Colvin et al. BMC Bioinformatics 2010, 11:404
http://www.biomedcentral.com/1471-2105/11/404

BMC Bioinformatics

SOFTWARE Open Access

RuleMonkey: software for stochastic simulation of rule-based models

Joshua Colvin[1], Michael I Monine[2], Ryan N Gutenkunst[2], William S Hlavacek[2,3], Daniel D Von Hoff[1], Richard G Posner[1,4*]

Abstract

Background: The system-level dynamics of many molecular interactions, particularly protein-protein interactions, can be conveniently represented using reaction rules, which can be specified using model-specification languages, such as the BioNetGen language (BNGL). A set of rules implicitly defines a (bio)chemical reaction network. The reaction network implied by a set of rules is often very large, and as a result, generation of the network implied by rules tends to be computationally expensive. Moreover, the cost of many commonly used methods for simulating network dynamics is a function of network size. Together these factors have limited application of the rule-based modeling approach. Recently, several methods for simulating rule-based models have been developed that avoid the expensive step of network generation. The cost of these "network-free" simulation methods is independent of the number of reactions implied by rules. Software implementing such methods is now needed for the simulation and analysis of rule-based models of biochemical systems.

Results: Here, we present a software tool called RuleMonkey, which implements a network-free method for simulation of rule-based models that is similar to Gillespie's method. The method is suitable for rule-based models that can be encoded in BNGL, including models with rules that have global application conditions, such as rules for intramolecular association reactions. In addition, the method is rejection free, unlike other network-free methods that introduce null events, i.e., steps in the simulation procedure that do not change the state of the reaction system being simulated. We verify that RuleMonkey produces correct simulation results, and we compare its performance against DYNSTOC, another BNGL-compliant tool for network-free simulation of rule-based models. We also compare RuleMonkey against problem-specific codes implementing network-free simulation methods.

Conclusions: RuleMonkey enables the simulation of rule-based models for which the underlying reaction networks are large. It is typically faster than DYNSTOC for benchmark problems that we have examined. RuleMonkey is freely available as a stand-alone application http://public.tgen.org/rulemonkey. It is also available as a simulation engine within GetBonNie, a web-based environment for building, analyzing and sharing rule-based models.

Background

Introduction

A great deal of knowledge about signal transduction, which is mediated largely by the interactions of proteins, has been built up over the years [1,2], in part because molecular changes that affect signal-transduction systems play a role in many diseases, such as cancer [3]. Signal-transduction systems are exceedingly complex, and as a result, our ability to manipulate the behaviors of these systems (e.g., through therapies based on molecularly targeted drugs [4]) is limited. To extend our understanding beyond that reachable through intuition alone, researchers are attempting to develop predictive mathematical models for signal-transduction systems [5-7]. These systems are difficult to model for a variety of reasons [8], and new modeling approaches, such as rule-based modeling [9,10], are needed. Rule-based modeling is notable because it provides a solution to the problem of combinatorial complexity [11].

Central to rule-based modeling is the concept of reaction rules, which will be elaborated in detail below. One can think of a rule as a generalized reaction or a coarse-grained description of a molecular interaction and its

* Correspondence: rposner@tgen.org
[1]Clinical Translational Research Division, Translational Genomics Research Institute, Phoenix, AZ 85004, USA

consequences. The power of a rule is that it can concisely capture the site-specific details and consequences of an interaction of a structured molecule, such as a protein [9,12]. Functionally, a rule defines 1) the necessary and sufficient properties of reactants in a class of reactions arising from a molecular interaction, 2) the products of a reaction in this class given any particular set of reactants, and 3) a rate law that governs all reactions defined by the rule.

A set of rules for the protein interactions in a signal-transduction system typically implies a much larger set of reactions. The reason is that processes such as multivalent binding and multisite phosphorylation can yield a combinatorial number of chemically distinct protein complexes and phosphoforms [11,13]. The sheer size of such networks, as well as the cost of simulating large-scale chemical reaction networks using conventional methods, has posed a formidable barrier to the development and analysis of models for signal-transduction systems that account for site-specific details of protein interactions (in terms of rules).

To address the problem of simulating models defined by rules that imply large-scale (bio)chemical reaction networks, Colvin et al. [14] generalized the stochastic simulation method of STOCHSIM[15,16] and implemented this method in software called DYNSTOC, which is freely available [17]. The STOCHSIM/DYNSTOC simulation method involves the use of rules to determine if randomly selected components of molecules undergo a reaction. The method has a computational cost that is independent of the size of the reaction network implied by a set of rules. In this respect, the method is similar to methods reported by Danos et al. [18], Yang et al. [19], and Yang and Hlavacek [20]. These methods have been called "network-free" methods [21], in part because the efficiency of these methods is independent of the size of the reaction network implied by rules.

A key feature of DYNSTOC is its ability to simulate models encoded in the BioNetGen language (BNGL), a general-purpose model-specification language that has been described in detail and that can be interpreted by a number of software tools [21]. Thus, DYNSTOC can be used to simulate a wide array of rule sets that imply large-scale chemical reaction networks. Unfortunately, the network-free simulation method implemented in DYNSTOC is a null-event stochastic simulation method, and as a result, simulation of models with fast reactions is inefficient, because the size of the fixed time step used in the simulation procedure is determined by the rate of the fastest reaction.

Here, we present a software tool, RuleMonkey, which implements a network-free stochastic simulation method for determining the system-level dynamics of molecular interactions represented by rules. In this method, unlike in the null-event method of STOCHSIM/DYNSTOC, the size of the time step is variable, and each time step results in a change of the state of the system being simulated. Like DYNSTOC, RuleMonkey is capable of simulating models defined using BNGL.

Overview of graphical formalism underlying BNGL

The algorithm presented below is intended for simulation of rule-based models that can be specified using BNGL [21], particularly models for the system-level dynamics of protein-protein interactions in signal-transduction systems. In such a model (for examples, see [22-24]), molecules and molecular complexes are represented as graphs, and molecular interactions are represented as graph-rewriting rules. It will be useful to briefly review the graphical formalism underlying BNGL [21,25-27].

Molecules (e.g., proteins) are taken to be the building blocks of chemical species (e.g., protein complexes) and to be composed of reactive components, such as amino acid residues subject to post-translational modifications, linear motifs (e.g., proline-rich sequences of the form PxxP), and protein interaction domains [e.g., Src homology 3 (SH3) domains]. Thus, in a rule-based model, molecules and their components comprise a set of chemical species $S = \{S_1, S_2, ...\}$, and each species S_i is represented by a graph $G_i = (V_i, E_i)$, where the vertices V_i represent molecular components and the edges E_i represent (non-covalent) bonds between components. Thus, the connectivity of molecules within a chemical species is explicitly represented, and the components that mediate molecular interactions are identified. The vertices of a graph are labeled. By convention, vertices that share the same label are taken to be chemically indistinguishable. Labels are immutable. In addition to labels, vertices can be associated with mutable attributes, which can be used to represent the (time-dependent) internal states of components. It is often convenient to introduce an internal state to represent the phosphorylation status of an amino acid residue [25]. By introducing an internal state for each amino acid residue (S/T/Y) subject to modification by kinases and phosphatases, one can track all possible phosphoforms of a protein.

Rules are used to implicitly represent reactions arising from molecular interactions. A rule $g_{\mathrm{LHS}} \rightarrow g_{\mathrm{RHS}}$ is composed of 1) sets of left-hand side (LHS) and right-hand side (RHS) pattern graphs, g_{LHS} and g_{RHS}, which can be viewed as subgraphs of the graphs used to represent chemical species or species graphs, which are denoted $G = \{G_1, G_2, ...\}$; 2) a mapping of the vertices of LHS pattern graphs to the vertices of RHS pattern graphs, which defines a graph transformation; 3) a rate law; and 4) application conditions.

Application conditions are optional but almost all rules encoded in BNGL are endowed with an application condition that restricts molecularity [21,27]. A particular chemical species is a reactant in a rule-defined reaction if its corresponding species graph is matched by a LHS pattern graph (i.e., the species graph contains a subgraph that is isomorphic to the LHS pattern graph) and, in addition, it and any potential co-reactant satisfy the application conditions associated with the rule (if any). The set of molecular components (vertices) directly affected by a transformation will be called a reaction center. Because many species graphs can potentially be matched by a single LHS pattern graph, a rule generally defines not a single reaction, but an entire class of reactions, all of which involve a common transformation. This transformation affects the same reaction center but can take place in multiple molecular contexts, depending on the LHS pattern graph(s) of the rule and the application conditions (if any) of the rule.

Rules are typically associated with application conditions that impose constraints on the molecularity of reactions and the number of reaction products [19,27]. In BNGL, separation of two LHS pattern graphs by a "+" symbol indicates a molecularity of 2, and a rule with a single LHS pattern graph indicates a molecularity of 1. Similarly, separation of two RHS pattern graphs by a "+" symbol indicates two reaction products, and a single RHS pattern graph indicates a single reaction product. Application conditions can be introduced for a variety of other reasons. For example, Monine et al. [28] recently modeled steric effects on multivalent ligand-receptor binding using rules with application conditions that place constraints on the allowable geometric configurations of ligand-induced receptor aggregates. Here, we will only consider application conditions related to molecularity and number of reaction products.

Above, we have implicitly assumed that rules define unidirectional reactions. In BNGL, a short hand notation (< - >) can be used to indicate that the pattern graphs of a rule can be reversed to define reverse reactions [21]. This notation is convenient, as it allows one to avoid redundancy in a model specification. However, the distinction between LHS and RHS graphs is obfuscated. In the discussion that follows, for clarity, we will continue to assume that rules define unidirectional reactions only, which is equivalent to assuming that only the - > notation within BNGL is available. From this point of view, two rules are required to define reversible transformations. For example, A - > B and B - > A must be used instead of simply A <-> B where A and B are pattern graphs. In short, we will refer to those graphs in rules that are used to identify reactants as LHS pattern graphs and those graphs that are used to define transformations as RHS pattern graphs.

The advantage of the rule-based modeling formalism is that a modeler can use a rule to concisely and comprehensivley account for the site-specific details and consequences of a molecular interaction, no matter how many distinct reactions might arise from the interaction. However, this expressiveness comes at a cost. Each individual reaction defined by a rule is assigned the same rate law up to a statistical factor, which can vary from reaction to reaction within a reaction class [21,27]. (Statistical factors are discussed further below.) In reality, each reaction in a reaction class may be characterized by a unique rate law. Thus, the rate law associated with a rule provides only a coarse-grained description of the kinetics of the reactions within the rule-defined reaction class. However, because a rule can specify the molecular context that is permissive for reaction, the coarseness of a rule can be adjusted by tuning the contextual elements of the rule. At the finest level, the contextual elements are highly specific and a rule defines a unique reaction. At the coarsest level, the rule is free of contextual constraints, meaning that the rule indicates that a reaction center can undergo a reaction regardless of the molecular context in which that reaction center is found.

Model assumptions

Having presented an overview of the graphical formalism underlying BNGL, we can now introduce the assumptions upon which the algorithm implemented in RuleMonkey is based.

We consider a well-mixed isothermal reaction compartment of volume V containing a set of M molecules $P = \{P_1, ..., P_M\}$, which we take to be proteins. Each molecule P_i is associated with a set of components $c_i = \{c_{i1}, c_{i2}, ...\}$. These components, each of which are optionally associated with an internal state, undergo reactions according to a set of N reaction rules $R = \{R_1, ..., R_N\}$. Each rule R_i defines a class of reactions, and each member of this class is characterized by a single rate law up to a statistical factor, as mentioned earlier. Given this rate law, a cumulative rate r_i can be determined for the class of reactions defined by R_i, at least in principle. This cumulative rate corresponds to the sum of the rates for the individual reactions in the class of reactions defined by R_i. By convention, the rate of each reaction will be taken to be given in terms of the number of molecules undergoing the reaction per unit time in V. In other words, each r_i has the same units as $1/t$.

As a simplification, we will consider only mass-action rate laws. Thus, for example, we will assume that a bimolecular reaction $A + B \rightarrow$ product(s) is characterized by a rate law of the form $fk_V n_A n_B$, where n_A is the number of A type molecules or molecular complexes in V, n_B is the number of B type molecules or molecular complexes in V, k_V is a rate constant expressed in the

same units as $1/t$, and f is a statistical factor that accounts for the number of (chemically indistinguishable) ways that the A and B molecules (or molecular complexes) can react, i.e., the number of degenerate reaction paths from reactants to products. These molecules (complexes) can react in more than one way if, for example, A and B associate via binding sites that are present in A and/or B in multiple copies. Thus, k_V is a so-called "single-site" rate constant. Rate constants associated with rules are assumed to be single-site rate constants according to the conventions of BNGL [21]. Note that k_V is volume dependent and can be equated to $k/(N_A V)$, where N_A is Avogadro's number and k is the rate constant expressed in the same units as $N_A V/t$. The statistical factor f, which is related to symmetry, can be determined as described by Blinov et al. [27].

As another simplification, we will assume that internal-state dynamics are described by two-state trajectories. Thus, a component c_{ij} of molecule P_i will be associated with an internal state $\sigma_{ij}(t) \in 0[1]$ or none at all. (Recall that a component need not be associated with an internal state.) If a component is a tyrosine residue subject to phosphorylation and dephosphorylation, then an internal state could be introduced to represent its phosphorylation status [25], with one state value representing the unphosphorylated form and the other state value representing the phosphorylated form. Software tools for rule-based modeling, including RuleMonkey, actually allow for greater than two internal states to be associated with a component.

For purposes of discussion, we will focus on rules specifying reactions that result in the association or dissociation of two components or that change the internal state of a component (i.e., reactions that can be modeled as graph transformations involving the addition or removal of an edge in a graph or the change of a vertex attribute in a graph). Note that a component is taken to have at most one binding partner at a time, as usual [21]. Extension of the discussion that follows to account for other types of rules, such as rules for synthesis, degradation and trafficking between compartments, is straightforward. RuleMonkey is capable of handling these types of reactions, which can be specified in accordance with the conventions of BNGL [21]. For a system in which only association, dissociation, and state-change reactions are possible, counts of molecules and components are conserved quantities. Moreover, for such a system, the LHS pattern graph(s) of a rule will serve to identify either one or two reactants for each reaction defined by the rule.

The population-based approach to simulation of a rule-based model

The state of a system governed by a rule set R can be taken to be given by the vector $\mathbf{x}(t) = (x_1(t), ..., x_n(t)]^T$, is the population level of the molecule or molecular complex S_i described by graph G_i and n is the number of chemical species that are populated over the time interval of interest. In the limit of continuous population sizes, $\mathrm{x}(t)$ evolves deterministically according to a system of coupled ordinary differential equations (ODEs):

$$\frac{d\mathrm{x}}{dt} = \mathbf{f}(\mathbf{x}) \tag{1}$$

where the vector field $\mathbf{f}(\mathbf{x})$ is composed of mass-action terms derived from the rate laws associated with the rule set R. For a rule set that implies m unidirectional reactions, there are m mass-action terms.

In principle, Eq. 1 can be derived from R [25,29]. However, derivation and numerical integration of Eq. 1 is impractical when a rule set implies a large-scale chemical reaction network (i.e., large m and n), which is often the case [9,14,18,19]. Reduced forms of Eq. 1, involving transformations of the variables x, can sometimes be found through analysis of R [30-34], and on-the-fly stochastic simulation procedures [25,35,36] can sometimes be used to limit enumeration of the reactions implied by R, which is expensive. Nevertheless, these methods fail to be practical for many rule sets.

To overcome this problem, kinetic Monte Carlo (KMC) procedures have been developed in which R is used directly to generate stochastic trajectories consistent with the chemical master equation that corresponds to Eq. 1 [14,18-20]. These procedures avoid enumerating the reaction network implied by R. Thus, the term "network-free" is apt for these simulation procedures. The algorithm presented below is a network-free procedure.

The particle-based approach to simulation of a rule-based model

The RuleMonkey algorithm, like other network-free simulation procedures, is particle based, meaning that each and every material component of a system is tracked, regardless of whether the material components are chemically distinguishable. In other words, given a set of rules R and a set of instances of chemical species graphs, each potentially present in multiple copies, these graphs and their constituent parts are tracked individually as the graph transformations defined by R are applied to them (as part of a simulation procedure) and the constituent parts move through the space of possible chemical species. Let us use $\hat{S}(t)$ to denote the set of instances of chemical species in a system of volume V at time t. These instances of chemical species can, in principle, be partitioned into disjoint sets, the set of n populated chemical species:

$$\hat{S}(t) = \{\hat{S}_{1,1}, \ldots, \hat{S}_{1,x_1(t)} \mid \cdots \mid \hat{S}_{n,1}, \ldots, \hat{S}_{n,x_n(t)}\} \tag{2}$$

where $\hat{S}_{i,j}$ is the jth copy of species S_i and $x_i(t)$ is the population level of (or equivalently, the number of copies of) species S_i at time t in V. In a network-free simulation procedure, the species instances $\hat{S}$ are tracked but they are not partitioned as indicated in Eq. 2, which is essentially equivalent to determining x(t), the population levels of the chemical species in V. To determine x(t) from $\hat{S}(t)$, one must identify and count the graphs in $\hat{S}(t)$ corresponding to each species in $S(t)$, which is an expensive procedure because it necessarily involves repeated solution of the graph isomorphism problem [21,27].

Instead of relying on a partition of $\hat{S}$ as indicated in Eq. 2 or knowledge of x(t) to advance a simulation, a network-free algorithm relies on knowledge of the complete component state of a system, i.e., the state of each component in the system. The state of an individual component c is the information required to uniquely identify 1) the component, including its label and the label of the molecule of which it is a part; 2) the internal state of the component (if any); and 3) the bound state of the component, including its binding partner (if any). Let us use $\Sigma_c(t)$ to denote the complete component state of a system, which we will also refer to as simply the system state. Note that $\Sigma_c(t)$ fully defines $\hat{S}(t)$ (and vice versa). Because the graph-rewriting operations defined by rules directly alter Σ_c, the system state as expressed in terms of component states can be easily tracked (with a memory requirement that depends on the number of material components in V) and $\Sigma_c(t)$ can be dynamically updated as rules in R are applied to graphs in $\hat{S}$ in a simulation procedure.

Observables

There is a one-to-one relationship between $\Sigma_c(t)$ (or $\hat{S}(t)$) and x(t), but as indicated above, the RuleMonkey algorithm, like other network-free simulation procedures, avoids determining x(t). How then are system properties of interest determined as a function of time?

The output of a network-free simulation procedure is determined by a specification of a set of ω observables, $O = \{O_1, ..., O_\omega\}$. Each observable O_i is composed of a pattern graph g_i, making it similar to a rule. It is also associated with a time-dependent match number $m_i(t)$, which is a function of g_i and the system state:

$$m_i(t) = \sum_{s \in \hat{S}(t)} w(g_i, s)\mu(g_i, s) \tag{3}$$

$$\mu(g_i, s) = \begin{cases} 1 & \text{if a subgraph of } s \text{ is isomorphic to } g_i \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

and $w(g_i, s)$ is a weight that is specified to be either 1 or the number of times that s is matched by g_i, i.e., the number of distinct images of g_i in s or the number of distinct mappings (injections) of the elements of g_i to a subset of the elements of s. In a BioNetGen input file, the former case is indicated by the BNGL keyword Species, whereas the latter case is indicated by the BNGL keyword Molecules [21]. The match number associated with an observable of the Species type is a count of the number of chemical species in V containing at least one moiety matched by the pattern graph of the observable. The match number associated with an observable of the Molecules type is a count of the number of moieties in V matched by the pattern graph of the observable. Because the match numbers associated with observables are functions of the system state (Eqs. 3), these quantities can be easily tracked and dynamically updated as the system state changes in a simulation procedure.

Algorithm

Given the background presented above, we can now present an outline of the simulation procedure implemented in RuleMonkey. This procedure has more or less already been described in earlier work [18-20], although key implementation details, described below, are novel. The procedure can be used to produce stochastic trajectories consistent with the chemical master equation corresponding to a system of ODEs of the form of Eq. 1. A comparison of methods is included below.

We take as given a set of N rules $R = \{R', R''\}$, where R' denotes the set of rules with a single LHS pattern graph and R'' denotes the set of rules with a pair of LHS pattern graphs. We also take as given an initial time t(= 0), a system volume V, and a set of initial species instances $\hat{S}(t)$, from which the component state of the system, $\Sigma_c(t)$, can be determined. For now, we will assume that there is a procedure that allows us to calculate the set of rates $r(t) = \{r_1(t), ..., r_N(t)\}$ associated with R. The method by which these rates are calculated will be described in detail below.

We will also assume that there are procedures that allow us to identify the potential reactants in rule-defined reactions and the reaction centers in these reactants. Thus, for a rule R_u in R' we will assume that a set of species instances $\bar{S}_u \subseteq \hat{S}$ can be identified as matching the LHS pattern graph of R_u. The species instances in $\bar{S}_u$ are potential reactants in a reaction defined by R_u. The reaction centers in $\bar{S}_u$ are the components that are matched by the LHS pattern graph of R_u and that are directly modified in a reaction defined by R_u (e.g., a component that joins or leaves a bond with another component or that changes internal state). Likewise, for a rule R_v in R'', we will assume that sets of species instances $\bar{S}_u \subseteq \hat{S}$ and $\bar{S}'_v \subseteq \hat{S}$ can be identified that match the first and second LHS pattern graphs of R_v, respectively. Pairs of species instances in $\bar{S}_v \times \bar{S}'_v$ are potential co-reactants in a reaction defined by R_v.

We can now outline the key steps of the network-free simulation procedure implemented in RuleMonkey, which is similar to Gillespie's method [36].

Initialize

Specify V, t, R, including the single-site rate constants in rate laws associated with rules, and a seed set of species instances $\hat{S}(t)$, which determines $\Sigma_c(t)$. For each rule R_i in R, calculate the rate r_i associated with this rule and identify the species instances in $\hat{S}(t)$ that are matched by the LHS pattern graph(s) of the rule. Finally, specify a set of observables, O, and use Eq. 3 to calculate the match numbers associated with these observables.

Increment time and select a rule in accordance with rule rates

Calculateτ, the waiting time to the next reaction event in the system, using

$$\tau = -\log(\rho_1) / r_{\text{tot}} \tag{5}$$

where $\rho_1 \in (0,1)$ is a uniform deviate and $r_{\text{tot}} = \sum_{i=1}^{N} r_i$ Recall that the rule rate r_i, which is taken to have units of inverse time in Eq. 5, is the cumulative rate of all possible reactions defined by rule R_i. Equation 5 follows from $P(\tau) = r_{\text{tot}} e^{-r_{\text{tot}}\tau}$, where $P(\tau)$ is the probability distribution function for τ [37]. This distribution holds if the rate at which the system leaves state $\Sigma_c(t)$ is simply r_{tot}, the sum of the cumulative rates for the classes of reactions through which the system state can change.

Determine the class of the next reaction event by finding the smallest integer I that satisfies

$$\sum_{i=1}^{I} r_i > \rho_2 r_{\text{tot}} \tag{6}$$

where $\rho_2 \in (0,1)$ is a second uniform deviate. The next reaction event will be in the class defined by rule R_I, where $I \in [1, ..., N]$. Equation 6 indicates that the class of the next reaction event is randomly selected with probability in proportion to the cumulative rate of reactions within that reaction class. It should be noted that the cost of finding R_I using Eq. 6 is proportional to N. A more efficient procedure, with cost proportional to log$N$, could be implemented [38,39], but Eq. 6 is used in RuleMonkey for simplicity. For simulation problems that we have considered, performance profiling indicates that this simplification does not limit the efficiency of RuleMonkey.

Select a set of reactants and use the selected reactant(s) and rule to change the system state

Additional uniform deviates are needed in the simulation of a reaction event. If $R_I \in R'$ then a species instance $s \in \overline{S}_I$ is randomly selected with probability in proportion to the number of distinct reaction centers in s matched by the LHS pattern graph of R_I. On the other hand, if $R_I \in R''$ then a first species instance $s_1 \in \overline{S}_I$ is randomly selected with probability in proportion to the number of distinct reaction centers in s_1 matched by the first LHS pattern graph of R_I, and a second species instance $s_2 \in \overline{S}'_I \setminus s_1$ is randomly selected with probability in proportion to the number of distinct reaction centers in s_2 matched by the second LHS pattern graph of R_I. The graph transformation defined in rule R_I is then applied to a randomly selected set of reaction centers in the selected reactant(s), which changes the system state Σ_c (or equivalently, $\hat{S}$). In other words, the randomly selected set of species instances, s or $\{s_1, s_2\}$, is essentially removed from the system and replaced by a new set of species instances. The new species instance(s) correspond to the product(s) of the reaction event defined by R_I.

Update sets and quantities that depend on the system state

Increment time $t \leftarrow +\tau$ and update the sets of reactants associated with rules. Likewise, update the match numbers associated with observables. The steps outlined above that follow initialization are iterated until a stopping criterion is satisfied.

Implementation

The simulation method described above has been implemented in software called RuleMonkey. RuleMonkey is available as a stand-alone application, which is freely available via the RuleMonkey web site [40], and as a simulation engine within the GetBonNie environment [41,42]. The input for a simulation is a BNGL-encoded model and simulation specification. In other words, RuleMonkey reads and interprets standard BioNetGen input files, like DYNSTOC [14] and BioNetGen [25,29,43]. The conventions of BNGL are described in detail elsewhere [21]. The novel details of algorithm implementation, which are related to the calculation of rule rates, are described below. Usage considerations are also discussed.

Calculation of rule rates

Below, equations are given for exactly calculating the rates associated with five types of reaction rules, which generate reactions of the types illustrated in Fig. 1. The five rule types are sufficient to specify a wide range of models. For efficiency, the equations given below are used only in the initialization step of the simulation algorithm. After initialization, as described in detail for rejection-free simulation of multivalent ligand-receptor interactions [20], the rates given by these equations are updated on the fly as reactions occur, which requires that matches of LHS pattern graphs in rules be tracked during a simulation.

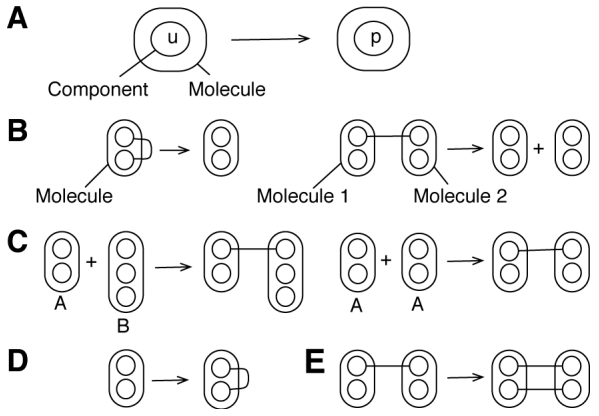

Figure 1 Examples of reactions. Five types of reactions are illustrated using the graphical conventions of Faeder et al. [26]: a) a state-change reaction; b) two dissociation reactions, one that results in a single product and one thatresults in two products; c) two bimolecular association reactions, one of the form $A + B \rightarrow$ product (s) and one of the form $A + A \rightarrow$ product(s); d) a monogamous ring-closure reaction; and e) a non-monogamous ring-closure reaction.

Unimolecular state-change reactions

We will use R_a to denote a rule that defines unimolecular state-change reactions, such as that illustrated in Fig. 1(a). We will assume that R_a has the following form: $l_a \rightarrow \rho_a$, where l_a is a LHS pattern graph and ρ_a is a RHS pattern graph. Thus, R_a defines unimolecular reactions that each have a single reactant and a single reaction product. We will assume that the pattern graph l_a identifies a reaction center composed of a single molecular component in a particular internal state.

Given a set of species instances $\hat{S}(t)$ at time t, the instantaneous rate $r_a(t)$ associated with R_a is given by the following expression:

$$r_a(t) = k_a \sum_{s \in \hat{S}(t)} \nu_a(l_a, s) \tag{7}$$

where k_a is the single-site rate constant associated with the mass-action rate law of R_a and $\nu_a(l_a, s)$ is a function that gives the number of distinct reaction centers in species instance s matched by l_a (i.e., the number of components in species instance s in an internal state consistent with the component and internal state identified by l_a). The function ν_a is similar to that of μ (Eq. 4). Note that $\nu_a(l_a, s) = 0$ if no subgraph of s is isomorphic to l_a. Also note that $\nu_a(l_a, s)$ is not necessarily equivalent to the number of times that s is matched by l_a but rather is equivalent to the number of distinct reaction centers among the subgraphs of s matched by l_a. It is important to make the distinction between number of matches and number of distinct reaction centers in matches when a LHS pattern graph specifies a contextual constraint that can be satisfied in more than one way.

In the course of a RuleMonkey simulation, rates of the form of r_a are updated on the fly as reactions occur and are never recalculated *de novo* after initialization, which would be inefficient. Thus, for example, if a species instance s matched by l_a is removed from $\hat{S}$, then r_a is decremented by $k_a \nu_a(l_a, s)$. Similarly, if a species instance s matched by l_a is added to $\hat{S}$, then r_a is incremented by $k_a \nu_a(l_a, s)$. Updates of rule rates are facilitated by tracking matches of LHS pattern graphs of rules to subgraphs of the graphs representing the species instances in a system. As part of this bookkeeping procedure, when a new species instance appears in a system (as a result of reaction), it is checked against all the LHS pattern graphs of rules to determine the types of reactions in which it can potentially participate and to calculate updates of the rates associated with rules. When a species instance disappears from a system (as a result of reaction), the rates associated with rules are appropriately adjusted as well. The cost of updating a rate such as that given by Eq. 7, or any of the equations given below, depends on the number of rules in a model and the number of chemical species instances created and destroyed in a reaction. However, the cost per time step is more or less fixed over the course of a simulation. The overall efficiency of simulation depends on the efficiency of the procedures used to update rates and to store and update other information needed to implement the simulation algorithm described above.

Unimolecular dissociation reactions

We will use R_b to denote a rule that defines unimolecular dissociation reactions, such as those illustrated in Fig. 1(b). We will assume that R_b has one of two forms: $l_b \rightarrow \rho_b$ or $l_b \rightarrow \rho'_b + \rho''_b$ where l_b is a LHS pattern graph, and ρb, ρ'_b, and ρ''_b are RHS pattern graphs. In the former case, R_b defines unimolecular reactions that each have a single reaction product, as when a bond opens but no constituent parts of the reactant species dissociate. In the latter case, R_b defines unimolecular reactions that each have two reaction products, as when a bond opens and constituent parts of the reactant species dissociate from each other. We will assume that the pattern graph l_b identifies a reaction center composed of two molecular components that are directly connected by a (non-covalent) bond.

Given a set of species instances $\hat{S}(t)$ at time t, the instantaneous rate $r_b(t)$ associated with R_b is given by the following expression:

$$r_b(t) = k_b \sum_{s \in \hat{S}(t)} \nu_b(l_b, s) \tag{8}$$

where k_b is the single-site rate constant associated with the mass-action rate law of R_b and $\nu_b(l_b, s)$ is a

function that gives the number of distinct reaction centers in species instance s matched by l_b (i.e., the number of pairs of connected components in species instance s identified by l_b). Note that a species instance s is considered to be matched by l_b only if application of the graph-rewriting operation of R_b to s produces the correct number of reaction products, i.e., the number of products consistent with the RHS of R_b. The function v_b in Eq. 8 is analogous to the function v_a in Eq. 7.

As noted above for rates of the form of r_a, rates of the form of r_b are updated on the fly as reactions occur during the course of a RuleMonkey simulation. These updates are straightforward and follow from Eq. 8.

Bimolecular association reactions

We will use R_c to denote a rule that defines bimolecular association reactions, such as those illustrated in Fig. 1 (c). We will assume that R_c has the following form: $l_c + l'_c \rightarrow \rho_c$ where l_c and l'_c are LHS pattern graphs and ρ_c is a RHS pattern graph. We will assume that the pattern graphs l_c and l'_c identify two reactants and a single component within each. This pair of components forms the reaction center identified by R_c and these components are connected in the product of reaction. Consistent with the conventions of BNGL [21], the plus sign that separates l_c and l'_c indicates that the molecularity of reactions defined by R_c must be 2.

Given a set of species instances $\hat{S}(t)$ at time t, the instantaneous rate $r_c(t)$ associated with R_c is given by

$$r_c(t) = k_c(A_c - B_c) \tag{9}$$

where

$$A_c = \sum_{s \in \hat{S}(t)} \left(v_c(l_c, s) \sum_{s' \in \hat{S}, s' \neq s} v_c(l'_c, s') \right) \tag{10}$$

and

$$B_c = \frac{1}{2} \sum_{s \in \hat{S}(t)} \left((v'_c(l_c, l'_c, s) \sum_{s' \in \hat{S}, s' \neq s} v'_c(l_c, l'_c, s') \right) \tag{11}$$

In the above equations, k_c is the single-site volume-dependent rate constant associated with the mass-action rate law of R_c, $v_c(l_c, s)$ is a function that gives the number of distinct reaction centers in species instance s matched by l_c, and $v'_c(l_c, l'_c, s)$ is a function that gives the number of distinct reaction centers in species instance s matched by both l_c and l'_c. In the case where no species instance is matched by both l_c and l'_c, the positive, first term in Eq. 9 accounts for all possible combinations of reactants and the second term reduces to zero. In other cases, the first term over counts the number of combinations, and the second term corrects for the over counting.

On-the-fly updates of rates of the form of r_c are fairly complicated but follow from Eq. 9. Basically, when the system state changes, r_c is incremented or decremented by the terms in Eq. 9 that are added or removed as a result of the system state change.

Unimolecular monogamous ring-closure reactions

We will use R_d to denote a rule that defines unimolecular monogamous ring-closure reactions, such as that illustrated in Fig. 1(d). We assume that R_d has the following form: $l_d l'_d \rightarrow \rho_d$, where l_d and l'_d are LHS pattern graphs and ρ_d is a RHS pattern graph. Because l_d and l'_d are not separated by a plus sign, these pattern graphs can be understood to identify two reactive components within the same chemical species, consistent with the conventions of BNGL [21]. Here, we further assume that these reactive components lie within the same molecule. The pair of components identified by l_d and l'_d forms the reaction center identified by R_d, which defines a graph-rewriting operation that connects the two vertices in the reaction center.

Given a set of species instances $\hat{S}(t)$ at time t, the instantaneous rate $r_d(t)$ associated with R_d is given by

$$r_d(t) = k_d(A_d - B_d) \tag{12}$$

where

$$A_d = \sum_{s \in \hat{S}(t)} \sum_{m \in s} v_d(l_d, m) v_d(l'_d, m) \tag{13}$$

and

$$B_d = \frac{1}{2} \sum_{s \in \hat{S}(t)} \sum_{m \in s} v'_d(l_d, l'_d, m) \tag{14}$$

In the above equations, k_d is the single-site rate constant associated with the mass-action rate law of R_d, $v_d(l_d, m)$ is a function that gives the number of distinct reaction centers within a common molecule m in a species instance s matched by l_d, and $v'_d(l_d, l'_d, m)$ is a function that gives the number of distinct reaction centers within a common molecule m in a species instance s matched by both l_d and l'_d. In BNGL, the graphs that represent chemical species are composed of graphs that represent types of molecules [21,27]. Thus, m in Eq. 12 references a particular subgraph of a species instance graph s that corresponds to a type of molecule. Updates of r_d necessitated by system state changes follow from Eq. 12.

Unimolecular non-monogamous ring-closure reactions

We will use R_e to denote a rule that defines unimolecular non-monogamous ring-closure reactions, such as that illustrated in Fig. 1(e). We assume R_e has the following form: $l_e l'_e \rightarrow \rho_e$ where l_e and l'_e are LHS pattern graphs and ρ_e is a RHS pattern graph. According to the conventions of BNGL [21], the LHS pattern graphs le and l'_e identify two reactive components within the same chemical species. Here, we further assume that these reactive components lie in different molecules within the same chemical species. The pair of components identified by l_e and l'_e forms the reaction center identified by R_e, which defines a graph-rewriting operation that connects the two vertices in the reaction center.

Given a set of species instances $\hat{S}(t)$ at time t, the instantaneous rate $r_e(t)$ associated with R_e is given by

$$r_e(t) = k_e \sum_{s \in \hat{S}(t)} [A_e(s) - B_e(s)] \tag{15}$$

where

$$A_e(s) = \sum_{m \in s} \nu_e(l_e, m) \sum_{m' \in s, m' \neq m} \nu_e(l'_e, m') \tag{16}$$

and

$$B_e(s) = \frac{1}{2} \sum_{m \in s} \nu'_e(l_e, l'_e, m) \sum_{m' \in s, m' \neq m} \nu'_e(l_e, l'_e, m') \tag{17}$$

In the above equations, k_e is the single-site rate constant associated with the mass-action rate law of Re, $\nu_e(l_e, m)$ is a function that gives the number of distinct reaction centers within a common molecule m in a species instance s matched by l_e, and $\nu'_e(l_e, l'_e, m)$ is a function that gives the number of distinct reaction centers within a common molecule m in a species instance s matched by both l_e and l'_e. Updates of r_e necessitated by system state changes follow from Eq. 15.

Usage considerations

In the Actions block of a BioNetGen input file [21], a RuleMonkey-specific command, simulate_rm, is used to initiate a simulation. This command takes the same arguments as analogous commands that invoke other simulation engines, such as DYNSTOC [14]. The output of a RuleMonkey simulation is a .info file and a .gdat file. The .info file reports metadata about a simulation run, such as the execution time. The .gdat file reports a time course for each observable quantity defined in an input file (a file with a .bngl extension). Observables are sums or weighted sums of population levels of chemical species (Eq. 3), and they are defined according to the conventions of BNGL [21]. The time points at which results are reported in the .gdat file are specified by a user using the simulate_rm command. These results are generated through evaluation of the system state at each of the time points of interest.

Results

Validation

We validated RuleMonkey by comparing simulation results against those obtained using BioNetGen [25,29], DYNSTOC [14], and problem-specific codes [19,28,44]. The following models were considered (Table 1; Figs. 2, 3, 4): 1) the multisite phosphorylation model introduced by Colvin et al. [14], testcase1.bngl (see Additional file 1); 2) the TLBR (trivalent ligand-bivalent receptor) model introduced by Yang et al. [19] and considered by Colvin et al. [14] and here in Fig. 2B, testcase2a.bngl (see Additional file 2) and testcase2b.bngl (see Additional file 3); 3) a model introduced here with reaction events occurring on disparate time scales, stiff.bngl (see

Table 1 Relative efficiency of RuleMonkey for benchmark problems

Bench- mark	Input file (.bngl)	Reference(s)	RuleMonkey	DYNSTOC	Problem specific code	BioNetGen
1	testcase1	[14]	1.3×10^{-5}	1.6×10^{-2}	N/A	3.4×10^{-5}
2	testcase2b	[14,19]	2.4×10^{-5}	1.2×10^{-4}	2.2×10^{-5}	–
3	stiff	This study	4.6×10^{-6}	2.6×10^{-4}	N/A	5.0×10^{-7}
4	pltr	[20]	1.1	1.1	4.1×10^{-5}	–
5	egfr net	[45]	1.1×10^{-5}	1.8×10^{-4}	N/A	3.7×10^{-6}
6	fceri	[46,47]	1.9×10^{-5}	1.9×10^{-3}	N/A	6.7×10^{-6}
7	lat	[44]	9.9×10^{-3}	1.3×10^{-2}	1.8×10^{-5}	–

The performance of RuleMonkey for seven benchmark problems is compared against that of DYNSTOC [14], problem-specific codes implementing network-free procedures [20,44], and the simulate_ssa procedure of BioNetGen [21,25,29]. The problem-specific codes for benchmark problems 2, 4 and 7 were provided by M. I. Monine; these codes have been described by Yang et al. [19], Monine et al. [28], and Nag et al. [44]. Each table entry indicates seconds of CPU time per reaction event. For benchmark problems 2, 4, and 7, BioNetGen is unable to exhaustively generate the reaction network needed for generate-first simulation (without network truncation), as these problems involve simulating polymerization-like reactions. For BioNetGen simulations, the cost of network generation is not included in the table entries. All simulations were performed on a Macintosh desktop computer, equipped with a single G5 processor. Simulations were performed as specified in the indicated BioNetGen input files, which are available as Additional files 1 and 3-8 and at the RuleMonkey web site [40]. The benchmark problems are described in more detail in the text and in the references cited.

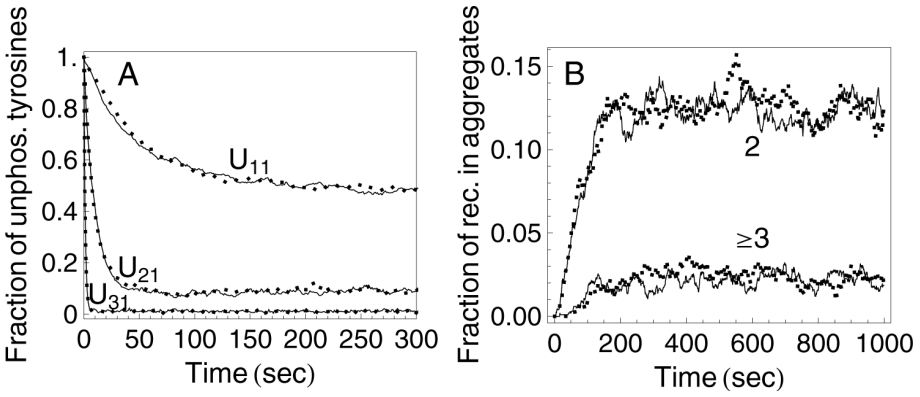

Figure 2 Example validation results. a) Simulation of the model of Test Case I of Colvin et al. [14] using DYNSTOC (dotted line) and RuleMonkey (solid line). Parameters used are the same as those reported for Fig. 2A of Colvin et al. [14]. The BioNetGen input file processed by DYNSTOC and RuleMonkey is called testcase1.bngl (Additional file 1). b) Simulation of the TLBR model, the model of Test Case II of Colvin et al. [14], using DYNSTOC (dotted line) and RuleMonkey (solid line). Parameters used are the same as those reported for Fig. 2B of Colvin et al. [14]. The BioNetGen input file processed by DYNSTOC and RuleMonkey is called testcase2b.bngl (Additional file 3).

Additional file 4); 4) a model introduced here for interaction of a pentavalent ligand with a trivalent cell-surface receptor, pltr.bngl (see Additional file 5); 5) the model of Blinov et al. [45] for early events in epidermal growth factor receptor (EGFR) signaling, egfr net.bngl (see Additional file 6); 6) the model of Goldstein et al. [46] and Faeder et al. [47] for early events in IgE receptor (FcϵRI) signaling, `fceri.bngl` (see Additional file 7); and 7) the model of Nag et al. [44] for crosslinking of phosphorylated LAT molecules by intracellular Grb2-Sos1 complexes, `lat.bngl` (see Additional file 8). The BioNetGen input files that specify these models and RuleMonkey simulations of them are available as Additional files 12345678. They are also available at the RuleMonkey web site [40].

Each model is specified as a BioNetGen input file, which can be processed by BioNetGen, DYNSTOC, and RuleMonkey. The input files for these different software tools are the same, except for the simulation commands in the Actions blocks [21]. As noted above, the simulation command for RuleMonkey is simulate_rm. BioNetGen accepts two simulation commands, simulate_ode and simulate_ssa. In calculations with BioNetGen, we used simulate_ssa, which invokes an efficient extension of Gillespie's method, preceded by a generate_network command, which prompts BioNetGen to implement the procedures needed to explicitly derive the reaction network implied by a set of rules. In other words, we only used BioNetGen to perform simulations via the so-called generate-first approach [9,21,48].

The reason we only consider generate-first simulations using BioNetGen is that simulation cost and network-generation cost can be easily separated when this simulation approach is taken, as it consists of two steps. The first step is network generation. The second step is simulation. In the on-the-fly approach, execution switches back and forth between network generation and simulation. It has been observed that stochastic simulation via the on-the-fly approach tends to be faster than stochastic simulation via the generate-first approach [35]. Thus, if we find that generate-first simulation is faster than network-free simulation, we can assume with some confidence that on-the-fly simulation is also faster than network-free simulation.

Typical validation results are shown in Fig. 2. In all cases, RuleMonkey was found to produce results consistent with those obtained using the other methods/codes considered.

Performance

As can be seen from the results of Table 1, RuleMonkey is more efficient than DYNSTOC, but less efficient than BioNetGen and problem-specific codes (for cases where BioNetGen and problem-specific codes can be applied). BioNetGen is unable to simulate three of the test models via the generate-first approach because the reaction networks implied by the rule sets of these models cannot be generated exhaustively. Thus, of the available general-purpose simulation codes compliant with the conventions of BNGL, RuleMonkey and DYNSTOC are the most widely applicable (because these tools implement general-purpose network-free simulation methods), and RuleMonkey is more efficient than DYNSTOC (Table 1). It should be noted that we would expect the generate-first approach to simulation [9,21,48], which may involve either stochastic simulation or numerical integration of ODEs (although we consider only stochastic simulation here), to be more efficient than network-free simulation for cases where the generate-first approach is feasible because of the bookkeeping costs of

network-free simulation, which requires data structures and update schemes to track the component state of a system. Unfortunately, simulation methods that rely on derivation of the reaction network implied by a rule set, which is computationally expensive, are often not feasible for rule sets that imply large-scale chemical reaction networks. Feasibility depends on effective network size, which is a function of model parameter values. For some parameter values, network generation can be halted arbitrarily or limited as in the on-the-fly approach to simulation [9,21,48]. For other parameter values, as demonstrated by Yang et al. [19], for example, network generation is a limiting factor, and network-free simulation is essential.

The performance of RuleMonkey scales with problem size in a way that is similar to that of DYNSTOC [14] and the problem-specific codes of Yang et al. [19] and Yang and Hlavacek [20]. Results obtained from simulation of the TLBR model of Yang et al. [19] are shown in Fig. 3. The TLBR model, a kinetic version of the model of Goldstein and Perelson [49], characterizes the interactions of trivalent ligands with bivalent cell-surface receptors. In Fig. 3(a), the cost of simulation is shown as a function of the number of receptors N_R. As N_R increases, the frequency of reactions increases. In Fig. 3 (b), the cost of simulation is shown as a function of the dimensionless parameter β, which characterizes ligand-induced receptor crosslinking at equilibrium. As β increases, the equilibrium size of ligand-induced receptor aggregates increases. Thus, this parameter effectively controls the number of populated species and the number of reactions with non-zero flux. As can be seen, the cost of simulation is constant up to a critical value of β, where a percolation transition occurs [19,49]. Above this threshold, simulation cost increases linearly. Yang et al. [19] have shown that this scaling arises because of the expense of enforcing an application condition of a rule of the TLBR model that prohibits the formation of cyclic receptor aggregates. Thus, in the case of the TLBR model, simulation efficiency is affected by model parameter values. We expect that simulation of the model of Benchmark Problem 4 in Table 1, which is closely related to the TLBR model, is affected by model parameter values in a similar way.

Finally, we compared the performances of RuleMonkey and DYNSTOC for problems with reactions occurring on different time scales (Fig. 4). As can be seen, as the rate of the fastest reaction in a system increases, the efficiency of DYNSTOC degrades relative to that of RuleMonkey.

Conclusions

RuleMonkey is a general-purpose simulator for rule-based models encoded in BNGL. RuleMonkey complements BioNetGen [21,25,29], the first software tool to have the capability to simulate rule-based models encoded in BNGL. The rules that comprise BNGL-encoded models can be expanded by BioNetGen into the set of reactions implied by the rules, and BioNetGen can simulate the kinetics of these reactions using conventional methods, such as numerical integration of the corresponding system of ordinary differential equations

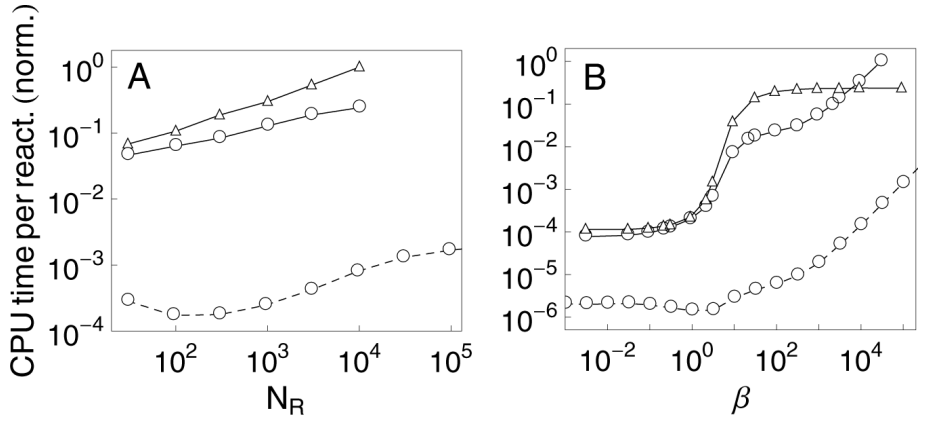

Figure 3 RuleMonkey is efficient for simulation of rule-based models characterized by large-scale networks. We compare RuleMonkey (solid lines marked by triangles), DYNSTOC (solid lines marked by open dots) and a problem-specific implementation of the method of Yang et al. [19] (dotted lines); these methods are used to simulate the TLBR model. a) Scaling of computational cost with system size, where size is measured by N_R, the number of cell-surface receptors. b) Scaling of computational cost with dimensionless parameter $\beta = N_R k+2/k_{off}$, which controls the (equilibrium) extent of ligand-induced receptor crosslinking. The rate constant k_{+2} characterizes receptor crosslinking, and the rate constant k_{off} characterizes dissociation of ligand-receptor bonds. The value of β was adjusted by varying k_{+2} while holding $N_R = 300$ and $k_{off} = 0.01\ s^{-1}$ fixed. In each panel, the y-axis indicates the normalized total CPU time per reaction event required to simulate the kinetics of the TLBR model from time $t = 0$ to 1000 s with all ligand initially free. Parameters used are the same as those reported for Fig. 3 of Colvin et al. [14]. See the BioNetGen input file testcase2a.bngl (Additional file 2).

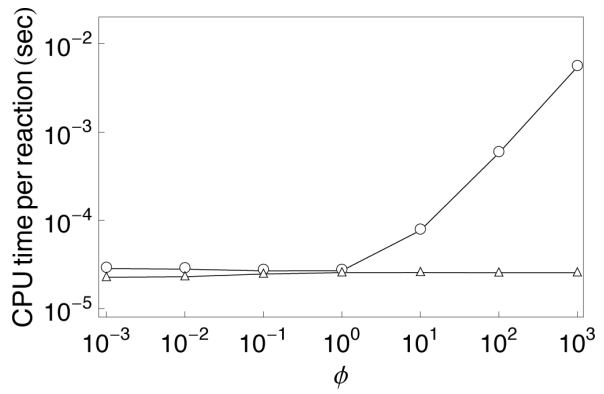

Figure 4 RuleMonkey is efficient for simulation of networks with fast reactions. The model considered here is specified in Additional file 4 (stiff.bngl). RuleMonkey (triangles) is compared with DYNSTOC (open dots). The y-axis indicates the total CPU time per reaction event required to simulate the kinetics of two first-order reactions. The x-axis indicates the value of φ, the ratio between the rate constants that characterize the two reactions. In RuleMonkey, the time step is sampled from an exponential distribution scaled by the total reaction rate (Eq. 5). In contrast, in DYNSTOC, the time step is fixed and limited by the rate of the fastest reaction [14]. This difference in how the time step is selected accounts for the performance differences seen for cases where $\varphi \gg 1$.

(ODEs) for mass-action kinetics. Although BioNetGen has been used to study a number of systems [22-24,45,50], the reaction network implied by a set of rules is often so large as to make the simulation approaches implemented in BioNetGen impractical. Network generation is expensive, as is simulation of a large-scale reaction network using a conventional method. Such methods have a computational cost that depends on the size of the network being simulated. In the generate-first and on-the-fly approaches to simulating rule-based models [9,21,25,35,48], which are implemented in BioNetGen, there is both a network generation cost and a network simulation cost, because network generation is a prerequisite for network simulation via these approaches. In contrast, in network-free simulation approaches [14,15,18-20], such as the method presented here and implemented in RuleMonkey, there is only a network simulation cost. However, it should be noted that network-free simulation procedures have a relatively high cost per reaction event, as is evident in Table 1. Thus, one should choose such a simulation procedure only when the cost of an alternative procedure that relies on network-generation is prohibitive.

The cost of network generation is often prohibitive, which has motivated the development of several network-free simulation methods [14,15,18-20]. Like these methods, the method presented here and implemented in RuleMonkey avoids network generation and, consequently, has a computational cost independent of the number of reactions implied by a set of rules (Figs. 2 and 3). We note that Yang and Hlavacek [20] have described a method that is essentially the same as that implemented in RuleMonkey but without providing details about how the method could be implemented for general use (i.e., beyond problem-specific demonstrations of the method) and without providing general-purpose software.

RuleMonkey and DYNSTOC [14] are similar in that they can both simulate BNGL-encoded models while avoiding network generation. However the method implemented in RuleMonkey is a rejection-free method, whereas the method implemented in DYNSTOC is a null-event method. Both types of methods are capable of producing correct simulation results but null-event methods tend to have a higher cost for simulation of systems with fast reactions [51]. For cases that we have examined, RuleMonkey typically performs better than DYNSTOC (Table 1) and has a decided advantage for systems with fast reactions (Fig. 4). This advantage is perhaps significant because practical problems tend to involve reactions occurring on disparate time scales.

The capability of RuleMonkey to correctly process BNGL-encoded model and simulation specifications (as illustrated in Fig. 2) is significant in that this capability allows RuleMonkey to be used to simulate a wide array of models that account for the site-specific details of protein-protein interactions (e.g., multisite phosphorylation) as well as the connectivity of proteins in protein complexes (e.g., cyclic aggregates of proteins can be distinguished from acyclic aggregates). RuleMonkey also distinguishes between intra- and intermolecular reactions, which is important for correct simulation results in many cases [19,28]. We have demonstrated that RuleMonkey can be used to efficiently simulate rule sets that imply large-scale chemical reaction networks (Table 1; Figs. 2 and 3). The capabilities of RuleMonkey complement those of BioNetGen and other available software tools for simulation of rule-based modeling [14,35,52-56]. In the future, we expect RuleMonkey will prove useful for simulating models of signal-transduction systems that are significantly larger and more comprehensive than the rule-based models of such systems that have so far been considered [22-24,45-47].

Availability and requirements

- **Project name**: RuleMonkey
- **Project home page**: http://public.tgen.org/rulemonkey
- **Operating system(s)**: Platform independent
- **Programming language**: C
- **Other requirements**: None
- **License**: GNU General Public License (GPL), version 3

- **Any restrictions to use by non-academics**: If the terms of the GPL are unacceptable, it may be possible to provide the software under a different license.

Additional material

Additional file 1: This plain-text file specifies a simulation of the multisite phosphorylation model introduced by Colvin et al. [14]. This simulation is considered in Fig. 2A and Table 1. The file can be processed by RuleMonkey and other BNGL-compatible tools for simulating rule-based models.

Additional file 2: This plain-text file specifies a simulation of a model for trivalent ligand-bivalent cell-surface receptor interaction (the so-called TLBR model) introduced by Yang et al. [19]. This simulation is considered in Fig. 3. The file can be processed by RuleMonkey and other BNGL-compatible tools for simulating rule-based models.

Additional file 3: This plain-text file specifies a simulation of a model for trivalent ligand-bivalent cell-surface receptor interaction (the so-called TLBR model) introduced by Yang et al. [19]. This simulation is considered in Fig. 2B and Table 1. The difference between the files testcase2a.bngl and testcase2b.bngl is in parameter values. The file can be processed by RuleMonkey and other BNGL-compatible tools for simulating rule-based models.

Additional file 4: This plain-text file specifies a simulation of reaction events occurring on disparate time scales. This simulation is considered in Fig. 4 and Table 1. The file can be processed by RuleMonkey and other BNGL-compatible tools for simulating rule-based models.

Additional file 5: This plain-text file specifies a simulation of a model for pentavalent ligand-trivalent cell-surface receptor interactions. The model is based on assumptions similar to those upon which the TLBR model [19] and the model of Goldstein and Perelson [49] are based. This simulation is considered in Table 1. The file can be processed by RuleMonkey and other BNGL-compatible tools for simulating rule-based models.

Additional file 6: This plain-text file specifies a simulation of the model of Blinov et al. [45] for early events in epidermal growth factor receptor (EGFR) signaling. This simulation is considered in Table 1. The file can be processed by RuleMonkey and other BNGL-compatible tools for simulating rule-based models.

Additional file 7: This plain-text file specifies a simulation of the model of Goldstein et al. [46] and Faeder et al. [47] for early events in IgE receptor (FcεRI) signaling. This simulation is considered in Table 1. The file can be processed by RuleMonkey and other BNGL-compatible tools for simulating rule-based models.

Additional file 8: This plain-text file specifies a simulation of the model of Nag et al. [44] for crosslinking of phosphorylated LAT molecules by intracellular Grb2-Sos1 complexes. This simulation is considered in Table 1. The file can be processed by RuleMonkey and other BNGL-compatible tools for simulating rule-based models.

Acknowledgements
We thank J. Yang for helpful discussions and B. Hu for integrating RuleMonkey into GetBonNie. This work was supported by National Institutes of Health grants AI35997, CA109552, NS042262, GM076570; DOE contract DE-AC52-06NA25396; and the Arizona Biomedical Research Commission.

Author details
[1]Clinical Translational Research Division, Translational Genomics Research Institute, Phoenix, AZ 85004, USA. [2]Theoretical Biology and Biophysics Group, Theoretical Division and Center for Nonlinear Studies, Los Alamos National Laboratory, Los Alamos, NM 87545, USA. [3]Department of Biology, University of New Mexico, Albuquerque, NM 87131, USA. [4]Department of Biological Sciences, Northern Arizona University, Flagstaff, AZ 86011, USA.

Authors' contributions
JC implemented the algorithm and designed the tool. MIM and RNG tested and validated the tool and evaluated its performance. DVH, RGP and WSH planned the project. RGP and WSH conceived the tool and wrote the manuscript. All authors have read and approved the final version of the manuscript.

Received: 21 June 2010 Accepted: 30 July 2010 Published: 30 July 2010

References
1. Hunter T: **Signaling–2000 and beyond.** *Cell* 2000, **100**:113-127.
2. Scott JD, Pawson T: **Cell signaling in space and time: where proteins come together and when they're apart.** *Science* 2009, **326**:1220-1224.
3. Hanahan D, Weinberg RA: **The hallmarks of cancer.** *Cell* 2000, **100**:57-70.
4. Hunter T: **Treatment for chronic myelogenous leukemia: the long road to imatinib.** *J Clin Invest* 2007, **117**:2036-2043.
5. Kholodenko BN: **Cell-signalling dynamics in time and space.** *Nat Rev Mol Cell Biol* 2006, **7**:165-176.
6. Aldridge BB, Burke JM, Lauffenburger DA, Sorger PK: **Physicochemical modelling of cell signaling pathways.** *Nat Cell Biol* 2006, **8**:1195-1203.
7. Chakraborty AK, Das J: **Pairing computation with experimentation: a powerful coupling for understanding T cell signalling.** *Nat Rev Immunol* 2010, **10**:59-71.
8. Breitling R, Hoeller D: **Current challenges in quantitative modeling of epidermal growth factor signaling.** *FEBS Lett* 2005, **579**:6289-6294.
9. Hlavacek WS, Faeder JR, Blinov ML, Posner RG, Hucka M, Fontana W: **Rules for modeling signal-transduction systems.** *Sci STKE* 2006, **2006**:re6.
10. Hlavacek WS, Faeder JR: **The complexity of cell signaling and the need for a new mechanics.** *Sci Signal* 2009, **2**:pe46.
11. Hlavacek WS, Faeder JR, Blinov ML, Perelson AS, Goldstein B: **The complexity of complexes in signal transduction.** *Biotechnol Bioeng* 2003, **84**:783-794.
12. Danos V, Feret J, Fontana W, Harmer R, Krivine J: **Rule-based modelling of cellular signalling.** *Lect Notes Comput Sci* 2007, **4703**:17-41.
13. Mayer BJ, Blinov ML, Loew LM: **Molecular machines or pleiomorphic ensembles: signaling complexes revisited.** *J Biol* 2009, **8**:81.
14. Colvin J, Monine MI, Faeder JR, Hlavacek WS, Von Hoff DD, Posner RG: **Simulation of large-scale rule-based models.** *Bioinformatics* 2009, **25**:910-917.
15. Morton-Firth CJ, Bray D: **Predicting temporal fluctuations in an intracellular signalling pathway.** *J Theor Biol* 1998, **192**:117-128.
16. Shimizu TS, Bray D: **Computational cell biology–the stochastic approach.** *Foundations of Systems Biology* Cambridge, MA: MIT PressKitano H 2001, **Ch 10.**
17. **The DYNSTOC web site.** [http://public.tgen.org/dynstoc/].
18. Danos V, Feret J, Fontana W, Krivine J: **Scalable simulation of cellular signaling networks.** *Lect Notes Comput Sci* 2007, **4807**:139-157.
19. Yang J, Monine MI, Faeder JR, Hlavacek WS: **Kinetic Monte Carlo method for rule-based modeling of biochemical networks.** *Phys Rev E* 2008, **78**:031910.
20. Yang J, Hlavacek WS: **Rejection-free kinetic Monte Carlo simulation of multivalent biomolecular interactions.**[http://arxiv.org/abs/0812.4619].
21. Faeder JR, Blinov ML, Hlavacek WS: **Rule-based modeling of biochemical systems with BioNetGen.** *Methods Mol Biol* 2009, **500**:113-167.
22. Barua D, Faeder JR, Haugh JM: **Structure-based kinetic models of modular signaling protein function: focus on Shp2.** *Biophys J* 2007, **92**:2290-2300.
23. Barua D, Faeder JR, Haugh JM: **Computational models of tandem Src homology 2 domain interactions and application to phosphoinositide 3-kinase.** *J Biol Chem* 2008, **283**:7338-7345.
24. Barua D, Faeder JR, Haugh JM: **A bipolar clamp mechanism for activation of Jak-family protein tyrosine kinases.** *PLoS Comput Biol* 2009, **5**:e1000364.
25. Faeder JR, Blinov ML, Goldstein B, Hlavacek WS: **Rule-based modeling of biochemical networks.** *Complexity* 2005, **10**:22-41.
26. Faeder JR, Blinov ML, Hlavacek WS: **Graphical rule-based representation of signal-transduction networks.** *Proceedings of the 2005 ACM Symposium on Applied Computing: 13-17 March 2005; Santa Fe, NM* ACM PressLiebrock LM 2005, 133-140.

27. Blinov ML, Yang J, Faeder JR, Hlavacek WS: **Graph theory for rule-based modeling of biochemical networks.** *Lect Notes Comput Sci* 2006, **4230**:89-106.
28. Monine MI, Posner RG, Savage PB, Faeder JR, Hlavacek WS: **Modeling multivalent ligand-receptor interactions with steric constraints on configurations of cell-surface receptor aggregates.** *Biophys J* 2010, **98**:48-56.
29. Blinov ML, Faeder JR, Goldstein B, Hlavacek WS: **BioNetGen: software for rule-based modeling of signal transduction based on the interactions of molecular domains.** *Bioinformatics* 2006, **20**:3289-3291.
30. Koschorreck M, Conzelmann H, Ebert S, Ederer M, Gilles ED: **Reduced modeling of signal transduction–a modular approach.** *BMC Bioinformatics* 2007, **8**:336.
31. Conzelmann H, Fey D, Gilles ED: **Exact model reduction of combinatorial reaction networks.** *BMC Syst Biol* 2008, **2**:78.
32. Conzelmann H, Gilles ED: **Dynamic pathway modeling of signal transduction networks: a domain-oriented approach.** *Methods Mol Biol* 2008, **484**:559-578.
33. Borisov NM, Chistopolsky AS, Faeder JR, Kholodenko BN: **Domain-oriented reduction of rule-based network models.** *IET Syst Biol* 2008, **2**:342-351.
34. Feret J, Danos V, Krivine J, Harmer R, Fontana W: **Internal coarse-graining of molecular systems.** *Proc Natl Acad Sci USA* 2009, **106**:6453-6458.
35. Lok L, Brent R: **Automatic generation of cellular reaction networks with Moleculizer 1.0.** *Nat Biotechnol* 2005, **23**:131-136.
36. Gillespie DT: **Stochastic simulation of chemical kinetics.** *Annu Rev Phys Chem* 2007, **58**:35-55.
37. Voter AF: **Introduction to the kinetic Monte Carlo method.** *Radiation Effects in Solids* Dordrecht, The Netherlands: SpringerSickafus KE, Kotomin EA, Uberuaga BP 2007, **Ch 1.**
38. Blue JL, Beichl I, Sullivan F: **Faster Monte Carlo simulations.** *Phys Rev E* 1995, **51**:R867-R868.
39. Gibson MA, Bruck J: **Efficient exact stochastic simulation of chemical systems with many species and many channels.** *J Phys Chem A* 2000, **104**:1876-1889.
40. **The RuleMonkey web site.** [http://public.tgen.org/rulemonkey/].
41. Hu B, Fricke GM, Faeder JR, Posner RG, Hlavacek WS: **GetBonNie for building, analyzing and sharing rule-based models.** *Bioinformatics* 2009, **25**:1457-1460.
42. **The GetBonNie web site.** [http://getbonnie.org].
43. **The BioNetGen web site.** [http://bionetgen.org].
44. Nag A, Monine MI, Faeder JR, Goldstein B: **Aggregation of membrane proteins by cytosolic cross-linkers: theory and simulation of the LAT-Grb2-SOS1 system.** *Biophys J* 2009, **96**:2604-2623.
45. Blinov ML, Faeder JR, Goldstein B, Hlavacek WS: **A network model of early events in epidermal growth factor receptor signaling that accounts for combinatorial complexity.** *BioSystems* 2006, **83**:136-151.
46. Goldstein B, Faeder JR, Hlavacek WS, Blinov ML, Redondo A, Wofsy C: **Modeling the early signaling events mediated by FcεRI.** *Mol Immunol* 2002, **38**:1213-1219.
47. Faeder JR, Hlavacek WS, Reischl I, Blinov ML, Metzger H, Redondo A, Wofsy C, Goldstein B: **Investigation of early events in FcεRI-mediated signaling using a detailed mathematical model.** *J Immunol* 2003, **170**:3769-3781.
48. Blinov ML, Faeder JR, Yang J, Goldstein B, Hlavacek WS: **'On-the-fly' or 'generate-first' modeling?** *Nat Biotechnol* 2005, **23**:1344-1345.
49. Goldstein B, Perelson AS: **Equilibrium theory for the clustering of bivalent cell surface receptors by trivalent ligands. Application to histamine release from basophils.** *Biophys J* 1984, **45**:1109-1123.
50. Faeder JR, Blinov ML, Goldstein B, Hlavacek WS: **Combinatorial complexity and dynamical restriction of network flows in signal transduction.** *Syst Biol* 2005, **2**:5-15.
51. Chatterjee A, Vlachos DG: **An overview of spatial microscopic and accelerated kinetic Monte Carlo methods.** *J Computer-Aided Mater Des* 2007, **14**:253-308.
52. Meier-Schellersheim M, Xu X, Angermann B, Kunkel EJ, Jin T, Germain RN: **Key role of local regulation in chemosensing revealed by a new molecular interaction-based modeling method.** *PLoS Comput Biol* 2006, **2**: e82.
53. Moraru II, Schaff JC, Slepchenko BM, L BM, Morgan F, Lakshminarayana A, Gao F, Li Y, Loew LM: **Virtual Cell modelling and simulation software environment.** *IET Syst Biol* 2008, **2**:352-362.
54. Mallavarapu A, Thomson M, Ullian B, Gunawardena J: **Programming with models: modularity and abstraction provide powerful capabilities for systems biology.** *J R Soc Interface* 2009, **6**:257-270.
55. Lis M, Artyomov MN, Devadas S, Chakraborty AK: **Efficient stochastic simulation of reaction-diffusion processes via direct compilation.** *Bioinformatics* 2009, **25**:2289-2291.
56. Andrews SS, Addy NJ, Brent R, Arkin AP: **Detailed simulations of cell biology with Smoldyn 2.1.** *PLoS Comput Biol* 2010, **6**:e1000705.

doi:10.1186/1471-2105-11-404
Cite this article as: Colvin *et al.*: RuleMonkey: software for stochastic simulation of rule-based models. *BMC Bioinformatics* 2010 **11**:404.