HHS Public Access

Author manuscript

Bull Math Biol. Author manuscript; available in PMC 2020 August 01.

Published in final edited form as:

Bull Math Biol. 2019 August ; 81(8): 2822–2848. doi:10.1007/s11538-018-0418-2.

Generalizing Gillespie's direct method to enable network-free simulations

Ryan Suderman,
Theoretical Biology and Biophysics Group, Theoretical Division, Center for Nonlinear Studies, Los Alamos National Laboratory

Eshan D. Mitra,
Theoretical Biology and Biophysics Group, Theoretical Division

Yen Ting Lin,
Theoretical Biology and Biophysics Group, Theoretical Division, Center for Nonlinear Studies, Los Alamos National Laboratory

Keesha E. Erickson,
Theoretical Biology and Biophysics Group, Theoretical Division

Song Feng,
Theoretical Biology and Biophysics Group, Theoretical Division, Center for Nonlinear Studies, Los Alamos National Laboratory

William S. Hlavacek
Theoretical Biology and Biophysics Group, Theoretical Division, Center for Nonlinear Studies, Los Alamos National Laboratory

Abstract

Gillespie's direct method for stochastic simulation of chemical kinetics is a staple of computational systems biology research. However, the algorithm requires explicit enumeration of all reactions and all chemical species that may arise in the system. In many cases, this is not feasible due to the combinatorial explosion of reactions and species in biological networks. Rule-based modeling frameworks provide a way to exactly represent networks containing such combinatorial complexity, and generalizations of Gillespie's direct method have been developed as simulation engines for rule-based modeling languages. Here, we provide both a high-level description of the algorithms underlying the simulation engines, termed network-free simulation algorithms, and how they have been applied in systems biology research. We also define a generic rule-based modeling framework and describe a number of technical details required for adapting Gillespie's direct method for network-free simulation. Finally, we briefly discuss potential avenues for advancing network-free simulation and the role they continue to play in modeling dynamical systems in biology.

Keywords

stochastic simulation; rule-based modeling; combinatorial complexity; kinetic Monte Carlo

1 Introduction

In a living cell, numerous biochemical species interact with each other, forming complex reaction networks. How the cell functions is largely determined by the dynamics of these networks. One of the goals of systems biology is to understand the emergence of phenotypes from the complex interactions present in these reaction networks.

Mathematical modeling and simulation are powerful tools for studying biochemical reaction networks. The interactions between chemical species can be rigorously defined and simulated using diverse techniques for representing nonlinear dynamical systems. Exploratory analyses and hypothesis testing can be performed efficiently in a computational setting. As a result, computational systems biology has been thriving over the past decade and has become a recognized field within quantitative biology.

1.1 Modeling chemical reaction networks

Traditionally, reaction networks have been modeled with systems of ordinary differential equations (ODEs) that are solved with numerical integration algorithms. These models are typically constructed on the basis of mass action kinetics. In many cases, this approach is appropriate and successful (Le Novere et al, 2006; Deuflhard and Röblitz, 2015). However, the ODEs describe the well-mixed concentrations of chemical species and may be inappropriate in the context of a cell because chemical species may be present in low copy numbers. A typical eukaryotic cell has a small volume, on the order of picoliters, and the enclosed chemical species have finite populations. Consequently, intrinsic noise (Elowitz et al, 2002; Kepler and Elston, 2001; Ozbudak et al, 2002; Blake et al, 2003; Thattai and Van Oudenaarden, 2004; Kærn et al, 2005; Acar et al, 2008; Munsky et al, 2009; Lin and Doering, 2016; Lin and Galla, 2016; Hufton et al, 2016; Lin and Buchler, 2017; Lin et al, 2017) due to the finite and discrete nature of the reactants could be an important factor to consider when studying the dynamics of intracellular reaction networks.

Stochastic and discrete-state models provide a sensible solution to describe the fundamentally stochastic biochemical reactions between finite and discrete reactants. Among them, continuous-time Markov chains (CTMC) have become a standard way to represent and simulate biochemical reaction networks. Formally, the joint probability distribution of a CTMC is described by a Chemical Master Equation (CME). Except for special cases, it is impossible to derive the full analytical solution and efficient numerical techniques must be employed to solve the CMEs. There are two ways to numerically solve the CME: the first way is to directly integrate or approximate the CMEs (Munsky and Khammash, 2006; Cao et al, 2016), and the second way is to use various continuous-time Monte Carlo techniques to generate sample paths of the random processes and use the sample paths to compute statistical quantities of interests (Bortz et al, 1975; Gillespie, 1976; Gillespie et al, 1977). In this review, we focus on the second approach.

Using Monte Carlo approaches to sample trajectories from dynamical systems can be traced back to the late 1940's in Los Alamos where Enrico Fermi and Robert D. Richtmyer (Fermi and Richtmyer, 1948) and Nicolas Metropolis and Stanislaw M. Ulam (Metropolis and Ulam, 1949) proposed the independent random sampling idea to solve problems in kinetic theory (Boltzmann and Fokker–Planck equations). While a dynamical interpretation was clearly given in (Metropolis and Ulam, 1949), the proposal soon evolved to the famous Metropolis algorithm (Metropolis et al, 1953), which did not capture time dependence and focused instead on the equilibrium distribution of a thermal system. In the 1960's, several seminal papers generalized the idea of Monte Carlo sampling to dynamical processes in continuous time (Cox and Miller, 1965; Young and Elcock, 1966). Soon, the technique was applied to various problems from material science to statistical physics (a review of topics can be found in (Voter, 2007)). The first rejection-free algorithm, the n-fold BKL algorithm, was proposed by Bortz, Kalos, and Lebowitz in 1975 (Bortz et al, 1975). With this algorithm, the advanced time is a function of all the possible transition events which may not take place in a specific sample path. Soon, Gillespie introduced such rejection-free kinetic Monte Carlo methods to the study of chemical reaction networks (Gillespie, 1976). The method became a standard way to simulate such networks and is often termed Gillespie's algorithm or the stochastic simulation algorithm (SSA). Ever since Gillespie's paper, there have been many proposals to improve the performance of the SSA by introducing novel data structures, such as indexed priority queues (Gibson and Bruck, 2000) or constructing an implementation that takes advantage of some general structure of the reaction network, such as a sparse transition matrix (Ramaswamy and Sbalzarini, 2010); however, the core of the algorithm remains intact.

An intrinsic, often unstated, assumption of the SSA is that the chemical reaction network must be fully specified. However, in biology we commonly only have data about pairwise interactions. Further complicating the situation is the fact that biochemical reaction networks often involve macromolecules (e.g., proteins) that have multiple domains for interacting with other molecules and that may each occupy a number of chemical states. For example, a protein may have multiple residues that are subject to phosphorylation or dephosphorylation. The platelet-derived growth factor receptor (PDGFR) has an intracellular domain containing 10 amino acid residues that may be phosphorylated. If each residue is independently subject to phosphorylation and dephosphorylation, a single PDGFR molecule can occupy $2^{10} = 1024$ possible biochemical states (Mayer et al, 2009). However, the PDGFR dimerizes upon binding a ligand, increasing biochemical state space to over 500,000 possible states (Mayer et al, 2009). Representing the PDGFR as a system of differential equations would thus require over 500,000 equations. Similarly, biochemical species can form large complexes through pairwise binding, and in some cases the total number of possible biochemical species exponentially increases with the number of distinct proteins in the model (Suderman and Deeds, 2013; Faeder et al, 2005a; Bray, 2003; Endy and Brent, 2001). This vastness in state space is termed combinatorial complexity.

1.2 Representing complex systems simply

Rule-based modeling was introduced in systems biology to address the problem of combinatorial complexity, particularly for models of intracellular cell signaling networks

(Chylek et al, 2014b; Danos et al, 2007a). Its core insight is the use of rules to represent a class of reactions that operate on identical reaction centers. Specifically, rules use molecular patterns common among multiple reactant molecules, allowing for more succinct representation of reactions, which precludes the need to enumerate all possible reactions between all possible biochemical species (Chylek et al, 2014b). Rule-based modeling enables a precise and comprehensive, yet tractable, representation of complex systems (de Oliveira, Luís P. et al, 2016). With a rule-based approach, a modeler can represent the previously described >500,000-state PDGFR system with a mere 22 rules, assuming mutual independence of the rules involved (Code 1): 2 ligand-receptor interaction rules (binding and unbinding), 2 receptor dimerization rules (binding and unbinding), 10 phosphorylation rules and 10 dephosphorylation rules (one of each per residue). The complete formulation of this model, written in the BioNetGen language (Faeder et al, 2009), is given in Appendix A.

Rule-based modeling in systems biology was initially developed to automate construction of biochemical reaction networks, as describing biological phenomena with rules significantly reduces the work of model construction (Blinov et al, 2006). However, the resulting network can be of intractable size from a computational perspective, or it may even be unbounded except by the number of molecules present in the system (Yang et al, 2008). A natural generalization to mitigate this issue is to generate the network on-the-fly (Lok and Brent, 2005; Faeder et al, 2005b; Faulon and Sault, 2001). Even if the number of chemical species can be (possibly infinitely) large, on-the-fly network generation assumes that only a finite (and sometimes small) number of species are typically populated. Therefore, instead of insisting on calculating reaction rates for all possible reactions, which may be impossible, one generates the local network (and thus determines reaction rates) for the reactions whose reactants are present in the system. As the system evolves, this local network is updated when new species are introduced or existing species are removed completely. While sometimes useful, the size of the boundary (the part of the network requiring generation of new reactions) may increase exponentially, and so simulation becomes very inefficient in such a case.

This motivates an approach that avoids generating the chemical reaction network altogether: instead, each molecule is treated as an object and the simulation is performed directly on the objects. This *network-free* approach, is a form of agent-based simulation grounded in physical and chemical principles. Various network-free software packages have been deployed throughout the years, most of which correspond to a particular rule-based modeling framework (Yang et al, 2008; Colvin et al, 2010; Le Novere and Shimizu, 2001; Colvin et al, 2009; Danos et al, 2007b; Sneddon et al, 2011; Zhang et al, 2005; Sweeney et al, 2008). The algorithm implemented in these simulators is a modified form of Gillespie's direct method (Gillespie, 1976; Gillespie et al, 1977). The modified algorithm differs from Gillespie's method in that careful bookkeeping of the state of the system is required.

Throughout the rest of this work, we provide a brief review of research applications of existing implementations of network-free simulation algorithms (Section 2) followed by a high-level description of how Gillespie's direct method can be generalized for a rule-based modeling framework (Section 3). We then define a generic rule-based modeling framework and proceed to describe a basic implementation of a network-free simulation algorithm

(Sections 4 and 5). Finally, we investigate a number of technical details that need consideration when implementing such an algorithm (Sections 6 and 7).

2 Applications of network-free simulation

A number of software packages exist to facilitate network-free simulation for specific rule-based modeling languages. Two of the most prominent rule-based modeling languages for biomolecular simulation are the Kappa language (Danos and Laneve, 2004) and the BioNetGen language (BNGL) (Faeder et al, 2009). These languages each have associated network-free simulation tools: KaSim for the Kappa language (Boutillier et al, 2017b) and NFsim (Sneddon et al, 2011) (among others) for BNGL. Both the KaSim and NFsim engines have been used in situations where model complexity necessitates network-free simulation.

A straightforward application of network-free simulation seeks to understand how combinatorial complexity influences the protein complexes that assemble during signal transduction. Explicitly investigating combinatorial complexity precludes any notion of species or reaction network enumeration, and so network-free simulation is the only path forward. Deeds et al. (Deeds et al, 2012) constructed a rule-based model in the Kappa language that characterized the protein-protein interaction network in yeast. They found that existing knowledge of the yeast interaction network led to extreme combinatorial complexity. Ultimately, they found that simulations beginning from the same initial state diverged rapidly in the sets of complexes that were formed, where any two independent simulations exhibited only 20% overlap among unique complexes formed (Deeds et al, 2012). The same methodology was applied to the pheromone signaling network in yeast, this time revealing that reliable signal transduction can occur in networks even when the assembled molecular complexes responsible for signaling are highly variable (Suderman and Deeds, 2013). These bodies of work highlight two key facts about complexity in interaction networks: first, that clonal cells are likely never in the same state at the same time, and second, that cells can still reliably process extracellular information despite a seemingly chaotic environment.

Intracellular signaling in mammalian cells often involves oligomerization or polymerization of cell surface receptors (Su et al, 2016). These receptors also recruit a wide array of downstream effector proteins that govern the dynamics of signal transduction. Examples of these systems include growth factor signaling (Creamer et al, 2012; Stites et al, 2015) and the antigen recognition receptor signaling (Chylek et al, 2014a; Nag et al, 2010, 2009). Because of the formation of complex multimeric structures, the state space of these intracellular signaling models is bounded only by the number of molecules in the simulation. Using network-free simulation algorithms is often the only approach available to characterize system dynamics without undue simplification of a model. Most of the work on mammalian signaling referenced here used models written in BNGL and were simulated with a BNGL-compatible network-free simulator, NFsim.

Another domain of research involving biopolymers considers nucleic acids, and one example of nucleic acid research facilitated by rule-based modeling examined the phosphorylation

states of RNA Polymerase II as it binds to an arbitrary number of positions on a DNA sequence (Aitken et al, 2013). In this case, the number of possible biochemical states is susceptible to the problem of combinatorial complexity as the size of the DNA molecule increases. A second example focused on base excision repair in DNA (Köhler et al, 2014). In this work, the model includes a single DNA molecule composed of 100,000 base pairs, something that would be utterly intractable in any other modeling framework. The model incorporates a number of protein-protein interactions in addition to a mechanistic representation of base repair catalysis involving endonucleases, polymerases, and ligases to investigate how certain molecules (such as scaffold proteins) contribute to the speed and efficacy of DNA repair.

Finally, rule-based modeling also has applications outside biology. One example explores the usefulness of applying rule-based modeling and network-free simulation in simulating labor markets (Kühn and Hillmann, 2016). In particular, it highlights the differences between general agent-based modeling frameworks and rule-based modeling, which is a type of agent-based modeling that is based on formal chemical kinetics (Danos et al, 2007b; Faeder et al, 2009). A notable difference between existing agent-based modeling frameworks for modeling labor markets and rule-based modeling coupled to network-free simulation is the absence of spatial information in network-free simulation (i.e., the well-mixed assumption). Although these examples serve as a reminder for the general applicability of rule-based modeling approaches and network-free simulation algorithms, we focus primarily on biological applications for a detailed description of the network-free simulation approach.

3 A minimalist description of network-free simulations

The algorithm to generate exact sample paths for a rule-based model using a network-free approach is largely similar to Gillespie's direct method as seen in Fig. 1 (Gillespie, 1976; Gillespie et al, 1977). In this section, we focus on a high-level general description of the algorithm, leaving the technical (but important) details specific to network-free simulations to the next sections. This description includes some jargon that we define more rigorously in Section 4:

- a molecule is an explicit object that is tracked by the simulation engine
- a rule is composed of patterns that identify molecular moieties in reactant molecules
- a pattern is used to match reactant molecules during simulation
- a rule defines a chemical transformation that is to be applied to a reactant molecule
- a species defines a class of some molecular complex (i.e. a particular biomolecular configuration)
- a mixture explicitly defines all interacting molecules in the simulation

We decompose the algorithm into four essential blocks: initialization of the system, computation of the rules' rates, advancing time and sampling a rule, and updating the system.

Step 1. Initialization—At the beginning of the simulation, the system configuration is defined by the user. Individual instances of all the chemical species' component molecules are populated.

Step 2. Computation of rule rates—In this step, matches are constructed between the patterns defined in the rules and the explicit molecule instances in the simulation mixture. One of the major differences between rule-based models and traditional models is that chemical transformations are defined in terms of patterns instead of in terms of chemical species (Gillespie, 1976; Gillespie et al, 1977). This means that many distinct chemical species may participate as a reactant in a rule, provided each of these species matches the same reactant pattern of the rule. The rate of the rule is then proportional to the number of matches of its reactant patterns.

Step 3. Advancing time and sampling a rule with specific reactants—Once all the rules' rates are computed, the time to the next rule application is a random number sampled from an exponential distribution whose rate parameter is the sum of all the rules' rates: r_T. A random waiting time Δt is sampled from this exponential distribution

$$\Delta t = \frac{-\log u_0}{r_T}$$

where u_0 is a uniform random number on the interval [0,1). The system's simulation time is then advanced by Δt. The rule to be applied to the system is sampled with probability proportional to its rate. This is done via the conditioning procedure:

- Sample a uniform random number u_1 on the interval [0,1).
- Iterate over all rules to find the minimum rule index, i, such that

$$\sum_{j=1}^{i \leq N} r_j > r_T \cdot u_1$$

where N is the number of rules.

These steps are identical to Gillespie's direct method (Gillespie, 1976; Gillespie et al, 1977). However, since the rules are defined by patterns, the sampled rule does not specify which molecule instances should be modified by the transformation defined by the rule. The simulator therefore samples molecules with uniform probability from a list of matches for each of the selected rule's reactant patterns.

Step 4. Updating the system's configuration—After the rule is applied (modifying the matched molecules sampled in Step 3) the matches between rule patterns and the molecules in the simulation mixture are updated. To increase simulation efficiency, the rules' rates can be updated incrementally to avoid recalculating all the matches. After the rules' rates are updated, the simulation continues from Step 3 until a stopping condition is met. We provide a detailed discussion of how this step can be implemented in Section 5.

Although the above description is very similar to Gillespie's original direct method (Gillespie, 1976; Gillespie et al, 1977), interfacing the algorithm with a particular rule-based modeling framework requires careful consideration of how pattern-matching affects a rule's rate. We discuss a number of relevant issues that may arise in building a network-free algorithm in Sections 6 and 7.

4 Nomenclature

In this and the following sections, we consider a number of technical details required for implementing a network-free simulation algorithm. To do so, we first must define the objects involved in the simulation. Different rule-based modeling frameworks use distinct nomenclature for similar, sometimes identical, constructs. Additionally, many descriptions of rule-based modeling approaches contain jargon that can easily confuse readers without the relevant domain-specific knowledge[1]. Here, we try to provide intuitive and sufficiently precise definitions of the objects required to construct a network-free simulation algorithm independent of any specific rule-based modeling framework. Note that our terminology reflects the usage of rule-based modeling in molecular and cellular biology (Chylek et al, 2014b; Danos et al, 2008).

The atomic unit of rule-based modeling applied to biochemical reactions is what we term the *molecule*. A molecule is typically a representation of a biological macromolecule, such as a protein, but a molecule could also represent a metabolite, drug, other small molecule, or any object of interest in the model. Rule-based modeling frameworks often require molecules to be first defined with a particular signature, and specific instances of these molecules (such as the molecules tracked during simulation) must conform to this definition. Such signatures are composed of a name that acts as a label for the molecule's type, as well as a predefined list of *sites* that typically represent physical or chemical attributes of the molecule (Fig. 2a). Sites are defined with names[2], and a predefined list (which can be empty) of internal states that represent any other property of the site (e.g., whether an amino acid residue in a protein is phosphorylated) (Fig. 2a). Sites also engage in binding to other sites, allowing association between molecules and thus representation of higher order molecular structures. Modification of sites' states (both internal and binding) typically comprise the majority of rule applications (reaction events) during simulation (e.g., binding, unbinding, or chemical modification).[3]

[1]Most objects in rule-based modeling can be represented visually (and formally) as graphs. Rules then become transformations (i.e., rewriting operations) on these graphs, and much of the jargon relating to rule-based modeling has its origin in graph theory. We will occasionally mention these terms, but will not rely on them.
[2]To our best knowledge, BioNetGen is the only framework that allows molecules with multiple identically-named sites. These sites are treated as equivalent. Molecule types must have unique names.
[3]Rules may also define synthesis and degradation of molecules.

Molecules that are used for representing specific molecular moieties and not specific instances of physical objects are termed *pattern molecules*. Sites in pattern molecules may be completely absent (unspecified) or a site's states may be partially specified to capture the necessary and sufficient features required for representing a molecular moiety. Of course, partial specification of a site's states (binding or internal) relies on syntax to convey incomplete knowledge about the site's states. For example, the modeling language used to write a rule may be able to express whether a site is bound, but its binding partner is unknown, or whether the site's internal state is in some subset of its predefined set of internal states. Furthermore, this partial specification has an ordering, where a site's absence or partial specification of its states in a pattern molecule is less specific than a site with a more specific or complete specification of its state (Fig. 2b). The notion of such an ordering is useful in pattern-matching; a less-specific object may match a more-specific or completely-specified object provided that the information in the less-specific object is preserved in the more-specific object. This concept of preservation of information is implicitly assumed in later discussions of partial specification of molecules in pattern matching. Similarly, if the internal state of one site is identical to another site and the binding states for the sites are both explicitly bound or explicitly free, then we say that the sites are equivalent. Sets of sites (and thus molecules) can similarly be ordered according to specificity or equivalency.

We define *complete molecules* or simply *molecules*, as molecules whose sites are all fully specified. Full specification requires that sites are either bound to another site with a labeled bond[4] or unbound, and that sites express an explicit internal state assuming that they have predefined internal states to occupy. Complete molecules are those that are involved in the simulation, being modified by rule applications.

We can now define larger objects composed of complete molecules and pattern molecues. An object composed of a list of one or more complete molecules that only connect to other molecules in their list (a *connected component* in graph terminology) is called a *species*, reflecting standard chemical nomenclature[5]. Note that a molecule with no bound sites is both a molecule and a species. Similarly, we define a *pattern* as a list of pattern molecules that are explicitly connected only amongst themselves (Fig. 2c, blue region elements)[6]. Finally we use the term *mixture* or *simulation mixture* to refer to the pool of complete molecules that are interacting during the simulation of a rule-based model (Fig. 2c, entire yellow region). Network-free simulation engines track individual objects in a mixture, as opposed to Gillespie's direct method, which tracks the populations of predefined chemical species. Note that populations of chemical species can be reconstructed from the list of complete molecules that defines a mixture, and so ours is an equivalent representation of the state of the system compared to the traditional SSA for the purpose of tracking population dynamics.

[4]A labeled bond links two sites, meaning that both partners in a bond can be determined by the bond label.
[5]In chemistry and ecology, the term *species* refers to a class of things (molecular configurations or organisms). We retain this convention and refer to specific *instances* of a species when discussing an individual object that conforms to the features that define a particular species.
[6]In some cases, patterns may be defined as involving unconnected molecules, but we do not adopt this convention

To avoid the need to a priori generate the reaction network, rates are calculated via pattern matching. We define a *match* as a mapping from a pattern to a list of molecules in a simulation mixture (Fig. 2c). This association depends on two criteria:

1. Each pattern molecule in the pattern must have a corresponding molecule of the same name in the mixture
2. For a pattern molecule X in the pattern and its corresponding molecule Y in the mixture:
    a. if X's sites are less specific than Y's sites, then the information present in X's sites must be preserved in Y's sites (consistency)
    b. sites with fully specified states present in X must have equivalent corresponding sites in Y

A *rule* defines a set of reactions and involves all of the previously defined concepts (Fig. 2d). A rule is composed of a list of reactant patterns and a list of product patterns that together define a transformation (Fig. 2d, blue regions), as well as a rate law. The reactant and product patterns are commonly known as the *left-hand side* and *right-hand side* of a rule, respectively. The rate law is used to calculate a rule's *rate*, which is the sum of all of the rates of all of the reactions implied by the rule. Each reaction implied by a rule inherits the rate law associated with the rule.

Finally, we use the term *system* to generally describe everything needed to fully represent and simulate the model.

5 Distinctions from traditional SSA

The generalization presented in Section 3 differs from a conventional SSA in a few ways. Two early examples of the general approach described here can be seen in (Yang et al, 2008) and (Danos et al, 2007b). Rates are dependent on numbers of matches instead of population sizes of species. Another related distinction is the need to sample molecules that will be altered by a rule once the rule has been selected for application. Perhaps the most notable and complicating difference is the methodology used to store and update the state of the system. This section provides a high-level view of the data structures necessary to accommodate these differences in a reasonable manner. We make no claims regarding the efficiency of this approach, but present it as a means to understand the essential steps of constructing a network-free simulation algorithm.

5.1 Computing the rule rates

To calculate the initial rates of the rules in a model, we first count the number of matches associated with each reactant pattern in a rule. A simple way to store this information is to have, for each reactant pattern in each rule, a *match list* containing all of the matches from a pattern to the molecules in the simulation mixture (Fig. 3a). To initialize the match list for each pattern, we select an arbitrary pattern molecule in the pattern, which we will refer to as the anchoring pattern molecule. We then find all molecules in the initial mixture that match the anchoring pattern molecule: the anchoring molecule. We recursively traverse the species

instance that contains the anchoring molecule (which is possible because the molecules contain pointers referencing the other molecules to which they are bound) to determine all the unique matches arising from the choice of anchoring molecule[7]. The set of matches forms the initial match list of the pattern. For rules with elementary rate laws (i.e., rate laws consistent with mass-action kinetics), the rate will typically be the rule's rate constant multiplied by the product of the numbers of matches for all reactant patterns. A simple example of this can demonstrated for rule R_1 (Fig. 2) from the information present in Fig. 3a:

1. Determine the number of matches for each reactant pattern
   - There are 2 matches for rule R_1, pattern P_1
   - There are 2 matches for rule R_1, pattern P_2
2. Calculate the product of the numbers of matches: $2 \cdot 2 = 4$
3. Calculate the rate by multiplying the product of matches by the rule's rate constant: $k_f \cdot 4$

However, there are a few issues to consider that may affect the calculation, and these issues are discussed in Section 6.

5.2 Sampling the reactants

Provided that a match list is correctly maintained for each reactant pattern in each rule, the sampling step of the simulation is straightforward (Fig. 3b). A rule is sampled according to the method described in Step 3 of Section 3. After selection of a rule, it is necessary to sample which molecules will be modified by the transformation defined by the rule. To make this choice, a match for each pattern in the rule is sampled with uniform probability from the corresponding match lists, and the rule is applied to the reactant molecules in the mixture that correspond to the selected match.

5.3 Updating the system

When a reaction event[8] occurs, the state of the system must be updated appropriately (Fig. 3c). If a transformation changes a site's internal state, that state must be updated on the molecule, and if the transformation includes the addition or removal of bonds, the molecules involved must have the pointers to their binding partners updated accordingly. Finally, the most computationally expensive step is to update the match lists. A brute force approach is to check all of the reactant patterns of all rules to determine if their match lists have been affected by the transformation that has just been applied.[9]

After a reaction event, the first phase of the update process, termed the negative update phase (in accordance with the terminology of (Danos et al, 2007b)), is to consider the molecules that were altered or removed by the reaction event. In this phase, we traverse the

[7]The anchoring molecule simply serves as an arbitrary starting point for traversing the species instance (i.e. multiple starting points may be possible).
[8]Nonproductive (null) events are described in Section 6.6.
[9]Depending on how patterns are specified, a brute force approach may be the only way to correctly update the system. See Section 6.5.

species instances containing the molecules that were involved in the reaction before the rule application[10], and remove the matches involving those species instances from all match lists.

The second phase, termed the positive update phase (in accordance with the terminology of (Danos et al, 2007b)), is to consider the newly created species instances formed by the reaction event. Similar to the negative update phase, the new species instances must be fully traversed to determine which rules' patterns need to be updated. Any new matches are then added to the appropriate match lists.

After the negative and positive update phases, the match lists have been fully updated, and the changes to the match lists can be used to efficiently update rule rates. Rate updates are most efficient if current rates are modified up or down in accordance with match list changes, as opposed to de novo rate calculation (i.e., calculating the rate by constructing the match lists from scratch).

6 Considerations for rate calculation

A number of issues arise as a result of using rules and a pattern matching algorithm instead of the traditional SSA's use of species' populations and reactions. Here, we discuss some of the more prominent issues relevant for accurate and consistent calculation of rule rates.

6.1 Symmetry among patterns

In cases where there is symmetry among a rule's reactant patterns (i.e., when a rule's reactant patterns contain two or more identical patterns), one must take care to correctly count the number of reactions that may occur. In Gillespie's original notation (Gillespie, 1976), h_μ is the number of ways a reaction may occur, and it is computed from the populations of the chemical species involved as reactants in the reaction. The product of h_μ and the reaction's rate constant is the reaction's rate. With our proposed method for tracking matches, we must similarly count the number of distinct match combinations in the presence of symmetry:

$$h_{sym,T} = \binom{|T|}{N_T}$$

where N_T is the number of times pattern T is present in the list of reactant patterns, and |*| denotes the size of T's match list. A simple example is a homodimerization rule with a rate constant k for some molecule A. If we consider a mixture with 1000 monomeric A molecules, then we can calculate the rule's rate:

$$\binom{1000}{2} \cdot k = \frac{1000 \cdot 999}{2} \cdot k$$

[10]This traversal, and that in the positive update phase, allows the algorithm to accommodate rules with nonlocal constraints. See Section 6.5.

6.2 Symmetry within a pattern

Patterns may be defined in such a way that multiple matches may arise from a simple permutation of the molecules involved in the match. For example, a pattern may involve two identical pattern molecules that are bound to each other on identically named sites. In such a case, the pattern may match the same set of molecules more than once. In Fig. 4a, an example is shown for a simple dimer dissociation reaction. A single dimer exists in the simulation mixture, and the reactant pattern matches the dimer such that the first pattern molecule matches molecule A_1 and the second pattern molecule matches molecule A_2 (blue). By permuting the molecules in the mixture (substituting A_1 for A_2 and vice versa), we find a second match between the reactant pattern and the dimer (red). Correctly calculating the rule's rate requires dividing the rate by the number of molecule permutations in the pattern that preserve bond connectivity.[11] The divisor similarly corresponds to the number of ways an occurrence of the pattern in a species instance is matched by the pattern because of symmetry in the pattern. This rate calculation assumes a specific convention about the semantics of our framework: a rule's rate should be proportional to the number of distinct reactions that can occur, and not simply the number of matches.[12] As a result of this choice of convention, pathological cases requiring explicit accommodation in either the simulation engine or rule interpreter may arise in dissociation reactions when asymmetric patterns match symmetric molecules (see Section 7.1).

6.3 Reaction path degeneracy

A pattern may have identical sites to which a transformation defined by a rule (e.g., modification of a site's internal state or formation/dissolution of a bond) may apply equally. This is termed reaction path degeneracy (IUPAC, 1997), referring to multiple equivalent ways for a reaction to occur with respect to some particular biochemical species. To calculate rule rates such that the rate constant refers to the reaction applied to an individual site (or pair of sites in the case of bond formation), the number of matches in the pattern's match list is multiplied by the number of equivalent sites in the pattern that are competent to be modified by the transformation defined by a rule (Fig. 4b). The multiplicative factor used to adjust the rate in cases of reaction path degeneracy may also be termed a statistical factor. During the rule application process, a random site is selected to undergo the transformation defined by a rule with uniform probability. In the case of Fig. 4b, the rate is $4 \cdot k_f$:

$$\left(\frac{2_1 \cdot 2_2 \cdot 2_4}{2_3}\right) \cdot k_f = 4 \cdot k_f$$

where the subscripts denote:

1. the number of matches of the A pattern
2. the number of matches of the B dimer pattern

[11] When representing patterns as graphs, this number is the order of the *automorphism group* of the graph, roughly the number of ways a graph can be mapped to itself.

[12] The BioNetGen language follows the number-of-reactions convention for rate calculation, whereas the Kappa language follows the number-of-matches convention.

3. the correction for symmetry in the B dimer pattern
4. the number of equivalent reaction paths[13]

Note that reaction path degeneracy often coincides with the presence of pattern-preserving site permutations or symmetry, and in these cases, the rate must be further adjusted as described in Section 6.2.

Regardless of the framework's chosen convention, the user must be aware of how the matching procedure may influence a rule's rate to correctly specify the rate constants in the rate laws associated with the rules (Section 8).

6.4 Molecularity

An additional consideration is the molecularity of rules. When species composed of more than one molecule exist in a simulation mixture, it is possible for multiple patterns in a single rule to match the same species instance. For example, suppose we have a bond formation rule stating that pattern molecule A with free site x may bind to pattern molecule B with free site y, and the mixture contains a trimeric complex, in which molecules A and B satisfy the patterns' constraints, but A and B are also each bound to a molecule C via sites unspecified in the bond formation rule (Fig. 5a). When this rule is sampled, the algorithm might randomly choose matches corresponding to the molecules A and B that are members of the trimeric complex. The rule states that A should bind to B, forming a cyclic structure, although this is not obvious from the rule's reactant pattern. Furthermore, the rate of cyclization (a unimolecular reaction) will not be equivalent to the rate of bimolecular association, and so a distinction should be made between intermolecular and intramolecular bond formation.

To distinguish between unimolecular and bimolecular reactions, the simulator must perform a check to confirm that matches for a bimolecular rule are members of distinct species instances and not members of the same species instance. This can be done by traversing the matched molecules' species instances as is done when updating the system after a reaction event; the task can also be accomplished by tracking species instances with unique identifiers (see Section 6.6).

6.5 Locality

Rules may include negative application conditions (i.e., constraints on when they may be applied). Such conditions are especially relevant when considering the molecularity of rules in the previous section (i.e., the bimolecular vs. unimolecular bond formation). It is sometimes possible to ignore molecularity constraints in the determination of matches between patterns and molecules (e.g., when intramolecular binding is not possible). Consider the rule set from Fig 2c and specifically the binding rule. Regardless of the sequence or number of reaction events in a mixture initially composed of A and B monomers, we know that the patterns in the first rule will only match monomeric A and B molecules without any additional enforced constraint that the transformation defined by a

[13]The reaction path degeneracy is 2 because binding to either y site on the B dimer pattern results in the same species instance.

rule be bimolecular (i.e., the rule will never result in an intramolecular binding event). When molecularity does not need to be checked, a rule's application is local, meaning that no information outside the molecules involved in the patterns' matches is required to correctly sample matches, apply rules, and update the system (Harmer et al, 2010).

A strongly contrasting phenomenon exists when allowing patterns to include implicit bonds, meaning that a rule-based modeling framework can enforce connectivity between molecules without specifying how the molecules are connected. If a rule contains implicit bonds, prediction of which rule rates to update cannot be done until after application of the rule to a specific species instance. We accommodate the presence of implicit bonds in our description of the update scheme (Section 5) by requiring new species to be checked against *all* rule patterns for new matches.

However, if no rules in a model involve implicit bonds, then more efficient updating schemes can be realized. One example is a structure that can be computed directly from the set of rules, and relates how the application of one rule may increase or decrease the rate of another rule[14] similar to a dependency graph for traditional SSA implementations (Danos et al, 2007b, 2009). For special case models where molecularity in rules does not need to be checked, the system can even be updated without the need to traverse the species instances involved in a particular reaction (Danos et al, 2007b).

6.6 Null events

In the previously described problem of constraining molecularity (Section 6.4), there may be occurrences in which the rates are overestimated, because some choices of matches lead to invalid reactions (Fig. 5a). To correct for these overestimates, some network-free simulation approaches allow null events.

If the sampled molecules fail to satisfy the molecularity constraints in the sampled rule (Fig. 5a), then the potential reaction event is rejected, time is updated, and the algorithm proceeds to the next iteration with an identical system configuration (Yang et al, 2008). The time update without a corresponding system update corrects for the overestimated rate; the null event uses up the excess rate resulting from the invalid combination of sampled matches.

This type of null event can become a serious computational inefficiency in models that tend to form large aggregates. Consider a bimolecular association rule. If most of the molecules in a system are part of the same, large aggregate (Fig. 5b), it is possible that the majority of iterations of the algorithm will choose two matches from within the same species instance. However, the rule requires an intermolecular bond to form, yielding a correspondingly large number of null events. Alternative implementations can be realized that avoid such issues (see Section 8).

A related type of null event may arise when sampling matches from a match list. Consider a homodimerization rule whose two patterns are identical. Upon sampling this rule and a match for the first pattern, it is possible that the same match may be sampled for the second

[14]This is termed the influence map in the Kappa language and associated simulation engines.

pattern. If the matches overlap (i.e., they involve the same molecules' sites) this results in a null event, termed a clash, where time is similarly updated and the system remains the same (Danos et al, 2007b). For example, Fig. 5b shows a dimerization rule. In this case both reactant patterns in the rule can match any molecule in the mixture. A clash could occur in this system when, during match sampling (Step 3 in Section 3), both matches involve the same molecule in the mixture (e.g., the selected matches for both reactant patterns point to A_6).

7 Accommodating rule conventions

7.1 Dissociation pathologies

Briefly mentioned in Section 6.2 was the possibility of pathological cases for rate calculations of dissociation rules. This occurs when a pattern that has no pattern-preserving site permutations (i.e., symmetries, see Section 6.2) matches the same set of molecules multiple times (Fig. 6a). This stems from our convention that a rule's rate should be proportional to the number of distinct reactions that can occur as opposed to the number of matches of the pattern in the mixture. Such pathologies do not occur when a rule's rate is proportional to the number of matches (Fig. 6c); however, assuming a number-of-matches convention detracts from the physical meaning of the rule's rate constant (generally considered to describe the rate of bond dissolution). The choice of which convention to follow is a matter of design; the important point is that the user understand which convention is in use to avoid writing rules that have unexpected consequences.

In our presented framework, these cases must be distinctly considered. One possible approach would be to match bonds instead of molecules for dissociation rules. This would involve an initial check of the rule set to determine whether this pathology may arise. The check would perform a bidirectional site specificity check[15] for the two connected pattern molecules whose bond would be broken by the rule. Patterns that satisfy this check may produce multiple molecular matches to the same set of molecules in the mixture. Upon identification of these patterns, one may then use appropriate data structures to track bond matches instead of molecule matches.

7.2 Identical site pathologies

In some rule-based modeling frameworks, individual molecules are allowed to have identical sites. In cases involving molecules with identical sites, further care must be taken to correctly calculate rule rates, which we do not explicitly discuss in our specification of a simulation engine. Consider the rule and mixture visualized in Fig. 6b. If the algorithm generates all possible matches between the pattern sites and the single molecule in the mixture, A_1, there would be 12 matches (the 2-permutations of the set of 4 sites). However if the prescribed rate constant refers to the rate at which a site x in molecule A is dephosphorylated, the rule's rate would be incorrectly calculated as $12 \cdot 1 \cdot k_{cat}$ instead of $4 \cdot 1 \cdot k_{cat}$.

[15]For each site in one of the pattern molecules, the corresponding site in the other pattern molecule must either be equivalent or less specific and consistent.

Additional machinery beyond what we describe here is needed to accommodate such a problem. If the language interprets the rule in Fig. 6b as having a rate of reaction equal to $4 \cdot 1 \cdot k_{cat}$, then a possible solution would be to introduce an additional matching procedure that distinguishes between the sites in the pattern or pattern molecule based on how the sites participate in the transformation defined by a rule. Sites that are modified by the transformation defined by a rule would be the primary matching sites (e.g., the phosphorylated site x in Fig. 6b). The other sites that provide context, but are unmodified by reaction would not contribute to additional matches (e.g., the partially specified site x in Fig. 6b). Of course if the modeler intends molecule A_1 to be dephosphorylated proportional to the number of matches, then the original formulation would be desired, with rate $12 \cdot 1 \cdot k_{cat}$. Being able to specify both types of rules would require specific language features, or some sort of annotation so that the simulation engine correctly interprets how a rule's rate should be calculated in the presence of identical sites.

In general, care must be taken when designing network-free simulation algorithms for specific modeling languages. Features or conventions present in the modeling language, such as those described in this section, can result in language-specific behaviors. This is especially evident when attempting to translate a model into a different rule-based modeling language (Suderman and Hlavacek, 2017). Documentation of pathological cases (both for the modeling language and the simulation engine) as well as the semantics of the rule-based modeling language are essential for accurate and consistent modeling and simulation.

8 Discussion

Network-free simulation has now been around for a decade, and its continued use in dynamical systems biology research is strong evidence in favor of the utility of the methodology. Our aim here was to provide a foundation for developing a network-free, kinetic Monte Carlo simulation engine for rule-based models. Beyond the more accessible high-level description of how Gillespie's direct method is generalized for rule-based models, we proceeded to define a basic framework and document a number of nontrivial implementation details required for constructing a network-free simulation engine. While our approach covers a wide range of use cases, its explicit implementation is not general and requires extension to accommodate the edge cases described in Sections 7.1, 7.2 and perhaps others as well.

As the title states, the network-free methodology described here is a generalization of Gillespie's direct method. However, its impact is broader than simply a new approach to capture stochastic fluctuations in biochemical systems with small population sizes. Rule-based modeling, coupled with network-free simulation, enables modelers to define classes of reactions based on limited interaction information (i.e., omitting molecular context that is either irrelevant or not known to alter the rate of reaction for a particular molecular moiety), precluding the need to enumerate all species and reactions for a particular interaction network. Indeed, generating the entire reaction network is often impossible or not computationally feasible. Explicitly accommodating combinatorial complexity enables more detailed and more precise investigation of system dynamics without unjustified simplification of models (Suderman and Deeds, 2013; Deeds et al, 2012; Faeder et al,

2005a). Accommodating such complexity is especially relevant for characterizing systems involving biopolymers (Köhler et al, 2014; Aitken et al, 2013) or systems that can undergo a phase transition to a gel state (Goldstein and Perelson, 1984). Clearly, these studies and others that use network-free simulation engines are not at all concerned with stochastic effects, but with the ability to model and simulate these biochemical systems that contain a high degree of combinatorial complexity (Stites et al, 2015; Creamer et al, 2012). Network-free simulation algorithms provide the only available, general framework for exactly simulating the dynamics of such systems.

Frequent use of existing software suites that have network-free simulation capabilities has led to the discovery of software bugs and inefficiencies as well as the desire for new features. Indeed, the field of network-free simulation algorithm development is not as mature as a that of the SSA field. Ongoing work seeks to improve the performance of network-free simulation (Boutillier et al, 2017a) and there is much work to do. Upon considering the developmental trajectory of the traditional SSA based on Gillespie's direct method, we anticipate that increasing applications of rule-based modeling and network-free simulation will both drive innovation in network-free simulation algorithms and motivate hardening of the software.

As an example of such innovation, some of the complexities discussed in Section 6 can be eliminated at the cost of additional computational overhead with an alternative sampling implementation. Driven by the need to simulate systems that result in a high proportion of null events, species instances themselves could be explicitly tracked throughout the simulation and matches would be associations between patterns and unique sets of molecules as outlined in (Colvin et al, 2010). While this comes at significant computational cost, it is demonstrably useful in models that can produce gels or other large polymers, because it eliminates the need for null events (i.e., a rejection-free algorithm as opposed to the rejection algorithm we describe here) (Yang and Hlavacek, 2011). One useful extension of existing network-free simulation packages might be to implement adaptive algorithm selection that determines on-the-fly whether rejection or rejection-free methods are more appropriate for simulating a particular model's dynamics (e.g., whether or not the mixture contains large polymers). Another straightforward extension of network-free simulation that is becoming increasingly relevant is to consider spatial as well as temporal dynamics (Klann and Koeppl, 2012). As available computing power increases, performing increasingly complex spatial simulations will become feasible. Integrating spatial simulation engines with network-free algorithms are already beginning to emerge (Kochańczyk et al, 2017; Sorokina et al, 2013; Tapia Valenzuela, 2016). Ultimately, we expect that network-free simulation will play an increasingly prominent role in the modeling of complex biological dynamics.

Acknowledgments

We thank Jim Faeder for providing helpful feedback on the manuscript. This work was supported by the National Institute of General Medical Sciences (NIGMS) and the National Cancer Institute (NCI) of the National Institutes of Health (NIH) through grants R01GM111510, P50GM085273, and R01CA197398; by the U.S. Department of Energy (DOE) through contract DE-AC52-06NA25396; and by the Joint Design of Advanced Computing Solutions for Cancer (JDACS4C) program established by DOE and NCI/NIH. Additionally, RS, YTL and SF gratefully

acknowledge support from the Center for Nonlinear Studies (CNLS), which is funded by the Laboratory Directed Research and Development (LDRD) program at Los Alamos National Laboratory.

References

Acar M, Mettetal JT, Van Oudenaarden A. 2008; Stochastic switching as a survival strategy in fluctuating environments. Nat Genet. 40(4):471. [PubMed: 18362885]

Aitken S, Alexander RD, Beggs JD. 2013; A rule-based kinetic model of RNA polymerase II C-terminal domain phosphorylation. J R Soc Interface. 10(86)

Blake WJ, Kærn M, Cantor CR, Collins JJ. 2003; Noise in eukaryotic gene expression. Nature. 422(6932):633. [PubMed: 12687005]

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; 5th International Conference on Systems Biology; 2006. 136–151.

Bortz AB, Kalos MH, Lebowitz JL. 1975; A new algorithm for Monte Carlo simulation of Ising spin systems. J Comput Phys. 17(1):10–18.

Boutillier, P, Ehrhard, T, Krivine, J. Incremental Update for Graph Rewriting. In: Shao, Z, editor. Esop, Lect Notes Comput Sc. Vol. 4807. Springer; Berlin Heidelberg, Berlin, Heidelberg: 2017a. 201–228.

Boutillier P, Feret J, Krivine J, Quyên LK. 2017bKaSim and KaSa Reference Manual.

Bray D. 2003; Molecular Prodigality. Science. 299:1189–1191. [PubMed: 12595679]

Cao Y, Terebus A, Liang JIE. 2016; Accurate chemical master equation solution using multi-finite buffers. Multiscale Model Simul. 14(2):923–963. [PubMed: 27761104]

Chylek LA, Akimov V, Dengjel J, Rigbolt KTG, Hu B, Hlavacek WS, Blagoev B. 2014a; Phosphorylation Site Dynamics of Early T-cell Receptor Signaling. PLoS ONE. 9(8)

Chylek LA, Harris LA, Tung CS, Faeder JR, Lopez CF, Hlavacek WS. 2014b; Rule-based modeling: a computational approach for studying biomolecular site dynamics in cell signaling systems. WIRES Syst Biol Med. 6(1):13–36.

Colvin J, Monine MI, Faeder JR, Hlavacek WS, Hoff DDV, Posner RG. 2009; Simulation of large-scale rule-based models. Bioinformatics. 25(7):910–917. [PubMed: 19213740]

Colvin J, Monine MI, Gutenkunst RN, Hlavacek WS, Hoff DDV, Posner RG. 2010; Rule-Monkey: software for stochastic simulation of rule-based models. BMC Bioinformatics. 11:404. [PubMed: 20673321]

Cox, D, Miller, H. The Theory of Stochastic Processes. Methuen; London: 1965.

Creamer MS, Stites EC, Aziz M, Cahill JA, Tan CW, Berens ME, Han H, Bussey KJ, Von Hoff DD, Hlavacek WS, Posner RG. 2012; Specification, annotation, visualization and simulation of a large rule-based model for ERBB receptor signaling. BMC Syst Biol. 6(1):1. [PubMed: 22222070]

Danos V, Laneve C. 2004; Formal molecular biology. Theoretical Computer Science. 325:69–110.

Danos, V, Feret, J, Fontana, W, Harmer, R, Krivine, J. Rule-based modelling of cellular signalling. In: Caires, L, Vasconcelos, VT, editors. CONCUR 2007 – Concurrency Theory, Lect Notes Comput Sc. Vol. 4703. Springer; Berlin Heidelberg: 2007a. 17–41.

Danos, V, Feret, J, Fontana, W, Krivine, J. Scalable simulation of cellular signaling networks. In: Shao, Z, editor. Programming Languages and Systems. APLAS 2007, Lect Notes Comput Sc. Vol. 4807. Springer; Berlin Heidelberg: 2007b. 139–157.

Danos, V, Feret, J, Fontana, W, Harmer, R, Krivine, J. Rule-based modelling, symmetries, refinements. In: Fisher, J, editor. Formal Methods in Systems Biology. Springer; Berlin Heidelberg: 2008. 103–122. Lect Notes Comput Sc, 5054

Danos, V, Feret, J, Fontana, W, Harmer, R, Krivine, J. Rule-based modelling and model perturbation. In: Priami, C, Back, RJ, Petre, I, editors. Transactions on Computational Systems Biology XI. Springer; Berlin Heidelberg, Berlin, Heidelberg: 2009. 116–137.

de Oliveira, Luís P; HudebineDamienGuillaumeDenisVerstraete, J. Jan.2016 A review of kinetic modeling methodologies for complex processes. Oil Gas Sci Technol. 71(3):45.

Deeds EJ, Krivine J, Feret J, Danos V, Fontana W. 2012; Combinatorial complexity and compositional drift in protein interaction networks. PLoS ONE. 7(3)

Deuflhard, P, Röblitz, S. A Guide to Numerical Modelling in Systems Biology. Springer; Cham: 2015. ODE Models for Systems Biological Networks; 1–32.

Elowitz MB, Siggia ED, Levine AJ, Swain PS. 2002; Stochastic Gene Expression in a Single Cell. Science. 297(August):1183–1187. [PubMed: 12183631]

Endy D, Brent R. 2001; Modelling cellular behaviour. Nature. 409:391–395. [PubMed: 11201753]

Faeder JR, Blinov ML, Goldstein B, Hlavacek WS. 2005a; Combinatorial complexity and dynamical restriction of network flows in signal transduction. Syst Biol. 2(1):5–15.

Faeder JR, Blinov ML, Goldstein B, Hlavacek WS. 2005b; Rule-Based Modeling of Biochemical Networks. Complexity. 10(4):22–41.

Faeder, JR, Blinov, ML, Hlavacek, WS. Rule-based modeling of biochemical systems with bionetgen. In: Maly, IV, editor. Systems Biology. Humana Press; Totowa, NJ: 2009. 113–167.

Faulon JL, Sault AG. 2001; Stochastic Generator of Chemical Structure. 3. Reaction Network Generation. J Chem Inf Comp Sci. 41(4):894–908.

Fermi E, Richtmyer R. 1948Note on census-taking in Monte Carlo calculations. Technical Report.

Gibson MA, Bruck J. 2000; Efficient exact stochastic simulation of chemical systems with many species and many channels. J Phys Chem A. 104(9):1876–1889.

Gillespie DT. 1976; A general method for numerically simulating the stochastic time evolution of coupled chemical reactions. J Comput Phys. 22(4):403–434.

Gillespie DT, et al. 1977; Exact stochastic simulation of coupled chemical reactions. J Phys Chem. 81(25):2340–2361.

Goldstein B, Perelson AS. 1984; Equilibrium theory for the clustering of bivalent cell surface receptors by trivalent ligands. Application to histamine release from basophils. Biophys J. 45(6):1109–1123. [PubMed: 6204698]

Harmer R, Danos V, Feret J, Krivine J, Fontana W. 2010; Intrinsic information carriers in combinatorial dynamical systems. Chaos. 20(3):1–16.

Hufton PG, Lin YT, Galla T, McKane AJ. 2016; Intrinsic noise in systems with switching environments. Phys Rev E. 93(5)

IUPAC. Compendium of Chemical Terminology. 2. Blackwell Scientific Publications; Oxford: 1997 p. (the “Gold Book”)

Kærn M, Elston TC, Blake WJ, Collins JJ. 2005; Stochasticity in gene expression: from theories to phenotypes. Nat Rev Genet. 6(6):451. [PubMed: 15883588]

Kepler TB, Elston TC. 2001; Stochasticity in transcriptional regulation: origins, consequences, and mathematical representations. Biophys J. 81(6):3116–3136. [PubMed: 11720979]

Klann M, Koeppl H. 2012Spatial Simulations in Systems Biology: From Molecules to Cells. Int J Mol Sci. :7798–7827. [PubMed: 22837728]

Kochańczyk M, Hlavacek WS, Lipniacki T. 2017; Spatkin: a simulator for rule-based modeling of biomolecular site dynamics on surfaces. Bioinformatics. 33(22):3667–3669. [PubMed: 29036531]

Köhler, A, Krivine, J, Vidmar, J. A rule-based model of base excision repair. In: Mendes, P, Dada, JO, Smallbone, K, editors. Computational Methods in Systems Biology. CMSB 2014, Lect Notes Comput Sc. Vol. 8859. Springer; Cham: 2014. 173–195.

Kühn C, Hillmann K. 2016; Rule-based modeling of labor market dynamics: an introduction. J Econ Interact Coor. 11(1):57–76.

Le Novere N, Shimizu TS. 2001; STOCHSIM: modelling of stochastic biomolecular processes. Bioinformatics. 17(6):575–576. [PubMed: 11395441]

Le Novere N, Bornstein B, Broicher A, Courtot M, Donizelli M, Dharuri H, Li L, Sauro H, Schilstra M, Shapiro B, et al. 2006; Biomodels database: a free, centralized database of curated, published, quantitative kinetic models of biochemical and cellular systems. Nucleic Acids Res. 34:D689–D691. [PubMed: 16381960]

Lin YT, Buchler NE. 2017Efficient analysis of stochastic gene dynamics in the non-adiabatic regime using piecewise deterministic markov processes.

Lin YT, Doering CR. 2016; Gene expression dynamics with stochastic bursts: Construction and exact results for a coarse-grained model. Physical Review E. 93(2)

Lin YT, Galla T. 2016; Bursting noise in gene expression dynamics: linking microscopic and mesoscopic models. J R Soc Interface. 13(114)

Lin YT, Hufton PG, Lee EJ, Potoyan DA. 2017A stochastic and dynamical view of pluripotency in mouse embryonic stem cells.

Lok L, Brent R. 2005; Automatic generation of cellular reaction networks with Moleculizer 1.0. Nat Biotechnol. 23(1):131–136. [PubMed: 15637632]

Mayer BJ, Blinov ML, Loew LM. 2009; Molecular machines or pleiomorphic ensembles: Signaling complexes revisited. J Biol. 8(9)

Metropolis N, Ulam S. 1949; The Monte Carlo method. J Am Stat Assoc. 44(247):335–341. [PubMed: 18139350]

Metropolis N, Rosenbluth AW, Rosenbluth MN, Teller AH, Teller E. 1953; Equation of state calculations by fast computing machines. J Chem Phys. 21(6):1087–1092.

Munsky B, Khammash M. 2006; The finite state projection algorithm for the solution of the chemical master equation. J Chem Phys. 124(044104)

Munsky B, Trinh B, Khammash M. 2009; Listening to the noise: random fluctuations reveal gene network parameters. Mol Syst Biol. 5(1):318. [PubMed: 19888213]

Nag A, Monine MI, Faeder JR, Goldstein B. 2009; Aggregation of membrane proteins by cytosolic cross-linkers: Theory and simulation of the LAT-Grb2-SOS1 system. Biophys J. 96(7):2604–2623. [PubMed: 19348745]

Nag A, Faeder JR, Goldstein B. 2010; Shaping the response: the role of FcεRI and Syk expression levels in mast cell signalling. IET Syst Biol. 4(6):334–47. [PubMed: 21073233]

Ozbudak EM, Thattai M, Kurtser I, Grossman AD, Van Oudenaarden A. 2002; Regulation of noise in the expression of a single gene. Nat Genet. 31(1):69. [PubMed: 11967532]

Ramaswamy R, Sbalzarini IF. 2010; A partial-propensity variant of the composition-rejection stochastic simulation algorithm for chemical reaction networks. J Chem Phys. 132(4)

Sneddon MW, Faeder JR, Emonet T. 2011; Efficient modeling, simulation and coarse-graining of biological complexity with NFsim. Nat Methods. 8(2):177–183. [PubMed: 21186362]

Sorokina O, Sorokin A, Douglas Armstrong J, Danos V. 2013; A simulator for spatially extended kappa models. Bioinformatics. 29(23):3105–3106. [PubMed: 24021382]

Stites EC, Aziz M, Creamer MS, Von Hoff DD, Posner RG, Hlavacek WS. 2015; Use of mechanistic models to integrate and analyze multiple proteomic datasets. Biophys J. 108(7):1819–1829. [PubMed: 25863072]

Su X, Ditlev JA, Hui E, Xing W, Banjade S, Okrut J, King DS, Taunton J, Rosen MK, Vale RD. 2016; Phase separation of signaling molecules promotes T cell receptor signal transduction. Science. 352(6285):595–599. [PubMed: 27056844]

Suderman R, Deeds EJ. 2013; Machines vs. Ensembles: Effective MAPK Signaling through Heterogeneous Sets of Protein Complexes. PLoS Comput Biol. 9(10)

Suderman, R; Hlavacek, WS. TRuML: A Translator for Rule-Based Modeling Languages. Proceedings of the 8th ACM International Conference on Bioinformatics, Computational Biology, and Health Informatics; New York, NY, USA: ACM; 2017. 372–377. ACM-BCB ‘17

Sweeney B, Zhang T, Schwartz R. 2008; Exploring the parameter space of complex self-assembly through virus capsid models. Biophys J. 94(3):772–783. [PubMed: 17921216]

Tapia Valenzuela, JJ. PhD thesis. University of Pittsburgh; 2016. A study on systems modeling frameworks and their interoperability.

Thattai M, Van Oudenaarden A. 2004; Stochastic gene expression in fluctuating environments. Genetics. 167(1):523–530. [PubMed: 15166174]

Voter, AF. Introduction to the kinetic monte carlo method. In: Sickafus, KE, Kotomin, EA, Uberuaga, BP, editors. Radiation Effects in Solids. Springer Netherlands; Dordrecht: 2007. 1–23.

Yang J, Hlavacek WS. 2011; The efficiency of reactant site sampling in network-free simulation of rule-based models for biochemical systems. Phys Biol. 8(5)

Yang J, Monine MI, Faeder JR, Hlavacek WS. 2008; Kinetic Monte Carlo method for rule-based modeling of biochemical networks. Phys Rev E. 78

Young W, Elcock E. 1966; Monte carlo studies of vacancy migration in binary ordered alloys: I. P Phys Soc. 89(3):735.

Zhang, T; Rohlfs, R; Schwartz, R. Implementation of a discrete event simulator for biological self-assembly systems. Proceedings of the 37th Conference on Winter Simulation, Winter Simulation Conference, WSC ‘05; 2005. 2223–2231.

A PDGFR Activation Model

```
begin model
begin molecule types
  PDGFR(lig,pdgfr,y1 ~0~ P,y2 ~0~ P,y3 ~0~ P,y4 ~0~ P,y5 ~0~P,\
     y6~0~ P,y7 ~0~ P,y8 ~0~ P,y9 ~0~ P,y10 ~0~P)
  Lig(pdgfr)
end molecule types
begin seed species
  PDGFR(lig,pdgfr,y1 ~0,y2~0,y3~0,y4~0,y5~0,y6~0,\
        y7~0 ,y8~0 ,y9 ~0 ,y10 ~0) 1000
  Lig(pdgfr) 10000
end seed species
begin observables
Species PDGFR_dimers PDGFR(pdgfr !1).PDGFR(pdgfr !1)
Molecules phospho_PDGFR  PDGFR(y1~0) PDGFR(y2~0) PDGFR(y3~0)\
                         PDGFR(y4~0) PDGFR(y5~0) PDGFR(y6~0)\
                         PDGFR(y7~0) PDGFR(y8~0) PDGFR(y9~0)\
                         PDGFR(y10 ~0)
end observables
begin reaction rules
  PDGFR(lig) + Lig(pdgfr) <-> PDGFR(lig !1).Lig(pdgfr !1) 1, 1
  PDGFR(lig!+,pdgfr) + PDGFR(pdgfr) <-> \
    PDGFR(lig!+,pdgfr !1).PDGFR(pdgfr !1) 1, 1
  PDGFR(pdgfr!+,y1 ~0) -> PDGFR(pdgfr!+,y1~P) 1
  PDGFR(pdgfr!+,y2 ~0) -> PDGFR(pdgfr!+,y2~P) 1
  PDGFR(pdgfr!+,y3 ~0) -> PDGFR(pdgfr!+,y3~P) 1
  PDGFR(pdgfr!+,y4 ~0) -> PDGFR(pdgfr!+,y4~P) 1
  PDGFR(pdgfr!+,y5 ~0) -> PDGFR(pdgfr!+,y5~P) 1
  PDGFR(pdgfr!+,y6 ~0) -> PDGFR(pdgfr!+,y6~P) 1
  PDGFR(pdgfr!+,y7 ~0) -> PDGFR(pdgfr!+,y7~P) 1
  PDGFR(pdgfr!+,y8 ~0) -> PDGFR(pdgfr!+,y8~P) 1
  PDGFR(pdgfr!+,y9 ~0) -> PDGFR(pdgfr!+,y9~P) 1
  PDGFR(pdgfr!+,y10 ~0) -> PDGFR(pdgfr!+,y10~P) 1
  PDGFR(y1~P) -> PDGFR(y1~0) 1
  PDGFR(y2~P) -> PDGFR(y2~0) 1
  PDGFR(y3~P) -> PDGFR(y3~0) 1
```

```
  PDGFR(y4~P) -> PDGFR(y4~0) 1
  PDGFR(y5~P) -> PDGFR(y5~0) 1
  PDGFR(y6~P) -> PDGFR(y6~0) 1
  PDGFR(y7~P) -> PDGFR(y7~0) 1
  PDGFR(y8~P) -> PDGFR(y8~0) 1
  PDGFR(y9~P) -> PDGFR(y9~0) 1
  PDGFR(y10~P) -> PDGFR(y10 ~0) 1
end reaction rules
end model
begin actions
simulate ({ method=>“nf”,t_start=>0,t_end=>100,n_steps=>100})
end actions
```

Code 1

A model for platelet-derived growth factor receptor (PDGFR) activation written in the BioNetGen language. PDGFR can dimerize if at least one PDGFR is bound to ligand. Dimerized PDGFR can then undergo autophosphorylation of 10 distinct tyrosine residues, and phosphorylation (and dephosphorylation) on each residue occurs independently. Here, all rate constants are set to 1, and the ‘\’ character denotes line continuation for clarity. Actual simulation of the model with NFsim (see the ‘actions’ block) requires consolidating the continued lines in the ‘molecule types’ and ‘seed species’ blocks.

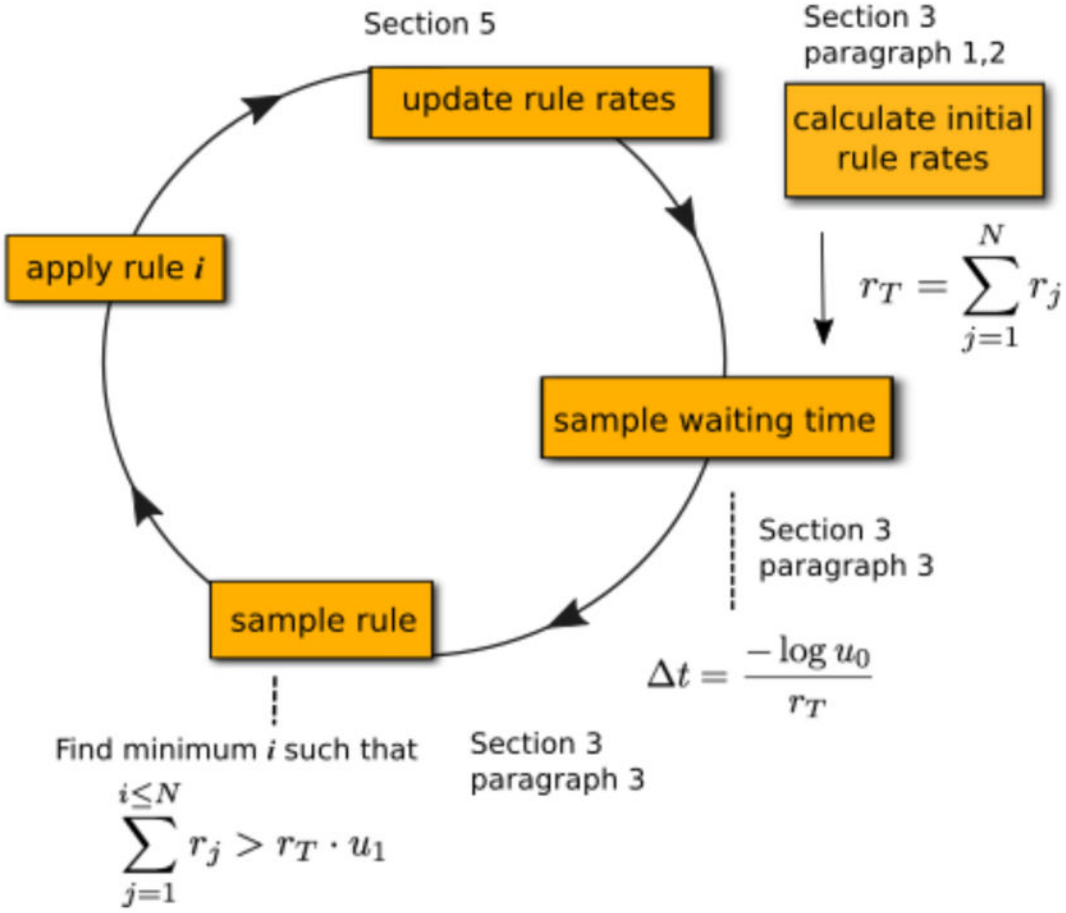

Fig. 1.

A simple graphical depiction of a generalization of Gillespie's direct method, where N is the total number of rules. Sections describing the relevant steps are labeled in the figure. After initially computing the rates for every rule, each iteration of the simulation loop requires generating two pseudorandom numbers u_0 and u_1 on the interval [0,1). The waiting time until the next event, Δt, is calculated from the first random number and it is inversely proportional to the total rate of the rules in the system: r_T. Then rule i is sampled, where i is the smallest number such that the cumulative rate of rules 0 through i is greater than the product of the second random number and the total rate, r_T. After both the next event time and rule (i.e., the type of reaction) are sampled, the rule is applied to a specific set of molecules from among all molecules that qualify as reactants, and the rates are updated based on the change of the system.

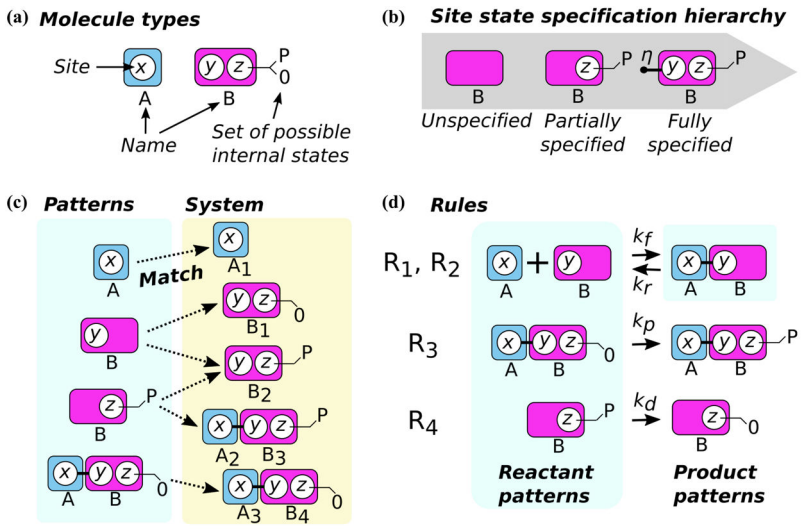

Fig. 2.

Nomenclature for rule-based modeling. **a** Molecule types are archetypes for individual objects (molecules) in a simulation. Each molecule type has a name and a list of sites. Sites contain information about binding state, and may occupy a state from a set of possible internal states. **b** Molecules can be ordered based on their degree of specification. Note that such an ordering assumes that information present in less specific molecules is preserved in more specific molecules. A molecule may have no specified sites. A partially specified molecule includes some information about binding site state (site z is phosphorylated). For full specification, the state of every site is explicit (site y is bound with edge η and site z is phosphorylated. **c** A match exists when a pattern's sites are either equivalent to or less specific and consistent with the site states in corresponding molecules in a simulation mixture (the subscript on the molecules are unique integer identifiers for molecules of a particular type). **d** Four rules are defined (R_1 to R_4): association/dissociation of molecules A and B, phosphorylation of B when bound to A, and dephosphorylation of B. A rule consists of reactant patterns, a transformation defined by product patterns, a rate law, and (optionally) any other conditions that influence the rule. Typically reactants are written on the left-hand side of the rule and products are written on the right hand side. However, note that since the first rule is reversible (i.e., it has the double-arrow operator), the right-hand side serves as the reactant pattern for the dissociation reaction and the left-hand side contains the product patterns.

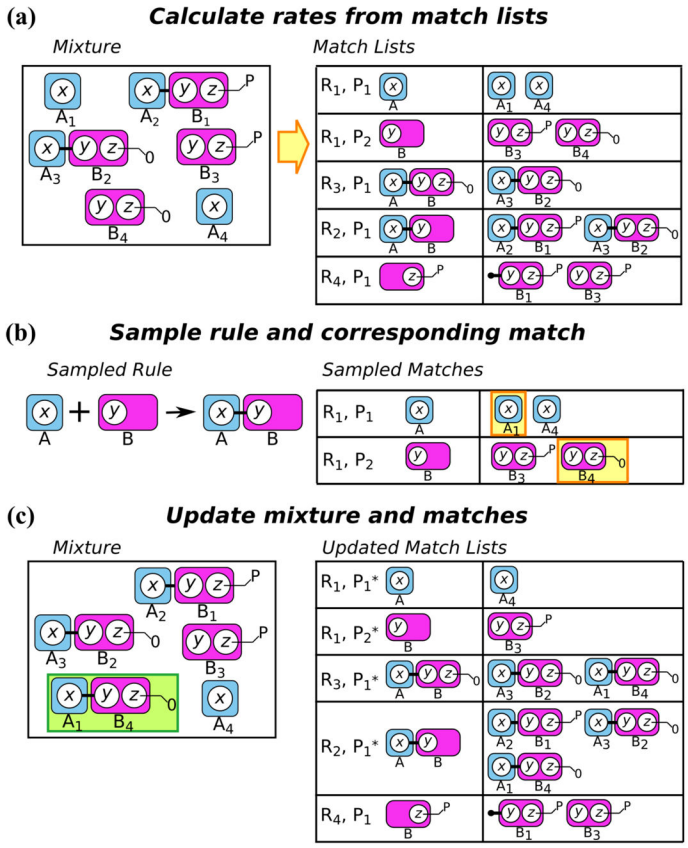

Fig. 3.

Schematic of a network-free simulation algorithm. **a** At any point during a simulation, the mixture is stored in memory, as is the match list for each pattern. Note that R_N, P_M refers to the N^{th} rule and M^{th} pattern in that rule. The rules considered here are those presented in Fig. 2. Rates for each rule can be calculated based on the rule's rate law and its match lists. **b** At each simulation step, a rule is chosen with a probability given by its rate relative to the overall rate of all rules. A match is then chosen from the match list of each reactant pattern in the rule. **c** The simulation mixture is updated according to the transformation defined in the rule. The species highlighted in green is the result of the transformation in panel b. The match lists are then updated; for each modified molecule, matches that have become invalid are removed, and new matches are added to the appropriate match lists. Match lists with an asterisk have been modified from the initial system state in panel a.

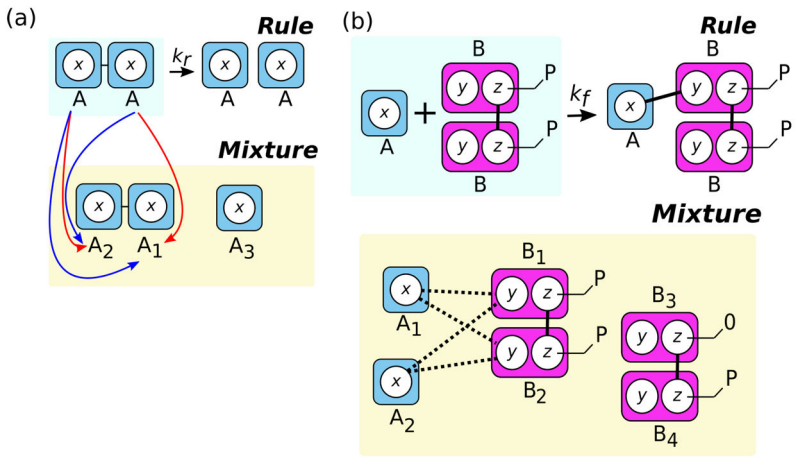

Fig. 4.

Symmetry and reaction path degeneracy influence rule rate calcuation. **a** Symmetry in a pattern results in multiple matches (red and blue arrows) from the pattern to a particular set of molecules. Correctly computing the rate of a rule (given certain assumptions outlined in Section 6.2) requires that the rate is divided by the number of symmetries present in the rule's patterns. **b** Rules where a pattern has multiple sites or pattern molecules that can be the target of the transformation defined by a rule have reaction path degeneracy. This can be seen in the simulation mixture illustrated at the bottom of the panel, where there are four possible binding events (denoted by the broken lines) implied by the rule illustrated at the top of the panel. Correctly calculating the rate for the rule requires multiplying the match-based rate by the number of possible reaction paths present among the rule's patterns (2 *y* sites competent for binding site *x* on *A*) and also accommodating other symmetries effects if necessary (dividing by 2 due to the symmetric *B*–*B* dimer pattern).

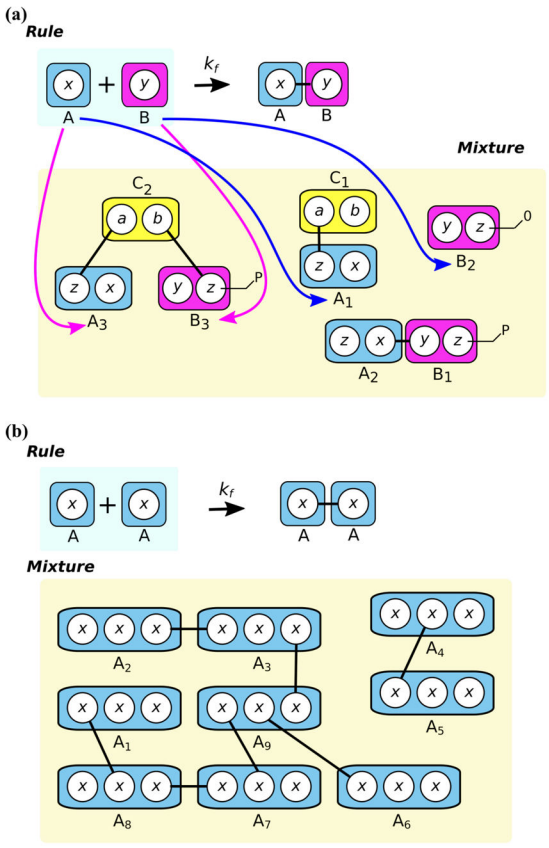

Fig. 5.

Null events arising from molecularity constraints. **a** Selecting the match corresponding to the magenta arrows would fail to satisfy molecularity, resulting in a null event, whereas selecting the match corresponding to the blue arrows does satisfy the molecularity constraint of the rule. **b** Example rule and mixture in which over half of all potential reaction events would be null events, making for a very inefficient simulation.

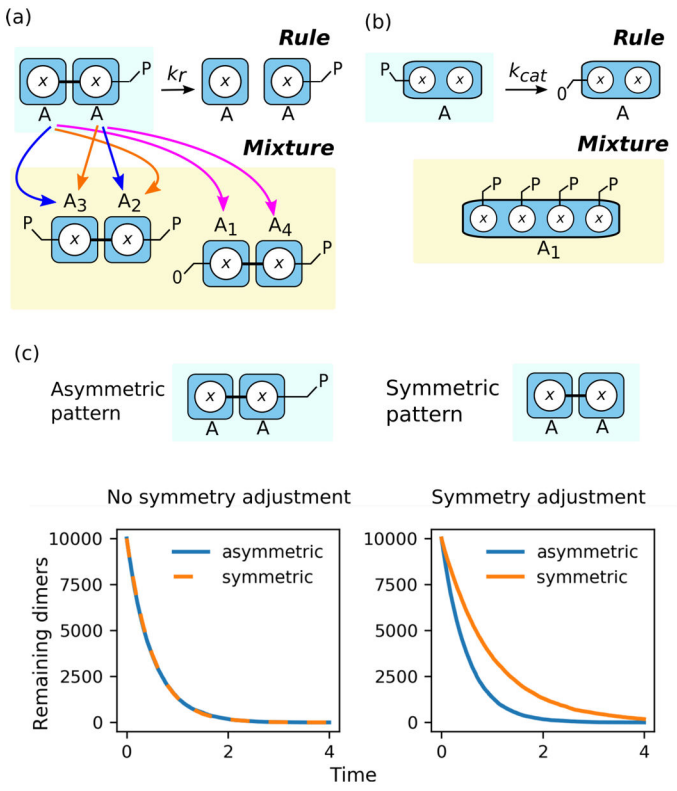

Fig. 6.

Certain pathological behaviors may arise due to language conventions. **a** Dimer dissociation example, where the pattern matches one potential reactant twice and another potential reactant once. If dissociation rate constants refer to the rate of a single bond breaking, then inconsistencies in rate calculation may arise when asymmetric patterns match symmetric molecules. **b** Dephosphorylation in the presence of multiple sites. The rate of this rule depends on interpretation. If each site *x* should be dephosphorylated independently with rate k_{cat} and the only other constraint is at least one unbound site *x*, then a hierarchical matching scheme needs to be implemented (Section 7.2). **c** Curves for dimer dissociation (as described in panel a) resulting from asymmetric patterns and symmetric patterns applied to the same system of symmetric species (*A*–*A* homodimers with all sites in the 'P' state). When the simulator automatically adjusts for pattern symmetry (Section 6.2), there is a discrepancy in the rate calculation that is observed in distinct dissociation curves (right), compared to a paradigm in which the rate constants are applied to the number of matches directly (left).