. L. Lok and R. Brent. Automatic generation of cellular networks with Moleculizer 1.0. *Nat. Biotechnol.*, 23:131–136, 2005.
25. T. McKeithan. Kinetic proofreading in T-cell receptor signal transduction. *Proc Natl Acad Sci*, 92(11):5042–5046, 1995.
26. F. Mu, R. F. Williams, C. J. Unkefer, P. J. Unkefer, J. R. Faeder, and W. S. Hlavacek. Carbon fate maps for metabolic reactions. *Bioinformatics*, 23:3193–3199, 2007.
27. T. Pawson and P. Nash. Assembly of cell regulatory systems through protein interaction domains. *Science*, 300(5618):445–452, 2003.
28. J. Rabinowitz, C. Beeson, D. S. Lyonsdagger, M. M. Davisdagger, and H. M. McConnell. Kinetic discrimination in T-cell activation. *Proc Natl Acad Sci*, 93(4):1401–1405, 1996.
29. M. Vardi. Alternating automata and program verification. *Computer Science Today*, pages 471–485, 1995.
30. A. Wald. sequential tests of statistical hypotheses. *Annals of Mathematical Statistics*, 16(2):117–186, 1945.
31. J. Yang, M. I. Monine, J. R. Faeder, and W. S. Hlavacek. Kinetic Monte Carlo method for rule-based modeling of biochemical networks. *arXiv:0712.3773*, 2007.
32. H. L. S. Younes. *Verification and Planning for Stochastic Processes with Asynchronous Events*. PhD thesis, Carnegie Mellon, 2005.