universität wien

MASTERARBEIT | MASTER'S THESIS

Titel | Title

Stochastic simulator of a chemical-flow reactor specified via rewrite rules

verfasst von | submitted by

Gerd Gerber BSc

angestrebter akademischer Grad | in partial fulfilment of the requirements for the degree of

Master of Science (MSc)

Wien | Vienna, 2024

Studienkennzahl lt. Studienblatt | Degree programme code as it appears on the student record sheet:	UA 066 862
Studienrichtung lt. Studienblatt | Degree programme as it appears on the student record sheet:	Masterstudium Chemie
Betreut von | Supervisor:	ao. Univ.-Prof. Mag. Dr. Christoph Flamm

Acknowledgements

I extend my deepest gratitude to Prof. Christoph Flamm for his invaluable guidance and expertise throughout my Master's thesis in the Theoretical Biochemistry Group. His mentorship has been instrumental in shaping my research and academic growth.

I also thank all members of the Theoretical Biochemistry Group for their supportive and intellectually stimulating environment, which greatly contributed to my project.

Special thanks to my family and friends for their unwavering support and encouragement.

Abstract

Stochastic chemical kinetics serves to analyze the time progression of a homogenously mixed chemical reaction system, regarding the number of molecules and their random dynamical behaviour. This analytical perspective is progressively gaining traction in the exploration of biological cellular systems. In the sphere of computational simulations for biochemical reactions, Gillespie's direct method stands as a cornerstone for the stochastic simulation of chemical kinetics. Nonetheless, the algorithm's necessity for the listing of every potential reaction and chemical species within a system can be a significant limitation. This is particularly evident in situations where a combinatorial explosion of reactions and species occurs within biological networks, which makes an explicit enumeration unfeasible. To address this, rule-based modeling frameworks were introduced as a solution capable of accurately depicting networks with extensive combinatorial complexity. These frameworks have been crucial in formulating extensions of Gillespie's direct method, fostering the creation of simulation engines designed for rule-based modeling languages. Additionally, atom-tracking is introduced, so that reaction trajectories can be constructed and analyzed for network simulations.
In conclusion, potential pathways for further development in the domain of network-free simulation are examined, with emphasize on its ongoing significance in building models and finding reaction motifs for dynamic systems in biology.

Kurzfassung

Stochachstische chemische Kinetik dient zur Analyse des zeitlichen Verlaufs eines homogenen gemischten chemischen Reaktionssystems, bezogen auf die Anzahl der Moleküle und ihr zufälliges dynamisches Verhalten. Diese analytische Perspektive gewinnt zunehmend an Bedeutung in der Erforschung biochemischer Systeme. Im Bereich der computergestützten Simulationen für biochemische Reaktionen steht Gillespies Direct Method als Eckpfeiler für die stochastische Simulation chemischer Kinetik. Dennoch kann die Notwendigkeit des Algorithmus, jede potenzielle Reaktion und chemische Spezies innerhalb seines Systems aufzulisten, eine erhebliche Einschränkung sein. Dies wird besonders in Situationen deutlich, in denen eine kombinatorische Explosion von Reaktionen und Spezies in biologischen Netzwerken auftritt, was eine explizite Aufzählung unpraktikabel macht. Um dies zu adressieren, werden regelbasierte Reaktionsnetzwerke als Lösung eingeführt, die in der Lage sind, Systeme mit umfangreichen kombinatorischen Komplexität genau darzustellen. Diese Netzwerke sind entscheidend bei der Formulierung von Erweiterung von Gillespies Direct Method, was die Schaffung von Simulatoren für regelbasierte Modellierungssprachen fördert. Zusätzlich wird das Atom-Tracking eingeführt, sodass Reaktionstrajektorien für Netzwerksimulationen konstruiert und analysiert werden. Abschließend werden potenzielle Wege für die weitere Entwicklung im Bereich der netzwerkfreien Simulation untersucht, mit Betonung auf ihrer anhaltenden Bedeutung beim Aufbau von Modellen und dem Auffinden von Reaktionsmotiven für dynamische Systeme in der Biologie.

Contents

1 Introduction

In a living cell, numerous biochemical units enter into complex interactions that form extensive reaction networks. The functionality of a cell is significantly influenced by the dynamic behaviour of these networks. A primary goal of systems biology is to understand the expression of various phenotypes that result from the complex interactions within these reaction networks. To decipher these complex networks, mathematical modelling and simulation are powerful tools. These methods enable precise definition and simulation of the interactions between chemical species, using a range of techniques to represent nonlinear dynamical systems. Atom tracking is emerging as a key component in this context, providing a deeper understanding of the different pathways and motifs, thus improving the analytical depth and predictive power of these models. By introducing atom tracking, researchers can reveal finer details of reaction dynamics, which is important for understanding the underlying mechanisms that govern cellular functions. By creating an environment for exploratory analysis and hypothesis testing, they increase the efficiency and effectiveness of research efforts. As a result, computational systems biology has experienced significant growth in recent years and has established itself as a prominent discipline in the field of quantitative biology, contributing significantly to a more nuanced and comprehensive understanding of cellular processes.

1.1 Chemical Reaction Networks

Traditionally, reaction networks have been represented by ordinary differential equations (ODEs) in combination with numerical integration algorithms based mainly on mass action kinetics. In a well-mixed and thermally equilibrated chemical environment, the number of molecules Xi for each chemical species Si (where $i = 1, ..., N$) changes over time as described by a series of interconnected ordinary differential equations (ODEs) as the reaction rate equation (RRE):

$$\frac{dX_i}{dt} = f_i(X_1, ..., X_N) \qquad i = (1, ..., N) \tag{1.1}$$

The interaction of the molecules is described by chemical reactions Rj (where $j = 1, ..., M$). The system is assumed to be in constant volume Ω and in thermal, but not chemical, equilibrium with constant temperature. The number of molecules of species Si in the system at time t is described by $\mathbf{X}(t) = (X_1(t), ..., X_N(t))$, assuming that the system was in the state $\mathbf{X}(t_0) = \mathbf{x}_0$ at the starting time t_0. The assumption that we can define the state of the system by looking only at the populations of molecules, without considering the positions and velocities of the individual molecules, is based on a thorough mixing of

the system. The primary hypothesis is that most molecular collisions in the system are elastic and non-reactive. These collisions cause the molecules to be uniformly distributed and their velocities to conform to the Maxwell-Boltzmann distribution (Gillespie, 2006 [1]). Therefore, we can disregard the non-reactive collisions that would otherwise consume much of the computational time in a molecular dynamics simulation and focus only on events that change the population of chemical species. The reaction channels are thus characterised by two mathematical quantities:

1) state-change vector: $\mathbf{v}_j = (v_{1j}, ..., v_{Nj})$:

The molecular population S_i undergoes a change caused by a reaction v_{ij}. When the system is in the state $\mathbf{x}$ and a reaction R_j takes place, the system enters the state $x + v_j$.

2) propensity function a_j:

Since $\mathbf{X}(t) = \mathbf{x}$, $a_j(\mathbf{x})dt$ represents the probability that a reaction R_j will occur within the small time interval $[t, t + dt)$. The second definition can be regarded as the fundamental postulate of stochastic chemical kinetics, since the rest of the theory is derived from it on the basis of the laws of probability.

While this method has proven successful in many applications (Deuflhard et al., 2015 [2]), it is not universally suitable for cells or comparable systems. This is because ODEs describe concentrations of chemical species under the assumption that they are well mixed, which may not be the case in cellular environments where species are present in limited amounts. In particular, eukaryotic cells have tiny volumes, such as picolitres, that contain limited populations of chemical species. This limitation, due to the discrete nature of these species, introduces noise inherent in the system that can interfere with the study of intracellular reaction dynamics (Elowitz et al., 2002 [3]).
To address these challenges, stochastic models, in particular continuous-time Markov chains (CTMCs), have become the preferred method to capture the unpredictable nature of biochemical reactions with finite and discrete reactants. The joint probability distribution of CTMCs is determined by the chemical master equation (CME). Solving the CME directly can be daunting and requires innovative numerical methods. There are two prominent approaches: one integrates or approximates the CME directly, the other uses continuous-time Monte Carlo methods to generate sample paths from which statistical metrics can be derived (Gillespie et al. 1977 [4]). The focus of this thesis is on Monte Carlo methods.
The concept of using Monte Carlo approaches to simulate trajectories of dynamical systems can be traced back to the 1940s, initiated by researchers such as Fermi and Richtmyer (1948) and Metropolis and Ulam (1949) at Los Alamos. Their focus was on kinetic theory, in particular the Boltzmann and Fokker-Planck equations. The later Metropolis algorithm shifted the focus from time-dependent dynamics to equilibrium distributions in thermal systems. In the 1960s, Monte Carlo sampling was extended to continuous-time dynamics and found applications in fields ranging from materials science to statistical

physics (Voter 2007). A milestone in the development was the introduction of the n-fold BKL algorithm by Bortz et al. (1975), a method without rejection. Gillespie (1976) soon adopted this procedure for chemical reaction networks, leading to the widely recognised Gillespie algorithm or stochastic simulation algorithm (SSA). Later developments focused on refining the SSA, with innovations such as indexed priority queues (Gibson and Bruck, 2000 [8]) or the use of network structures such as sparse transition matrices (Ramaswamy and Sbalzarini 2010, [9]), but the core of the algorithm remained the same.
The Stochastic Simulation Algorithm (SSA) works under the assumption that the complete chemical reaction network is well-defined. However, in the context of biology, we often deal with data limited to direct pairwise associations. A layer of complexity is added when considering the multiple reactions required to synthesize a specific product molecule in biological systems. A challenge arises from the number of potential reaction steps leading to the creation of a desired product. For instance, the synthesis of a particular biochemical compound might necessitate a cascade of sequential reactions. Each step in this sequence can have variations or alternate pathways, exponentially increasing the number of possible routes to achieve the end product. Representing these pathways in a differential equation system could result in a vast number of equations, accounting for each possible reaction route. This accumulation of potential reactions is a problem regarding combinatorial complexity in biological systems.

1.2 Simple Representation Of Complex Systems

Rule-based modelling has emerged in the field of systems biology as a solution to the challenge of combinatorial complexity, particularly when it comes to models related to intracellular signalling networks. The main advantage of this approach is that rules are used to represent a set of reactions that share common reaction centres. These rules identify recurring patterns between numerous reactive molecules and streamline the representation of reactions without the need to list all potential reactions between all biochemical entities (Chylek et al., 2014 [10]). This rule-based modelling method allows for a concise but detailed representation of complex systems. With this approach, the above-mentioned systems can be simplified to only a small number of rules, as the rules function independently of each other (example details).
Originally, the goal of rule-based modelling in systems biology was to facilitate the automatic creation of biochemical reaction networks. The use of rules to describe biological events minimises the modelling effort considerably (Blinov et al., 2006 [11]). However, the size of the resulting network can become computationally overwhelming or have no limit except the number of molecules in the system. An advanced solution to overcome this obstacle is real-time network generation. Even if the number of species is huge or possibly infinite, real-time network generation assumes that a fixed (and often limited) number of species is prevalent. So instead of determining the reaction rates of all possible reactions, which could be impractical, the reaction rates of the local network are calculated as a function of the reactants present. As the state of the system changes, this local network is dynamically updated. However, this method can be inefficient if the grid

boundary grows exponentially.
To speed it up, an alternative strategy does not create the chemical reaction grid at all. Here, each molecule is represented as a single entity, and the simulations act directly on those entities. This network-free method is based on fundamental physical and chemical theories (Yang et al., 2008 [12]; Sneddon et al., 2011 [13]; Zhang et al., 2005 [14]). Over time, several software tools have been developed using this strategy, each targeting a specific rule-based modelling paradigm. The algorithms of these tools adapt Gillespie's well-recognised direct method, but require careful monitoring of the state of the system (Suderman et al., 2018 [15]).

1.3 Application Of Network-Free Simulations

Several software tools have been developed to support network-free simulation ,which are specific to certain rule-based modelling syntaxes. One primary objective of network-free simulation is to comprehend the influence of combinatorial complexity on cellular reactions. Studying this complexity doesn't allow for traditional species or reaction network listings, making network-free simulation indispensable.
In mammals, intracellular signaling frequently engages in the formation of multimeric structures involving cell receptors (Su et al., 2016 [16]). Given their complexity, only network-free simulation methods can evaluate these systems without oversimplifying. Many studies on mammalian signaling utilize BNGL models and rely on the NFsim simulator for network-free analysis.
Furthermore, biopolymer studies, especially on nucleic acids, benefit from rule-based modeling. For instance, research on RNA polymerase II's binding to DNA positions considers the combinatorial complexity challenge as DNA size grows (Aitken et al., 2013 [17]). Another study delved into DNA repair, focusing on a DNA molecule with a massive number of base pairs, offering insights unattainable through other modeling methods (Köhler et al., 2014 [18]).
Beyond biology, rule-based modeling finds relevance in other domains, like labor market simulation (Kühn and Hillmann, 2016 [19]). These models underline the unique features of rule-based modeling compared to general agent-based models, especially the absence of spatial considerations in network-free simulations. While the versatility of rule-based modeling is evident, our emphasis remains on its biological applications, especially with a keen interest in network-free simulation methodologies.

1.4 Description Of Network-Free Simulations

The algorithm for generating precise pattern paths in a rule-based model using a network-free simulation shown in Fig. 1 is mostly similar to Gillespie's direct method (Gillespie, 1976 [7]; Gillespie et al., 1977 [4]). In this section we give a general overview of the algorithm, reserving the specific technical refinements associated with network-free simulations for subsequent sections. We establish some terminology, which will be explained in more detail.

1. molecule: distinct entity monitored by the simulation
2. rule: consists of patterns pinpointing specific segments within reactant molecules
3. pattern: recognize and align with reactant molecules during the simulation.
4. chemical transformation: executed on a reactant molecule, defined by a rule
5. species: denotes a category of a certain molecular compound
6. mixture: contains every molecule interacting within the simulation

For the description of the algorithm, it is classified into four steps: initialization of the system, computation of the rules' rates, advancing time and sampling a rule and updating the system.

- *Step 1*: At the beginning of the simulation, the user determines the configuration of the system. Specific instances of component molecules are defined for each chemical species.

- *Step 2*: During this phase, correlations are drawn between the patterns outlined in the rules and the specific molecular instances within the simulation mixture. A key distinction between rule-based models and conventional models is that chemical changes are described using patterns rather than specific chemical species, as indicated by Gillespie (1976) and Gillespie et al. (1977). Consequently, various unique chemical species might act as reactants in a rule, as long as all these species align with the rule's designated reactant pattern. The rate at which the rule operates is proportional to the count of its reactant pattern matches.

- *Step 3*: After calculating the rates for all the rules, the time until the next rule is applied is determined by a random number drawn from an exponential distribution with a rate parameter equal to the sum of the rates of all the rules r_T. A random waiting time Δt is derived from this distribution, where u_0 is a random number in the range [0,1):

$$\Delta t = \frac{\text{-log}\, u_0}{r_T} \tag{1.2}$$

  Subsequently, the simulation time for the system is advanced by Δt. The rule to implement in the system is chosen based on a probability that aligns with its rate through the following conditional procedure:

  1. Generate a uniform random number u_1 within the range [0,1).
  2. Iterate over all rules to identify the smallest rule index, i, corresponding to a particular rule, so

$$\sum_{j=1}^{i \leq N} r_j > r_T \cdot u_1 \tag{1.3}$$

This approach is an adaptation of Gillespie's direct method as presented in Gillespie 1976 and Gillespie et al. 1977. However, since rules are pattern-based, the chosen rule doesn't directly indicate which molecular instances are subject to the transformation defined by the rule. To address this, the simulator selects molecules from a match list for each of the picked rule's reactant patterns, using a uniform probability distribution.

- *Step 4*: Once the rule has been executed (altering the molecules identified in the previous step), the correlations between the rule patterns and the molecules in the simulation are refreshed. Once these rates are updated, the simulation proceeds from Step 3 until a predetermined endpoint is reached. A comprehensive explanation of this phase's execution is provided in Chapter 3. While the overview provided closely resembles Gillespie's foundational direct method, integrating this algorithm with a specific rule-based modeling system demands accurate evaluation of how pattern recognition impacts a rule's rate of application. Potential challenges and considerations in designing a network-free algorithm are explored in further sections.

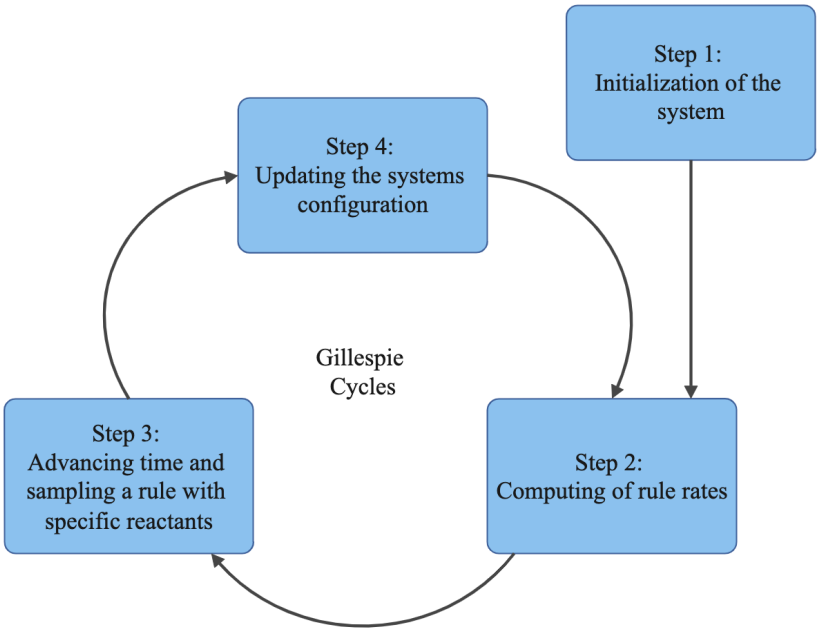

Figure 1.1: Schematic representation of the Gillespie Algorithm [15]

1.5 Implementation Of Network-Free Simulations

In the subsequent sections, the technical nuances essential for the execution of a network-free simulation algorithm are described. Initially, it's crucial to explain the entities participating in the simulation. Various rule-based modeling frameworks might use unique terminologies for analogous or even identical concepts. In rule-based modeling, the majority of objects can be visually (and formally) depicted as graphs. Consequently, rules act as transformations or modifications of these graphs, and a significant part of the terminology in rule-based modeling derives from graph theory. While these terms may be referenced from time to time, they won't be central the discussion. The goal is to present clear and accurate definitions of the components essential for building a network-free simulation algorithm, free from constraints of any specific rule-based modeling framework. It should be noted that the terminology aligns with the practices in molecular and cellular biology (Chylek et al.,2014 [10]). In the realm of rule-based modeling for biochemical reactions, the fundamental unit is termed the *molecule* (Fig. 1.2). This molecule usually represents a metabolite, drug, another small molecule, or any focal element in the model. Many rule-based modeling frameworks postulate that molecules have a specific signature. All instances of these molecules, including those monitored during simulation, must adhere to this definition. Molecules representing particular molecular moieties, rather than distinct physical entities, are called *pattern molecules* (see Fig. 1.2). Sites within these pattern molecules can be entirely unspecified, partially or fully defined to embody the essential characteristics needed for a representation of the molecular segment. The idea of an order in pattern specificity plays a significant role in pattern matching: an object with less specificity can correspond to an object with more specificity or full definition, as long as the data in the less detailed object is consistent with the more defined one. This underlying principle of information is assumed in subsequent discussions about molecules' partial specifications during pattern matching. If the internal state of one site mirrors another, those sites are considered equivalent. Groups of sites (and by extension, molecules) can be organized based on their specificity or equivalency. Complete molecules, often just referred to as molecules, are defined as those where every subpattern is fully specified. This detailed specification means that subpattern are either connected to another subpattern through a labeled bond or remain unbound. Such complete molecules actively participate in the simulation and undergo changes due to rule applications.

A molecular pattern in the context of a molecule can be thought of as a specification gradient ranging from unspecified to fully specified. At one end of this gradient, an unspecified pattern serves as a broad template, capturing a wide array of molecular structures without detailed constraints. As we move along the gradient toward a fully specified pattern, the constraints become more rigorous, providing a precise and unique representation of a molecule's substructure. The more specified a pattern becomes, the smaller its match scope. Essentially, this is similar to searching for a specific substructure or motif within a molecule, ensuring that each atomic position, bond type, and associated properties align precisely with the defined pattern. Objects composed of complete molecules and pattern molecules can be defined in hierarchical terms. When a list consists

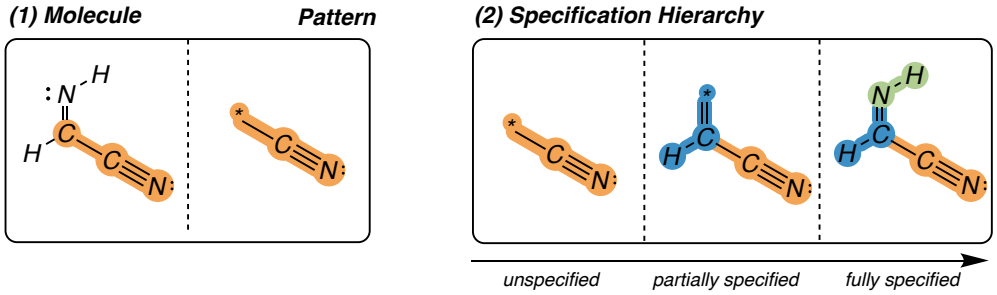

Figure 1.2: Nomenclature for rule-based simulations
(1) Molecule types serve as prototypes for individual entities (molecules) within a simulation. Each molecule type is characterized by a name and a set of sites, with each site containing details regarding its binding state. (2) Molecules can be classified according to their level of specification. It is important to note that this classification assumes that less specific molecules contain inherently the information that is present in more specific molecules. A molecule can fall into several categories: unspecified, partially specified and fully specified

of one or more complete molecules interconnected exclusively within that list (similar to a 'connected component' in graph theory), it's termed a "species", aligning with traditional chemical terminology. It's worth noting that a standalone molecule without any binding sites can be categorized as both a molecule and a species. Similarly, a "pattern" is described as a set of pattern molecules that are exclusively linked within that set (as depicted in Fig. 1.4). To describe the collective pool of complete molecules engaged during the simulation of a rule-based model, terms like "mixture" or "simulation mixture" are introduced (see in Fig. 1.4). Network-free simulation systems monitor individual entities within a mixture, contrasting with Gillespie's direct method that monitors populations of predetermined chemical species. It's worth noting that chemical species populations can be derived from the list of fully specified molecules that characterize a mixture. Thus, our method offers a representation of the system's state equivalent to the conventional SSA when observing population dynamics.
Rates are determined through pattern matching so the prior creation of the reaction network is avoided. A match is characterised as a link between a pattern and a set of molecules within a simulation mixture (Fig. 1.4). This linkage is based on two key factors:

1. Every pattern molecule within the pattern should align with a molecule of identical name in the mixture

2. For a given pattern molecule P in the pattern and its matching molecule M in the mixture:

   a) Should P's sites be less defined than those of M, the details in P's sites need to be retained in M's sites (maintaining consistency).

   b) Sites in P, which are completely defined must find equivalent matches.

A rule defines a series of reactions, incorporating all previously outlined concepts (see Fig. 1.4). It consists of a sequence of reactant patterns paired with product

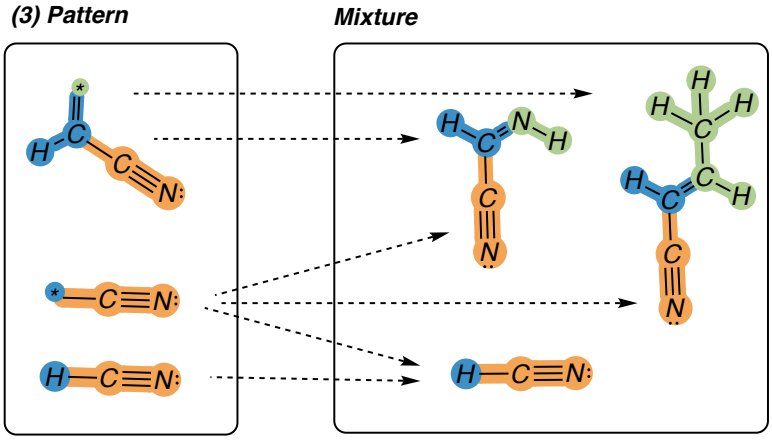

Figure 1.3: Nomenclature for rule-based simulations
(3) A match (arrow) is found when the sites of a pattern are either identical or less specified and are consistent with the corresponding molecules within a simulation mixture

patterns, delineating a transformation (refer to Fig. 1.4), and is governed by a rate constant. Typically, the reactant and product patterns are referred to as the left-hand side (reactants) and right-hand side (products) of a rule, respectively. The rate constant determines the rate of the rule, which is the sum of rates of all the reactions specified by that rule. Each specific reaction implied by a rule adheres to the rate constant associated with that rule. The term "system" in contains all elements necessary to accurately represent and simulate the model.

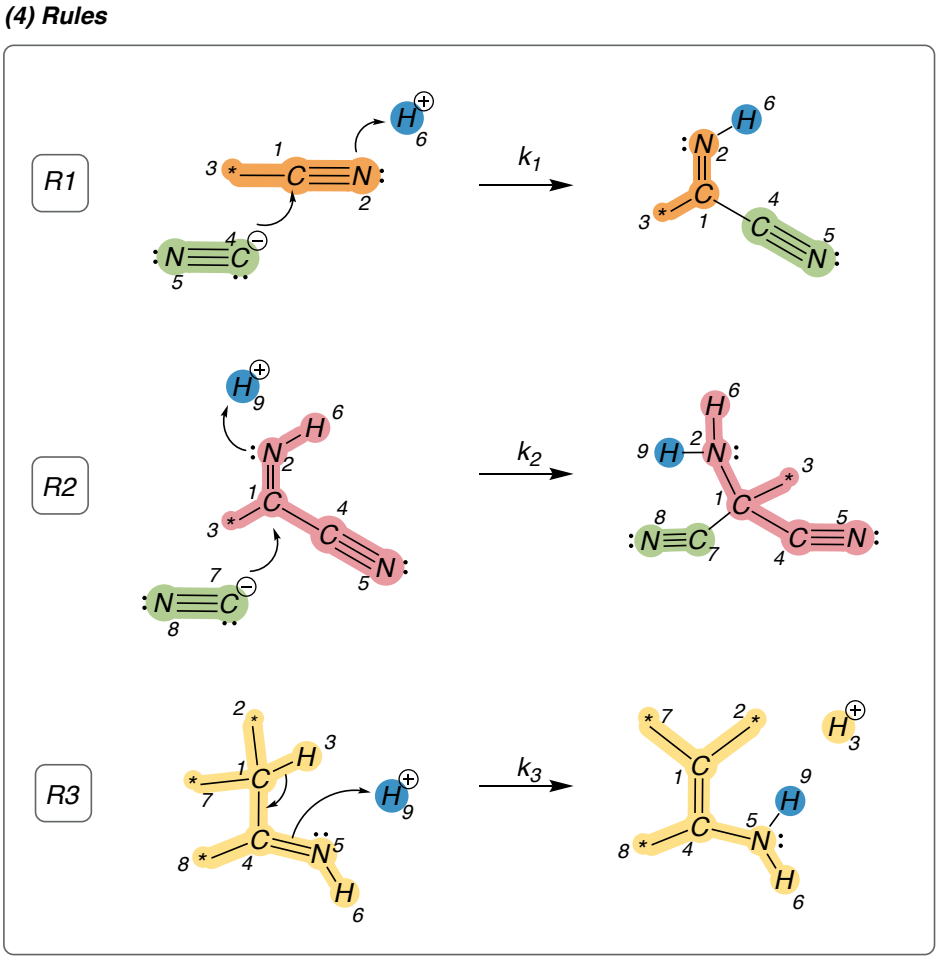

Figure 1.4: Nomenclature for rule-based simulations
(4) Three rules (pattern-based reactions) are defined. Each with different specification hierarchies. Every reaction consists of reactant pattern, product pattern (transformation) and a rate constant

2 Pattern-based Stochastic Simulation Algorithm

The overview given in Chapter 1.5 departs from a traditional SSA in several aspects. Instead of relying on species population sizes, rates are determined based on the number of matches and therefore number of possible reactions. Another significant difference is the requirement to select molecules for modification after choosing a rule for application. Arguably the most pronounced and complex deviation lies in the approach adopted to maintain and modify the system's state. This section provides a comprehensive overview of the data structures necessary to account for these variations. It is not claimed that this approach is the most efficient, but it helps to illustrate the basic procedures for developing a network-free simulation algorithm. It is also important to add, that this approach is not only pattern-based but also implements atom tracking into the simulation, which will be described in more detail in Chapter 3.

2.1 Rule Rates Calculation

Determining the initial rates of the model's rules, the number of matches to each reaction pattern in a rule must first be calculated. Therefore a pointer list is created for each molecule species. This pointer list contains information about the count of every species in the mixture and is used to create the match list in the next step. All molecules in the original mixture that match these pointer molecules are identified as molecules from the same species (see Fig 2.1). To create the match list for each reaction pattern, every pointer molecule matching the reaction pattern is added to the match list and the number of matches is multiplied with the number of the corresponding species. A simple way to store this data is to keep the match list for each reaction pattern within each rule. This list would include all matches of a pattern with the molecules present in the simulation mixture (see Fig. 2.2). The collection of matches results in the initial match list for the pattern. For rules following elementary rate laws (rate laws aligned with mass action kinetics), the rate is generally the product of the rule's rate constant and the number of possible reactions X_r. This can be easily illustrated using rule R2 based on the data in Fig. 2.3:

1. Find the number of matches for each reactant pattern:
    - 5 matches found for rule R2, pattern P1
    - 2 matches found for rule R2, pattern P2
    - 3 matches found for rule R2, pattern P3

2. Calculate the number of possible reactions X_r by multiplying the number of matches for each pattern: 5 x 2 x 3 = 30

3. Derive the rate by multiplying the number of possible reactions X_r with the rate constant of the rule: $k_f \cdot 30$

Nevertheless, there are some factors to consider that might affect this calculation, and these considerations are explained in more detail in Chapter 2.3 Rule Rate Consideration.

2.2 Reactants Sampling and Systems Update

Assuming that exactly one matching list is retained for each response pattern within each rule, the process of simulation selection becomes clear (see Fig. 2.4). One selects a rule based on the procedure described Section 1.4 Description Of Network-Free Simulations. The rules can be seen as segments of a roulette wheel with sizes proportional to their rule rates. For the rule selection, the roulette wheel is turned, in a figurative sense, and on the basis of the selected field a reaction is determined. Once a rule is selected, the next step is to determine which molecules will undergo the transformation dictated by that rule. To make this decision, a match is randomly selected given its contribution (number of molecules, number of matches within a molecule) from the respective match lists for each pattern in the rule. The rule is then applied to the selected reaction molecules in the mixture that are consistent with the selected match.
After a reaction event, the state of the system must be adjusted, as shown in Fig. 2.6. The reacting molecules are removed from the system and the newly formed products are updated into the system. Then the cycle starts from the beginning.

2.3 Rule Rate Consideration

When calculating rates in the context of rules and a pattern-matching algorithm, as opposed to traditional SSA based on species populations and responses, several important factors must be considered.

2.3.1 Symmetry Within Pattern

Patterns can be defined in a manner that allows for multiple matches to occur through simple permutations of the molecules involved. For instance, a pattern may include two identical pattern molecules bound to each other on identically named sites. In such scenarios, the same set of molecules may be matched multiple times. As illustrated in Fig. 2.7, consider a straightforward example of a dimer dissociation reaction. Within the simulation mixture, a single dimer is present, and the reactant pattern matches this dimer in such a way that the violet pattern as well as the yellow pattern correspond to the same subpattern in the molecule but on a different position due to the symmetry of the molecule. If the pattern matches are swapped, a second match between the reactant pattern and the dimer can be discovered. To accurately calculate the rate of the rule, the

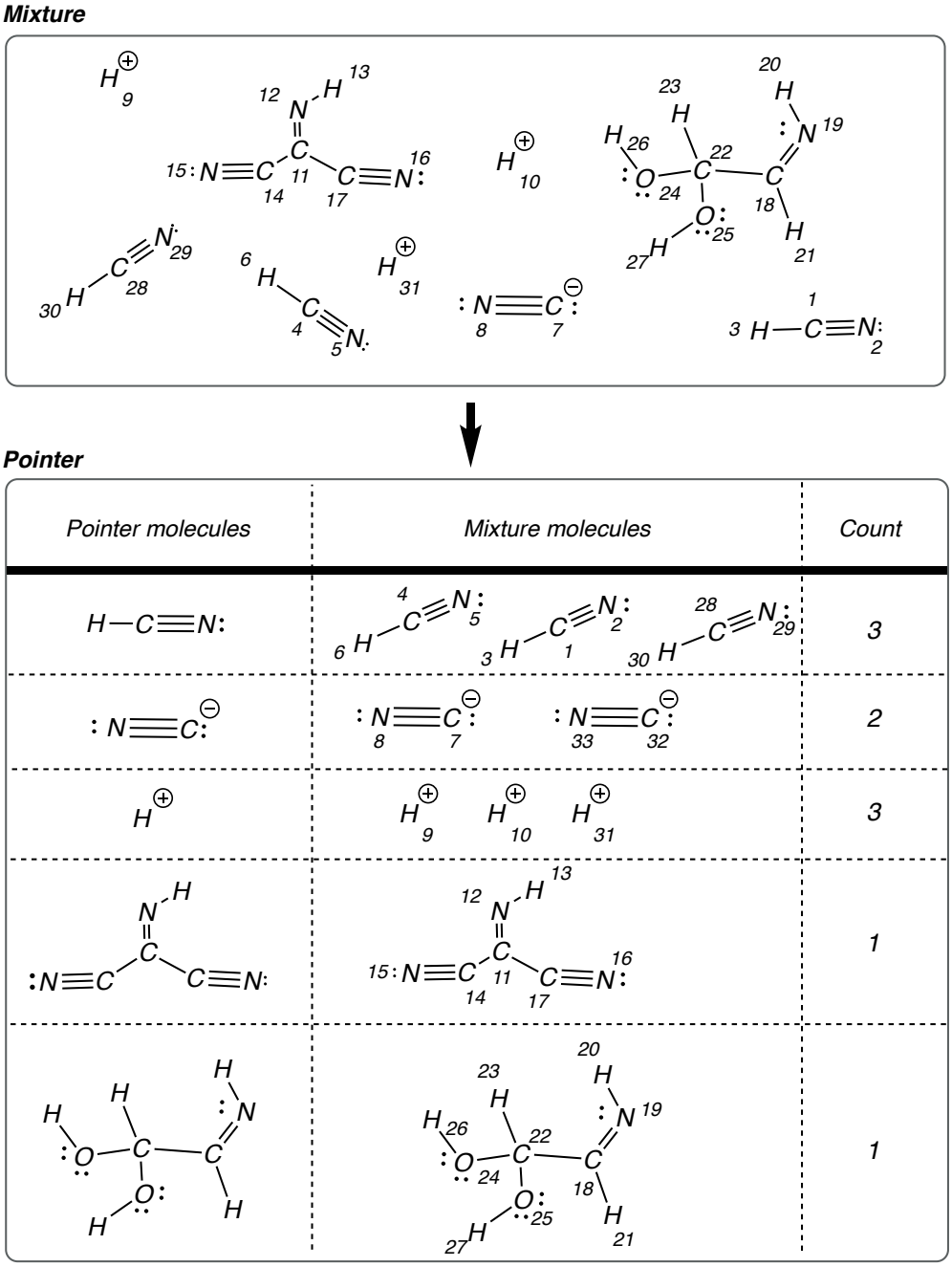

Figure 2.1: Mixture and Pointer List of a Network-free Simulation Algorithm
The composition of the mixture is stored in memory at any given moment during a simulation along with the match list corresponding to each pattern. The rules that are addressed in this context are the ones depicted in Figure 1.4. The rates for each rule can be computed according to section 2.1. Every atom in the mixture is labeled with an unique integer identifier. The pointer molecules are not labelled, since they are only used for rate calculation and not for chemical transformations

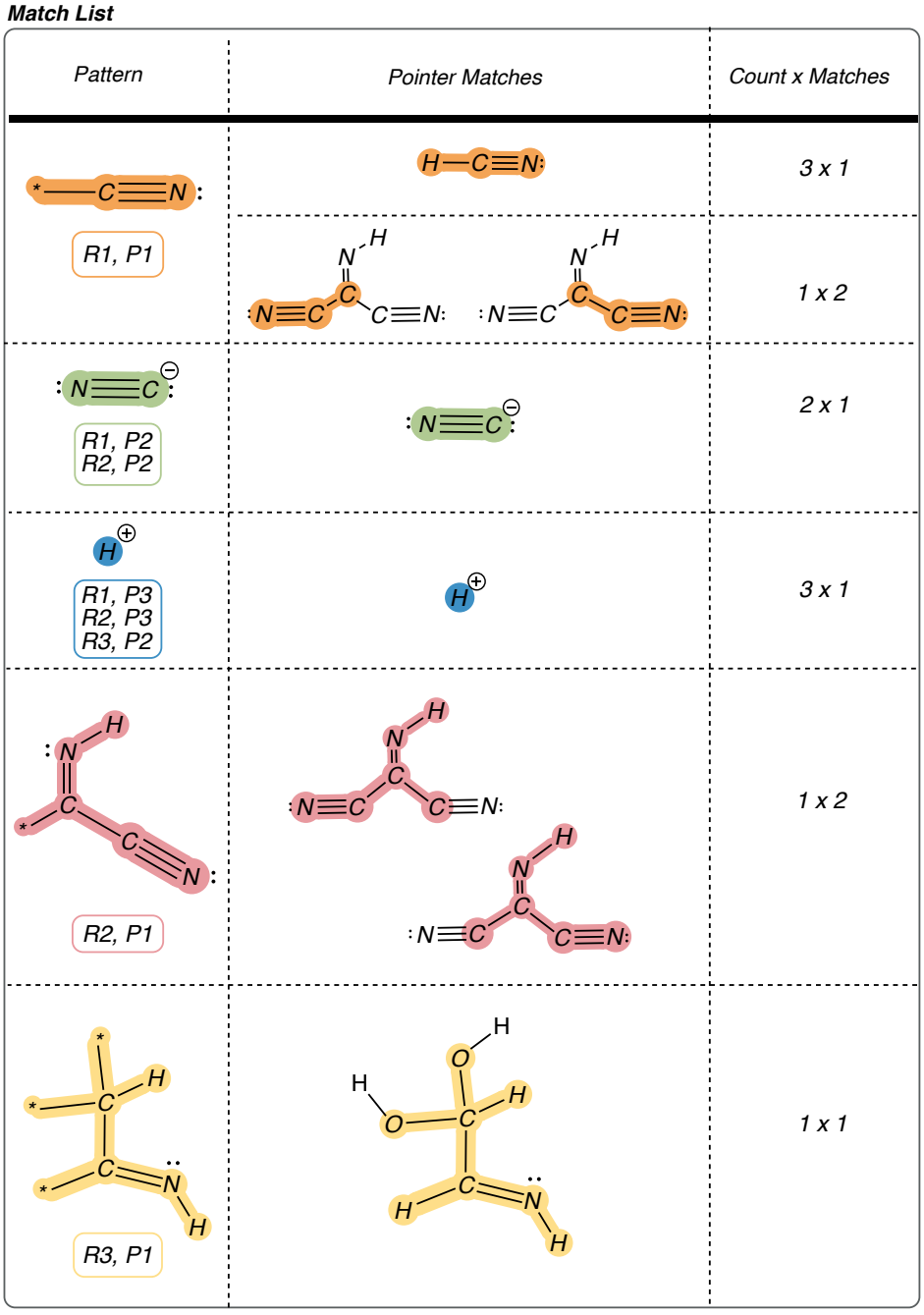

Figure 2.2: Matchlist for a Network-free Simulation Algorithm
Pattern-matching is applied on every pointer molecule from the pointer list (see Fig. 2.1). The number of matches within a one pointer molecule is multiplied with the number of the pointer molecule.

Rule Rate Calculation

Rule R		Possible Reactions X_r		Rule Constant k_r	Rule Rate r_R
R1	P1 P2 P3	(3x1) + (1x2) = 5 (2x1) = 2 (3x1) = 3	(5x2x3) = 30	k_{R1} = 1	r_{R1} = (30 x 1) = 30
R2	P1 P2 P3	(1x2) = 2 (2x1) = 2 (3x1) = 3	(2x2x3) = 12	k_{R2} = 5	r_{R1} = (12 x 5) = 60
R3	P1 P2	(1x2) = 2 (2x1) = 2	(1x3) = 3	k_{R3} = 20	r_{R3} = (3 x 20) = 60

Figure 2.3: Rule Rate calculation of a Network-free Simulation Algorithm
For the rule rate calculation, a match list according to Figure 2.2 is generated. It's important to note that "R_N, P_M" refers to the Nth rule and the Mth pattern within that particular rule. The rules which are addressed in this context are the ones depicted in Figure 1.4. The calculation of the rule rate r_R is done by multiplying the number of possible reactions X_r with the rules' constant k_r.

rate must be divided by the number of molecular permutations within the pattern that maintain binding connectivity. This corresponds to the number of ways an occurrence of the pattern in a species matches the pattern due to its inherent symmetry. It is important to note that this rate calculation follows a certain convention within our framework: the rate of a rule should be proportional to the number of different reactions that can potentially occur, and not simply to the number of matches.

2.3.2 Symmetry Among Pattern

When a rule's reactant patterns exhibit symmetry, meaning they contain two or more identical patterns, it becomes crucial to accurately determine the potential number of reactions. In Gillespie's initial notation (Gillespie, 1976 [7]), h_μ represents the count of possible reaction occurrences, and it is derived from the populations of the chemical species acting as reactants in the reaction. The rate of the reaction is then obtained by multiplying h_μ with the reaction's rate constant. Using our suggested approach for match tracking, we must employ a similar counting method to determine the number of unique match combinations, even when symmetry is present.

$$h_{sym,P} = \frac{|P|}{A_P} \tag{2.1}$$

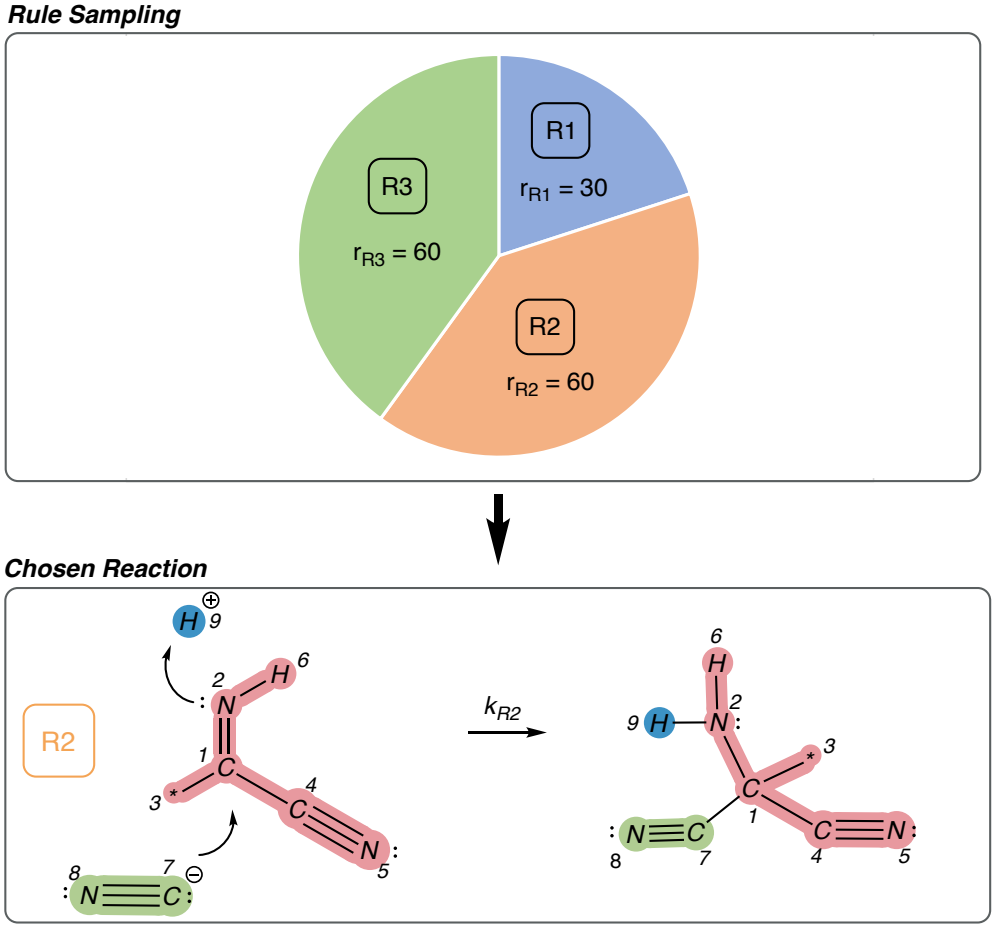

Figure 2.4: Chosen Rule of a Network-free Simulation Algorithm:
During each iteration of the simulation, a rule is selected with a probability proportional to its rate compared to the collective rate of all available rules. Subsequently, a match is selected from the match list associated with each reactant pattern within the chosen rule.[15]

A_P represents the count of the pattern P within the list of reactant patterns, and $|*|$ indicates the size of P's match list. We consider a straightforward scenario with a homodimerization rule and a rate constant k for molecule N. If the mixture is containing 500 monomeric molecules N, then the rule's rate can be calculated as follows:

$$\binom{500}{2} \cdot k = \frac{500 \cdot 499}{2} \cdot k \tag{2.2}$$

2.3.3 Molecularity

Another factor to consider is the molecularity of rules. In cases where species consist of more than one molecule within a simulation mixture, multiple patterns within a single

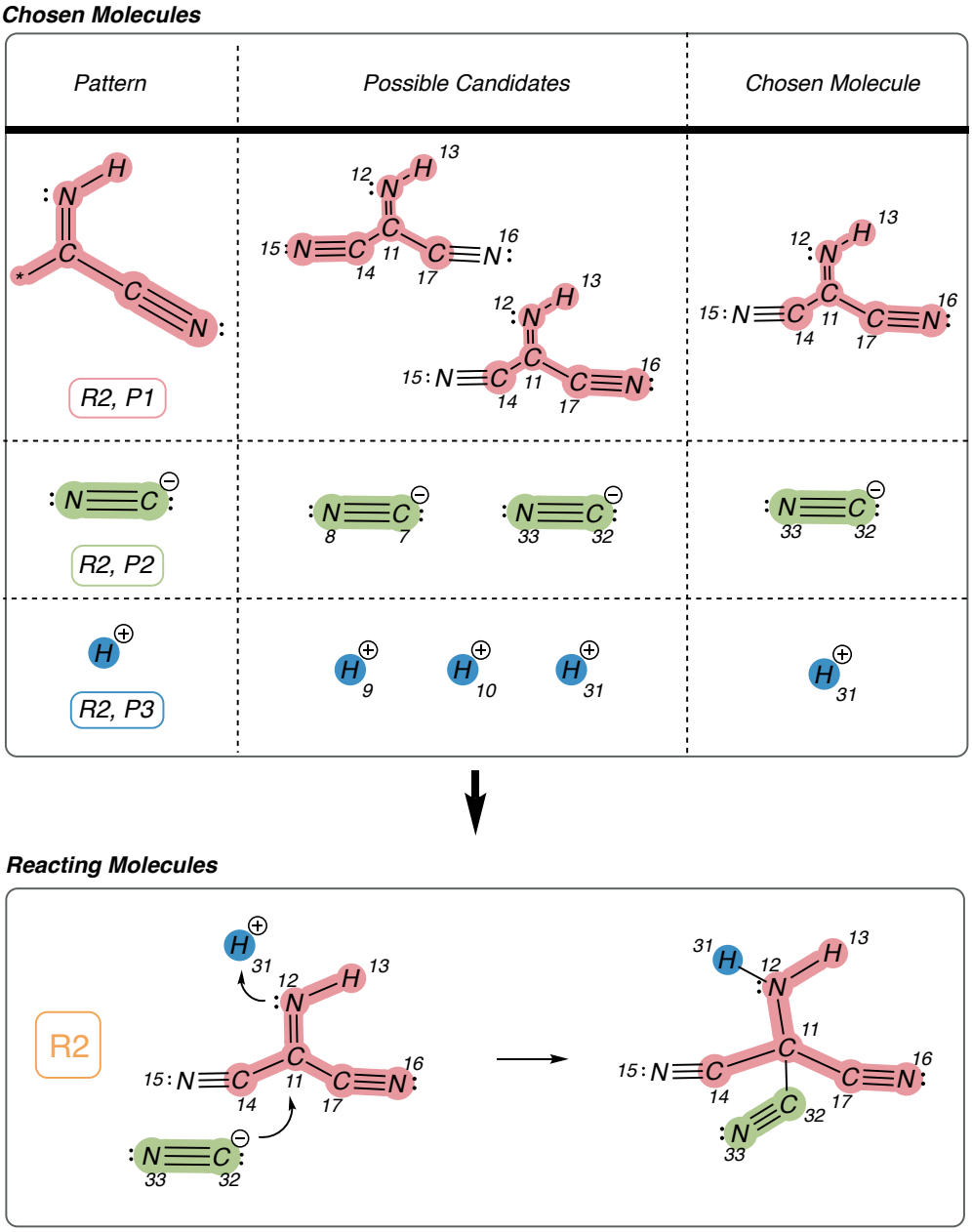

Figure 2.5: Chosen Molecules of a Network-free Simulation Algorithm:
A molecule is randomly selected from the match list associated with each reactant pattern within the chosen rule.

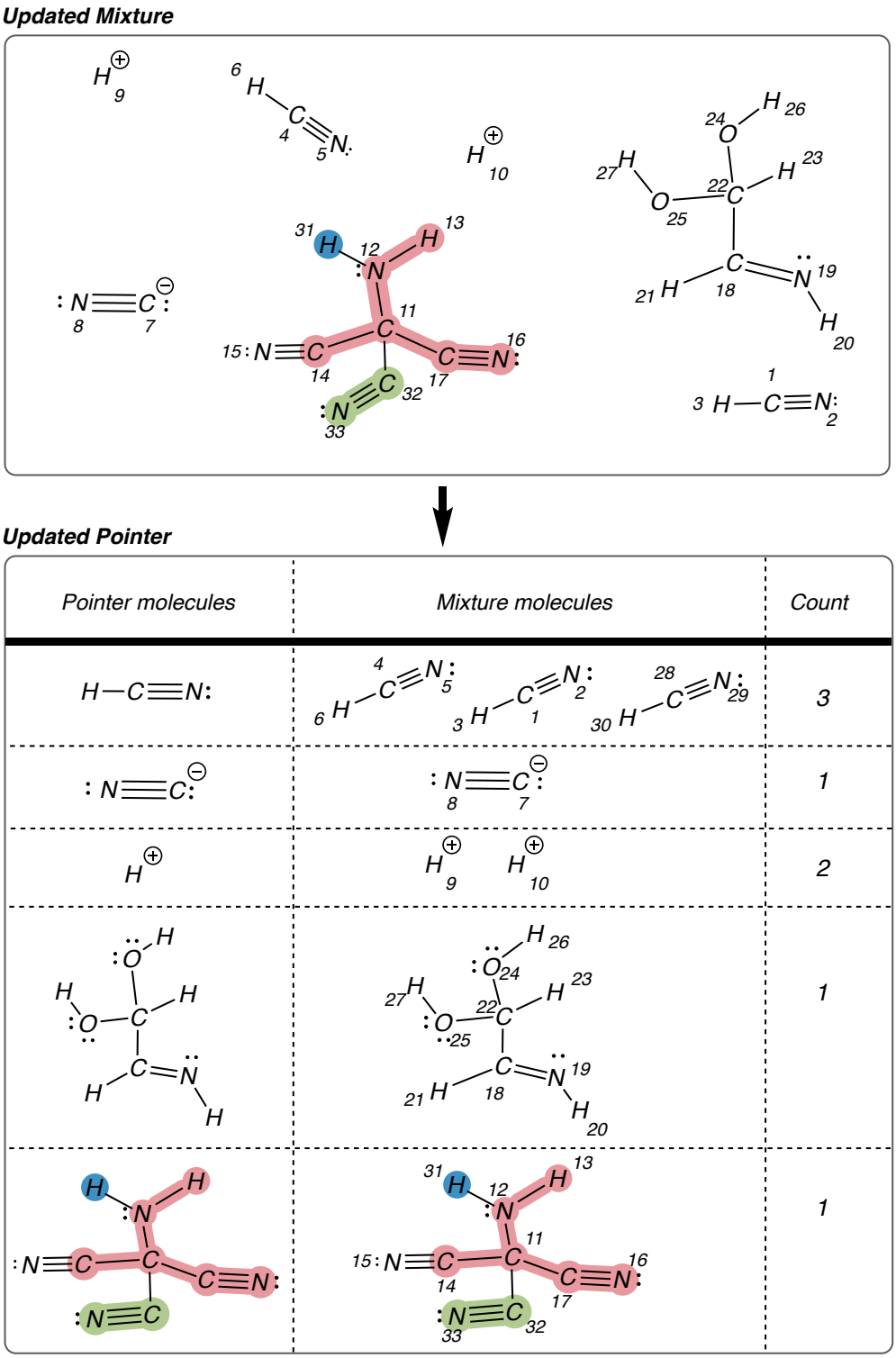

Figure 2.6: Updated Mixture and Updated Pointer List of a Network-free Simulation Algorithm: The simulation mixture undergoes an update in accordance with the transformation specified by the rule. You can observe that the species highlighted represents the outcome of this transformation. Subsequently, the match lists are also updated. For each molecule that has been modified, any matches that are no longer valid are removed, and new matches are introduced into the relevant match lists.

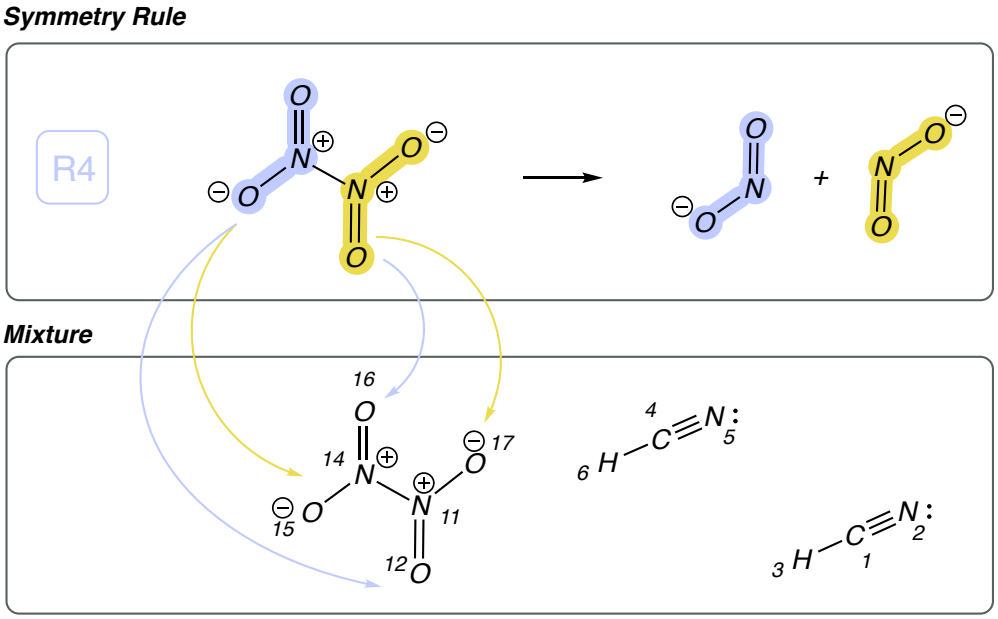

Figure 2.7: Influence on Rule Rate by Symmetry of a Reaction.
The calculation of rule rates is affected by symmetry. Symmetry within a pattern leads to the presence of multiple matches (indicated by purple and yellow arrows) between the pattern and a specific group of molecules. To accurately compute the rate of a rule it is essential to divide the rate by the number of symmetries existing within the patterns of the rule.

rule can potentially match the same species. For instance, considering a rule for a six-ring formation with pattern molecules from Figure 2.8 R5. This can arise, if the mixture contains a molecule where pattern molecules meet the patterns' criteria, but are also within the same molecule through unspecified sites in the bond formation rule (see Fig.2.8 1) Intramolecular Match). When this rule is selected for sampling, the algorithm could randomly choose matches associated with the pattern from the rule, which are part of the molecule containing multiple reaction pattern. The rule specifies that P1 (orange) should bind to P2 (red), resulting in an intramolecular cyclic structure, although this isn't evident from the reactant pattern of the rule. Additionally, the rate of cyclization, which is an *intramolecular reaction*, won't be the same as the rate of *intermolecular association*. Therefore, it's essential to differentiate between *intermolecular* and *intramolecular* bond formation. To differentiate between unimolecular and multimolecular reactions, the simulator needs to ensure that matches for a multimolecular rule belong to separate species instances and are not part of the same species instance. This can be accomplished by checking the possibility of intramolecular reactions in the matchlist (explained in Section 2.3.4 Intramolecular Reactions).

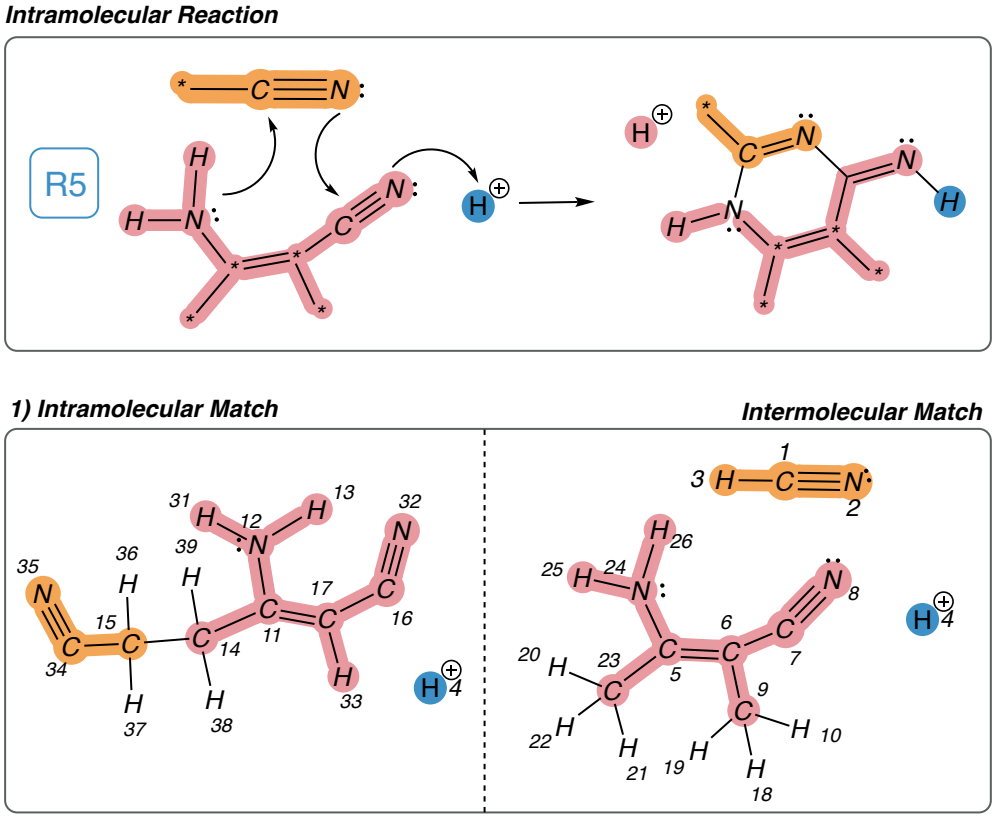

Figure 2.8: Example for Inter- and Intramolecular matches for a reaction: Two matches for the red pattern P1, two matches for the orange pattern P2 and one match for the blue pattern P3. the left reasction results in a intramolecular reaction. Blue arrows result in a intermolecular reaction

2.3.4 Intramolecular Reactions

In the issue previously discussed regarding molecularity constraints (Section 2.3.3 Molecularity), there might be instances where the rates are overestimated due to certain match choices resulting in *intramolecular* reactions (as illustrated in Fig. 2.8). To address these overestimations, certain network-free simulation methods permit the inclusion of *intramolecular* reactions. If the molecules selected through sampling do not conform to the molecularity constraints outlined in the sampled rule (as depicted in Fig. 2.8), the occurring reaction event is seen as a possible *intramolecular* reaction. If this happens some criteria for *intramolecular* reactions must to be considered:

1. Check if there is an overlap in the chosen intramolecular pattern matches (see Fig: 2.9 (2)). If the reaction patterns do not overlap, the intramolecular reaction is possible. There are cases where overlapping and not overlapping pattern are present within a molecule (Fig: 2.9 (3))

2. Recalculate the match list considering the impossible (overlapping) and possible

(not overlapping) pattern matches. Have in mind that the match list needs to be corrected for both cases possible and impossible intramolecular reactions, due to the overestimation of the pattern matches for the reaction.

3. Calculate the rule rate for *intermolecular* reactions and *intramolecular* reactions by using the adapted match list (see Figure 2.10).

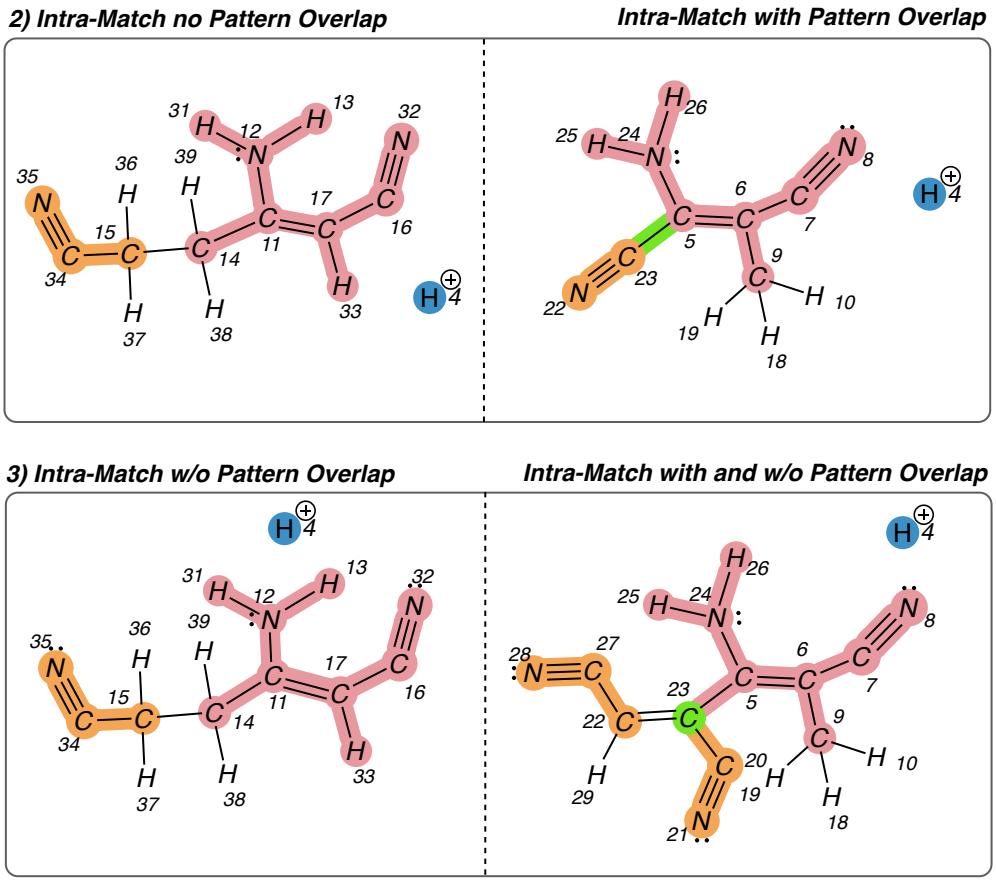

Figure 2.9: Different Scenarios for Intramolecular Reactions
(2) Example for a overlapping and not overlapping intramolecular matches. Molecules N2 and M2 have an overlap (green) site. Both pattern orange and red match the molecules from the mixture. The left molecule has no overlap in the reaction pattern (orange & red seperated). The right molecule has an overlap in the reacting sites (green; atom ID: 5 & 23) (3) Example for overlapping and not overlapping intramolecular matches and intermolecular matches. The right molecule has one overlap for the red and orange patter at the atom with ID: 5 (green). The second orange match (atom ID: 22, 27, 28) is not overlapping with the red pattern. Also intermolecular reactions need to be considered for both mixtures since the pattern from the left molecule can react with the right molecule and the other way around

Generally, following equations can be used for the calculation of rule rates for reactions

Rule Rate Calculation

Reaction Type	Match Type	Possible Reactions X_r
inter- and intramolecular w/o overlap	sum	(2x2x1) = 4
	intramolecular	(1x1x1) = 1
	intermolecular	(4 - 1) = 3
intramolecular with overlap and w/o overlap	sum	(2x2x1) = 4
	intramolecular w/o overlap	(1x1x1) = 1
	intramolecular with overlap	(1x1x1) = 1
	intermolecular	(4 - 1 - 1) = 2
intermolecular, intramolecular with overlap and w/o overlap	sum	(3x2x1) = 6
	intramolecular w/o overlap	(1x1x1) + (1x1x1) = 2
	intramolecular with overlap	(1x1x1) = 1
	intermolecular	(6 - 2 - 1) = 3

Figure 2.10: Calculation of matches and further rule rates for different scenarios derived from Figure 2.8 and Figure 2.9 for intramolecular reactions

where inter- and intramolecular bond formation could be possible:

$$r_R = (P_N * P_M) - P_{nO} - P_O \tag{2.3}$$

The sum of the possible reactions X_r for a certain rule R is the product of all molecules which match with the pattern of the reaction (product of P_N and P_M: described in 2.1 Rule Rates Calculation). From this product, the number of overlapping pattern matches P_O and the number of not overlapping pattern matches P_{nO} is subtracted. The number of not overlapping pattern matches P_{nO} is then used to calculate the rule rate of the intramolecular reactions' rule rate. If the reaction is bimolecular and both pattern are present in the matched molecule (see Fig. 2.8), then P_{nO} multiplied with the rule rates' constant becomes the rule rate. Having other pattern involved in the intramolecular reaction which are not a part of the intramolecular reacting molecules (e.g. H^+), the rule rate is calculated as described 2.1. Rule Rates Calculation. It is important to say, that the rule rates' constants for intramolecular reactions vary from intermolecular reaction because of the spatial nearness of the reacting pattern within a intramolecular reaction. It can be simplified a faster "finding" of the reacting pattern in intramolecular reactions compared to intermolecular reactions.

3 Setup

The primary objective of this project is the development and implementation of a network-free Stochastic Simulation Algorithm (SSA) utilizing rewrite rules, achieved through the utilization of `RDKit` within the Python programming environment. Furthermore, a crucial aspect of this endeavor involves atom tracking for all reacting molecules with `RDChiral`, resulting in the labeling of each atom within the mixture. This detailed atom tracking not only provides information about the current spatial position of each atom but also offers insights into its historical trajectory within the system.
Given the computationally intensive nature of the algorithm, the creation of an output file becomes essential. This output file serves as a comprehensive record, encompassing the requisite data for reconstructing the system at any given point in time or iteration cycle during the simulation. This capability allows users to perform simulations once and subsequently conduct in-depth analyses of various system characteristics, without the need for repeated simulations. In essence, the generated output file can be conceptualized as a temporal continuum, affording users the flexibility to navigate forward and backward in time or between iteration cycles. This feature not only facilitates a deeper understanding of the system's dynamics but also enables a more efficient and comprehensive exploration of its behavior and properties.
The data extracted from the output file provides comprehensive insight into the trajectories of molecules within the system. Leveraging these trajectories, it becomes possible to investigate the potential pathways leading to the transformation of an initial molecule into a specific target molecule and reconstruct mechanisms. This process includes the discovery of reaction sequences that culminate in the formation of the desired molecule, which is of particular relevance in the context of biological systems. These identified sequences, often referred to as motifs, offer valuable insights into novel mechanisms governing molecule formation within biological systems, thereby contributing to the exploration of previously uncharted routes in chemical and biochemical processes.

3.1 RDKit and RDChiral

As described above the focus of this project is the implementation of a network-free Stochastic Simulation Algorithm (SSA) with a particular emphasis on atom tracking within chemical systems. Central to this endeavor is the utilization of `RDKit`, a powerful and versatile cheminformatics toolkit in the Python programming environment. `RDKit` offers a wide array of functionalities, making it an invaluable tool for molecular informatics and computational chemistry.
`RDKit` provides an extensive set of capabilities, including molecular structure manipulation,

substructure searching, and chemical property prediction. Its rich feature set enables users to work with molecular structures, reactions, and chemical data efficiently. In the context of this project, `RDKit`'s pattern matching capabilities and reaction function play a crucial role in achieving the stated objectives. Pattern matching in `RDKit` involves searching for specific molecular substructures or motifs within a larger chemical structure or reaction. This process is integral to understanding the intricate details of chemical transformations, especially within stochastic simulation algorithms. `RDKit` facilitates this by enabling users to define and search for patterns based on atom types, bond types, and other molecular attributes by using so called SMARTS-Strings. The molecules are represented as SMILES-Strings.
In the context of the project, `RDChiral` is used for atom tracking, which involves labeling each atom in a mixture. `RDChiral` is a valuable extension of the `RDKit` toolkit, designed to enhance the handling of stereochemistry in chemical systems [source: GitHub - connorcoley]. Additionally, `RDChiral` serves as a powerful tool for atom tracking following chemical reactions, ensuring the preservation of crucial atom label information. This atom labeling allows for precise tracking of the position and trajectoral history of individual atoms within the system. `RDKit`'s pattern matching capabilities are harnessed to identify and label atoms as they participate in chemical reactions during the simulation. These labeled atoms provide a comprehensive view of the system's dynamics, essential for both the accurate representation of chemical processes and subsequent analysis.
Furthermore, the project's objective of generating an output file that has the information of the system's state at different time points or iteration cycles is facilitated by `RDKit`'s and `RDChiral`'s ability to capture and record the atom labels and their positions. This capability enables users to navigate through the system's temporal evolution, offering a unique opportunity to analyze and understand the behavior of molecules within the simulated environment.

3.1.1 SMILES Molecules and SMARTS Pattern

SMILES (Simplified Molecular Input Line Entry System) and SMARTS (SMILES Arbitrary Target Specification) are notations used in chemistry and cheminformatics to represent chemical structures and patterns. `RDKit` provides tools and libraries for working with chemical data, including support for SMILES and SMARTS.

SMILES (Simplified Molecular Input Line Entry System)

SMILES is a text-based notation for representing chemical structures in a concise and human-readable format. It represents molecules as a linear string of characters, where atoms and bonds are denoted by specific symbols: "H" hydrogen, "O" oxygen and "C" carbon and bonds are represented as "-" or "" single bond e.g. OH, "=" double bond e.g. C=C and "#" triple bond e.g. C#N provides tools to convert between SMILES and molecule objects, perform substructure searches, and manipulate molecules using SMILES.

SMARTS (SMILES Arbitrary Target Specification)

SMARTS is an extension of SMILES designed to represent chemical patterns or substructures within molecules rather than entire molecules. It uses special symbols and characters to define patterns that can match specific substructures in molecules. SMARTS patterns are often used for substructure searching, reaction matching, and querying chemical databases. Examples of SMARTS patterns:

- "[C]": Matches any (not aromatic) carbon atom.
- "[C][O]": Matches a carbon atom bonded to an oxygen atom.
- "[C;X3]": Matches a carbon atom with exactly three bonds.

`RDKit` provides tools to search for SMARTS patterns within molecules, identify substructures, and perform various cheminformatics tasks. For more information on SMILES and SMARTS string visit this website on how to use RDKit: RDKit Cookbook

3.2 Algorithm Implementation

The pattern-based SSA (Stochastic Simulation Algorithm) in `RDKit` employs the `Python` programming language as the primary environment for information storage and data processing. The utilization of `RDKit`'s functionalities, including molecule generation, pattern matching, and chemical transformations, are mandatory in constructing a chemical reaction network.

3.2.1 Generated Molecules and Defined Reactions

As described in 1.5 Implementation of Network-Free Simulations, the initial phase is the initialization of the system. This initialization process necessitates the generation of molecules that represent the constituents of the system of interest. The molecular generation is accomplished through the utilization of Simplified Molecular Input Line Entry System (SMILES) notations. Detailed instructions for this procedure are described in Chapter 4 Application.
Every atom within the generated molecules will receive a unique integer label. In the event that the system contains a total of 50 atoms, these labels will span from 1 to 50. This imperative labeling process serves as an essential requirement for subsequent analytical procedures, specifically for the purpose of reconstructing trajectories. The molecules, now uniquely labeled, are represented in `SMILES` format as follows:

```
'[N:1]#[C:2][H:3]'
'[N:4]#[C:5][H:6]'
'[N:7]#[C:8][H:9]'
'[N:10]#[C:11][H:12]'
        :
        :
```

The following process step involves the definition of chemical transformations expressed as pattern-based SMARTS strings. Comprehensive guidelines for this process are provided in Chapter 4 Application. Elaboration of the characteristics of these transformations is of utmost importance for the configuration of the simulation framework. The configuration properties of these transformations play a crucial role in determining the system behavior. Depending on the hierarchy of specificity assigned to these reactions, the system can behave either rigidly or dynamically. A higher degree of specificity within a reaction gives the system greater rigidity, while lower specificity gives it a more dynamic character. Therefore, it is essential to carefully evaluate the required specificity for each reactant (pattern) involved in each reaction before starting a simulation. If certain reactants have a greater influence on the reaction and the overall system, it is advisable to increase the specificity for these patterns. If, on the other hand, other reactants play a lesser role or contribute to the dynamics of the system, it is advisable to decrease their specificity (compare Fig. 1.2 (2) Specification Hierarchy). Examples of pattern-based SMARTS are given below.

```
oro1 = '([H+0:3][C-0:1]#[N:2])>>[C-:1]#[N:2].[H+:3]'

oro1_rv = '([H+:3].[C-:1]#[N:2])>>[C-0:1](#[N:2])[H+0:3]'
```

Each reaction requires the assignment of a unique identifier or name. The reaction structure itself is divided into two distinct segments: the educt side, referred to as the left-hand side (LHS), and the product side, referred to as the right-hand side (RHS). The use of parentheses to encapsulate the LHS conforms to programmatic syntax conventions. In cases where multiple molecular pattern are involved in the reaction, the presence of dots signifies the initiation of subsequent reactant patterns. The delineation between the reactant pattern (LHS) and the product pattern (RHS) is expressed by a pair of "greater than" signs, which function analogously to the reaction arrow in classical chemical reactions.
The final step in the setup process involves defining the reaction rate constants. By default, each constant for each reaction is initially set to the value "1". However, these constants can be changed in the file created specifically for this purpose according to a kinetic tendency of the reaction. It is also possible to use existing reaction constants from literature. Once all of the above aspects are configured, the first phase of system initialization is complete and the way is paved for the simulation to begin. However, before discussing the simulation procedures, a comprehensive explanation of the structure and functionality of the code is provided. This detailed description is intended to improve understanding of the generated data and facilitate subsequent analysis.

3.2.2 Pattern-Matching and Rule Rate Calculation

After system initialization, it is imperative to calculate the rule sets. Therefore, reactions previously defined via the SMARTS notation are parsed to delineate the left-hand side (LHS) from the right-hand side (RHS). The LHS patterns for each individual reaction are

catalogued in a dictionary. Representative molecules (hereafter referred to as "pointers") must then be identified for each type of molecule present in the mixture. It is also necessary to quantify each of these species. These pointer molecules facilitate pattern matching with the patterns compiled in the pattern dictionary relevant to the reaction. Pattern matching is performed using the `RDKit` library. This process results in a structured list containing the pointer molecule, the corresponding matching pattern, the frequency of matches within a single molecule and the associated reaction (as shown in the Figure 3.1). The rule rate of a given reaction quantifies the number of possible reactions multiplied with the reaction constant corresponding to that specific rule. A reaction is possible when the match list for every pattern associated with that reaction is not empty. For each pointer molecule that aligns with a pattern, a number of all possible matches within a species population S_{mol} emerges. The sum of all molecular matches S_{mol} from all species for a certain pattern i is then the sum of possible reacting pattern for this particular pattern P_i:

$$P_i = \sum_i S_{mol,i} \tag{3.1}$$

The derivation of the possible matches for the pattern can be seen in Figure 2.2. The simulation outputs a structured list describing the pattern match data for each pointer molecule during a given cycle. This output serves as a diagnostic tool during the initial simulation runs to ensure the correct functioning of the defined system, its individual molecules and associated rules. It is worth noting that this output data is volatile and is not fully saved in a file; rather, it acts as a control mechanism. The calculation of rule rates is then done as described in Section 2.1. At the

3.2.3 Time Advancing and Reaction Sampling

After calculating the rule sets for all possible reactions, the system proceeds to determine the time interval until the next rule application and samples the chosen rule. For this purpose, a random number u_0, is generated within the interval [0,1), which determines the sampling according to Equation 1.2. At the same time, a random number u_1 within the interval [0,1), is generated for rule sampling and subjected to sampling according to Equation 1.3. Since this approach is is pattern-based, the molecules involved in the reactions are selected randomly. Consequently, any molecule included in the reaction pattern matchlist is eligible for random selection. The probability for this selection depends on the number of possible matches for a given pattern within the species population S_{mol}. For each reaction pattern, one molecule is selected from the mixture and in the same time removed from the mixture. A single molecule may contain multiple reactive patterns in its structure, allowing the possibility of intramolecular reactions as described in Section 2.3.4 Intramolecular Reactions. The simulation assesses the possibility of such reactions and distinguishes between intramolecular reactions and their intermolecular counterparts. The parameters for these intramolecular reactions are explained in a latter section.

```
Cycle: 3:
-------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------
reacting system
-------------------------------------------------------------------------------------------------
Atom Counter({'H': 75, 'C': 75, 'N': 75})

single bonds: 76 || double bonds: 1 || triple bonds: 74 || aromatic bonds: 0

-------------------------------------------------------------------------------------------------
Molecule Counter({'[H][C]#[N]': 60, '[H][N]=[C]([H])[C]#[N]': 4,
'[H][N]([H])[C]([H])([C]#[N])[C]#[N]': 2, '[H+]': 1, '[C-]#[N]': 1})
-------------------------------------------------------------------------------------------------

molecule identity: [H][C]#[N]

[H][C]#[N] || ((0, 1, 2),) || Matches: 1 || Reaction: ['oro1']
[*][C]#[N] || ((0, 1, 2),) || Matches: 1 || Reaction: ['oro2', 'oro6']
-------------------------------------------------------------------------------------------------
molecule identity: [H][N]=[C]([H])[C]#[N]

[*][C]#[N] || ((2, 4, 5),) || Matches: 1 || Reaction: ['oro2', 'oro6']
[*][C]([C]#[N])=[N][H] || ((5, 4, 2, 3, 1, 0),) || Matches: 1 || Reaction: ['oro2_rv']
[*][C]([*])=[N][H] || ((3, 2, 4, 1, 0),) || Matches: 1 || Reaction: ['oro3']
-------------------------------------------------------------------------------------------------
molecule identity: [H][N]([H])[C]([H])([C]#[N])[C]#[N]

[*][C]#[N] || ((3, 5, 6), (3, 7, 8)) || Matches: 2 || Reaction: ['oro2', 'oro6']
[*][C]([*])([C]#[N])[N]([H])[H] || ((4, 3, 5, 6, 7, 1, 0, 2), (4, 3, 7, 8, 5, 1, 0, 2)) ||
Matches: 2 || Reaction: ['oro3_rv']
-------------------------------------------------------------------------------------------------
molecule identity: [H+]

[H+] || ((0,),) || Matches: 1 ||
Reaction: ['oro1_rv', 'oro2', 'oro3', 'oro4', 'oro4_rv', 'oro6', 'oro6_rv', 'oro7', 'oro7_rv']
-------------------------------------------------------------------------------------------------
molecule identity: [C-]#[N]

[C-]#[N] || ((0, 1),) || Matches: 1 || Reaction: ['oro1_rv', 'oro2', 'oro3']
-------------------------------------------------------------------------------------------------
reaction oro1 : possible
reaction oro2_rv : possible
```

Figure 3.1: Example Simulation of Pointer List and Matchlist
The displayed output illustrates the first part of structural framework of the simulation for one cycle. First, the `Cycle Number`, `Atom Counter` and `Bond Counter` is printed. After that, the `Molecule Counter` is seen, which corresponds to the Pointer List shown in Figure 2.1. The output after that correspond to the `Matchlist` explained in Section 2.1. At the end all possible reactions are listed.

```
-------------------------------------------------------------------------------------------
Time Advancing and Molecule Selection
-------------------------------------------------------------------------------------------
rule rates =  {'oro1': 73, 'oro2': 292, 'oro1_rv': 4}

sampling time: 0.005047022024056438

reaction time: 0.036363214443343554

Due to the rule rates of the reactions, following reaction is chosen: oro2
-------------------------------------------------------------------------------------------
These are the subpattern of oro2: ['[C-]#[N]', '[H+]', '[*][C]#[N]']

The product of molecule count x number patttern for each molecule: {'[H][C]#[N] [*][C]#[N]': 73,
'[H+] [H+]': 2, '[C-]#[N] [C-]#[N]': 2}

The selected molecules for reaction oro2: ['[H+]', '[C-]#[N]', '[H][C]#[N]']

-------------------------------------------
New Molecule Counter({'[H][C]#[N]': 72, '[H+]': 1, '[C-]#[N]': 1})
----------------------
reacting molecules:
[N:145]#[C:146][H:147]
[H+:144]
[N:142]#[C-:143]
------------------------
```

Figure 3.2: Example Time Advancing and Molecule Selection
This displayed output illustrates the second segment of the structural framework of the simulation for a single cycle. First, metrics such as the `Sampling Time` (which indicates the temporal progression from the current to the subsequent reaction), the `Reaction Time` (the sum of the temporal progress over all cycles within the reactive system) and the selected reaction are displayed. This is followed by the display of the `Reacting Subpattern` corresponding to the selected reaction. Next, the pointer molecules undergoing the reaction are determined, with S_{mol} serving as the probabilistic weight. Finally, the labelled reacting molecules of the system are sampled to prepare them for the upcoming transformation.

3.2.4 Transformation and Atom Tracking

The transformation rule is applied to the molecules with labelled atoms. The specific reaction (here labelled `oro2`) is shown together with the corresponding SMARTS reaction in Figure 3.3. It is important to recognise that the labelling embedded in the SMARTS pattern is crucial in describing the mechanisms; however, it is not related to the atomic labels contained in the molecules. These atomic labels are superimposed on the SMARTS pattern so that all feasible reaction products are build. In scenarios where different reaction products occur (due to variations in the embedding of the pattern) or where several potential product combinations exist (due to the presence of more than one pattern on the product side), a product or product combination is selected randomly. After a product or product combination is selected the product or the product combination is added to the mixture indicating the beginning of the next cycle.

```
-----------------------------------------------------------------------------------------
Transformation and Atom Tracking
-----------------------------------------------------------------------------------------
oro2
reaction template:
([C-:1]#[N:2].[H+:3].[*:4][C:6]#[N:5])>>[N:2]#[C-0:1][C:6](-[*:4])=[N:5]-[H+0:3]

This is the possible outcome of the reaction as mapped products
[N:142]#[C:143][C:146](=[N:145][H:144])[H:147]
-----------------------------------------------------------------------------------------
Chosen product:
[N:142]#[C:143][C:146](=[N:145][H:144])[H:147]
```

Figure 3.3: Example Transformation and Atom Tracking
After selecting the reactants from the mixture, the transformation rule is applied. Since all atoms are labeled in the educts, all atoms in the products are labeled as well. The chosen product is added to the mixture.

3.3 Algorithm Limitations

Having explained the potential benefits of this network-free SSA, it is imperative to recognise its inherent limitations. Some of these limitations have already been addressed, but this section will highlight their importance in a comprehensive manner.

3.3.1 Simulation of Huge Systems

A notable limitation of this algorithm concerns its efficiency in simulating systems characterised by a high number of molecules and/or large molecular sizes. In particular, when the initial mixture of the system includes a large number (on the order of tens of thousands) of molecules, or when the molecules to be analysed are exceptionally large (as in the case of elongated polymers or proteins), the ability of the programm to perform multiple simulation cycles within an acceptable time frame decreases with the size of the

system. However, if one can afford extensive computational resources, this limitation becomes less consequential. The main objective was therefore to develop a mechanism for simulating finite population numbers in a relatively small system, typically involving hundreds of (bio)molecules.

3.3.2 Systems with Strictly Dynamical Character

In systems where reaction transformations or mechanisms remain unclear and the network does not define a structured reaction framework, the accurate representation of the system could be compromised due to the inherent pattern orientation of the algorithm. If an excessive number of molecules react with each other within the system, this may lead to a proliferation of products that deviate from real scenarios. Therefore, careful definition of the rules and transformations that are part of the system is of utmost importance. Before simulation, these specifications should be rigorously validated against empirical data.

3.3.3 Systems with Strictly Rigid Character

Conversely, the search for innovative reaction pathways, pathway motifes or products becomes impracticable when the target system is constrained within a strictly rigid framework. Such rigidity often arises when response patterns are hierarchical high specificity and allow only minimal variability in pattern matching. If the system is only equipped to work within predefined trajectories, it will inevitably follow such paths. Circumventing this rigidity is critical in defining the system, as adherence to such bounded trajectories is not consistent with the fundamental goals of the simulation algorithms.

3.3.4 Intra- and Intra-Intra Reactions

As previously highlighted, the algorithm can differentiate between intramolecular and intermolecular reactions. An intramolecular reaction within a rule is characterized by one of the reactants matching multiple patterns from a specific transformation (or rule). A detailed categorization of various intramolecular reactions is described in Section 2.3.4 Intramolecular Reactions. An borderline case within intramolecular reactions could emerge when the count of reacting patterns exceeds three. The complexity arises when two distinct molecules match two patterns each from the transformation, leading both molecules to undergo different intramolecular reactions within the same transformation. This unique scenario is referred to as an `Intra-Intra Reaction` in this context. The simulation is able to recognise the scenario represented by `Intra-Intra-Reactions`. In cases where the number of reactant patterns within a rule exceeds 5 though, a potential third intramolecular reactant may be involved in the reaction and the simulation is not able to perform the appropriate transformation for `Intra-Intra-Intra Reactions`. Given the rarity of such events in typical reactions - most involve the participation of 3 to 4 patterns - this rarely poses much of a challenge. However, for systems characterised by a high number of patterns within a single reaction, two strategies can be used to avoid potential complications:

1. Extend the specificity hierarchy and lengthen the patterns. This refinement aims to exclude intramolecular matches for multiple patterns and introduce a single pattern match.
2. Break down reactions with a large number of patterns into more manageable reactions using intermediates.

4 Application

The network-free stochastic simulation algorithm with atom tracking is a powerful tool for studying molecular behavior within a given system, eliminating the need for a comprehensive system definition in terms of reactions and molecules involved. This chapter describes the use of the simulation algorithm, starting with the installation of the required modules and libraries for the `Python` environment.

4.1 Installation of RDKit and RDChiral

The first step is the installation of the `RDKit` module and `RDChiral` module. This was done in `Terminal` with following command line input:

```
pip install rdkit

pip install rdchiral
```

There are also other ways to install these modules, which can be found here:

- RDKit
- RDChiral

After the modules are installed the next step is to download the `rdchiral` folder from GitHub repository of `connorcoley`. Also download the files:

- `atom_mapped_reactions.py`
- `reactions_and_constants.py`
- `molecule_generator.py`

from the GitHub repository of `ggerber`. Put these files in the same directory e.g. the rdchiral folder from `connorcoley` and start with the implementation of the system.

4.2 `molecule_generator.py`

Open the Python file `molecule_generator.py` and adapt the molecules you want to use for your systems. Read the instruction given in the file for the generation of molecules:

```
###############################################
reactant_smiles_1 = 'N#C'
number_smiles_1   = 75

reactant_smiles_2 = ''
number_smiles_2   = 3

reactant_smiles_3 = ''
number_smiles_3   = 10

reactant_smiles_4 = ''
number_smiles_4   = 4

reactant_smiles_5 = ''
number_smiles_5   = 4

smiles_list       = [reactant_smiles_1, reactant_smiles_2,
                     reactant_smiles_3, reactant_smiles_4, reactant_smiles_5]

num_mols_list     = [number_smiles_1, number_smiles_2,
                     number_smiles_3, number_smiles_4, number_smiles_5]
###############################################

input_smiles_1 = 'N#CC(N)=C(N)C#N'
number_input_1   = 20

input_smiles_2 = ''
number_input_2   = 1

input_list = [input_smiles_1, input_smiles_2 ]
num_input_list = [number_input_1, number_input_2]
```

Incorporate your SMILES representations of molecules into the designated variable, denoted as `reactant_smiles_#`, by inputting them as string values. Concurrently, for each of these reactants, define the corresponding quantity using the `number_smiles_#` variable. The script has predefined placeholders for five distinct reactant SMILES strings and their respective quantities. An absence of a string in the placeholder results in no molecule generation for that specific reactant. Should your system demand more than five reactants, extend the list by appending, for instance, `reactant_smiles_6` to the `smiles_list`. Similarly, append the respective quantity, such as `number_smiles_6`, to the `num_mols_list`.
The variable `input_smiles_#` is intended for the temporal integration of molecule species within the simulation framework. This function is useful in analysing the system dynamics in response to the exogenous introduction of molecules at specific time intervals. Such

additions can modulate the reactivity of the system by increasing or decreasing certain reaction pathways depending on the type of species added. The script has predefined placeholders for two distinct reactant SMILES strings and their respective quantities. This variable can be changed according to the above procedure.

4.3 reactions_and_constants.py

In the subsequent phase, the reactions (rules, transformations) and the associated reaction constants are defined. This configuration is done in the file `reactions_and_constants.py`. Access this file and describe the rules of the system in `SMARTS` format as described in the introductory section of the file.

```
oro1 = '([H+0:3][C-0:1]#[N:2])>>[C-:1]#[N:2].[H+:3]'

oro1_rv = '([H+:3].[C-:1]#[N:2])>>[C-0:1](#[N:2])[H+0:3]'

oro2 = '([C-:1]#[N:2].[H+:3].[*:4][C:6]#[N:5])>>
        [N:2]#[C-0:1][C:6](-[*:4])=[N:5]-[H+0:3]'

oro2_rv = '([N:2]#[C-0:1][C:5]([*:3])=[N:4][H+0:6])>>
            [C-:1]#[N:2].[H+:6].[C:5](#[N:4])[*:3]'
                              :
input_reaction1 = '([*:1][C:2]#[N:3].[Hg+:4]])>>[*:1][C:2]=[N:3][Hg+0]'

input_reaction2 = '([*:1][C:2]=[N:3][Hg+0])>>[*:1][C:2]#[N:3].[Hg+:4]'

reaction_dict = {oro1    : 'oro1',
                 oro1_rv : 'oro1_rv',
                 oro2    : 'oro2',
                 oro2_rv : 'oro2_rv', ...,
                 input_reaction1 : 'input_reaction1'
                 input_reaction2 : 'input_reaction2'}

reaction_constant = {'oro1': 1,
                     'oro1_rv': 1,
                     'oro2': 1,
                     'oro2_rv': 1, ...,
                     'input_reaction1' : 100
                     'input_reaction2' : 1}

reactions = [oro1, oro1_rv, oro2, oro2_rv, ...]

input_reactions = [input_reaction1, input_reaction2]
```

This example reaction network is provided to illustrate the structure and syntax of the rule definition. This specific example represents a part of the Oro-Orgel synthesis of adenine, and the reactions are referred to as `oro#` for clarity. While it is possible to change the variable names to better suit other reactions or systems, the appropriate changes must be made in the associated dictionaries `reaction_dict` and `reaction_constant`. The `reaction_dict` transforms `SMARTS` reaction variable e.g. `oro2` into the string of this reaction variables name `oro2`. Also the varibale `input_reaction` can be used to introduce new reaction rules during the simulation. The possibilities of this feature are described in Section 4.4 `atom_mapped_reactions.py`. The `reaction_constant` dictionary defines the the reactions' constants. This can be adopted but is initially set to 1 for every reaction. It is also important that all desired reactions are included in the `reactions` list. The same procedure is applied to the `input_reactions` list with the input reactions. Here is a schematic syntax description of the reaction `SMARTS`, where the LHS of the reaction is always in parenthesis even if it is one reactant:

```
reaction_name = '([EDUCT1].[EDUCT2]) >> [PRODUCT1].[PRODUCT2]
```

4.4 atom_mapped_reactions.py

After determining the reactants that build the initial mixture and establishing of transformation rules, the next phase is to start the simulation. This is done by running the script `atom_mapped_reactions.py`. Change to the appropriate directory and enter the following command, making sure to replace `#CYCLES OF SIMULATION` with the desired integer value indicating the number of cycles in the command line. Additional reactants can be included in the simulation at specific points in time by adjusting the `#CYCLES FOR INPUT MOLECULES` parameter in the `atom_mapped_reactions.py` command line. By assigning this parameter to a specific cycle number or a list of numbers, the introduction of new molecular units into the system is initialised at the defined cycle in the simulation timeline. If the `#CYCLES FOR INPUT MOLECULES` parameter is left unassigned, the simulation will proceed without introducing any molecules into the system. Within the framework of this simulation, it is possible to integrate additional reaction rules at certain points of the simulation e.g. beyond a certain cycle threshold with the `#CYCLE FOR RULE IMPLEMENTATION` parameter. This parameter enables the activation of a pre-specified rule that becomes applicable exclusively at a designated cycle number. The methodology can serve as an effective strategy for down-regulating reactions, for example by facilitating the formation of metal complexes that exhibit catalytic inhibition properties. An illustrative example of this approach is the transformation of a key reactant that is central to the synthesis of the final product into a modified form such as a dimer or an alternative structural variant that is incompatible with the existing reaction pathway. When implementing such regulatory requirements, it is essential to maintain the potential for reversible reactions. The absence of this reversibility could lead to a permanent alteration of the molecules, resulting in the loss of their ability to return to their original state. This

consideration is crucial to ensure a dynamic equilibrium within the system that allows for a balanced interplay between the forward and reverse reactions. In the simulation of reaction networks, the inclusion of a new reaction always leads to an increase in the cumulative rule sets, as it extends the range of permissible reactions, even if the reaction introduced has a regulating influence on other reactions within the system. To minimize this effect, it is advisable to introduce a restriction mechanism for certain reactions at a certain point in the simulation. This regulation process can be achieved by using three different command line parameters: `#CYCLE OF LIMITATION`, which specifies the specific simulation cycle in which the limitation is to be imposed; `#LIMITATION FACTOR`, which determines the extent of the imposed limitation; and `#LIMITED REACTION`, which specifies the particular reaction that is subject to this regulating limitation.

```
python atom_mapped_reactions.py
-c          #CYCLES OF SIMULATION           (INT)
-i          #CYCLES FOR INPUT MOLECULES     (INT INT INT ...)
-r          #CYCLE FOR RULE IMPLEMENTATION  (INT)
-lim_cyc    #CYCLE OF LIMITATION            (INT)
-lim_val    #LIMITATION FACTOR              (FLOAT)
-lim_reac   #LIMITED REACTION               (STR)
```

For instance the command line for a simulation of 1000 cycles with the introduction of new molecular units at cycle 250 and 500 with the activation of one or more pre-defined rules at cycle 300 and the limitation of reaction `oro1` to 5% at cycle 70 would look like this:

```
python atom_mapped_reactions.py
-c          1000
-i          250 260 270 500 510
-r          300
-lim_cyc    70
-lim_val    0.05
-lim_reac   'oro1'
```

However, it is not mandatory to use each of these command line options; their inclusion in this case is primarily to illustrate the range of functionalities available. After starting the simulation, the program will print the information described in Section 3.2. At the end of the simulation, two crucial output files for the post processing analysis will be generated:

- `mapped_molecules.txt`
- `trajectory.json`

The file `mapped_molecules.txt` contains the systems' initial mapped molecules as `SMILES`. This file can be further processed to simulate the system constellation for a certain cycle number. The `trajectory.json` file contains information about:

```
{
"cycle_count": 2,
"mol_counter": {
  "[H][C]#[N]": 74,
  "[H+]": 1,
  "[C-]#[N]": 1
},
"rule_rate_distribution": {
  "oro1": 74,
  "oro2": 74,
  "oro1_rv": 1
},
"rule_rate_sum": 149,
"chosen_reaction": "oro1",
"sampling_time": 0.0001209451319468867,
"reaction_time": 0.03131619241928712,
"educts": [
  "[N:22]#[C:23][H:24]"
],
"products": [
  "[H+:24]",
  "[N:22]#[C-:23]"
]
},
```


After obtaining the data from the `json` file, the information contained can be carefully processed to facilitate the reconstruction of all trajectories. This is possible because the data structure comprehensively contains all relevant transformations with atom IDs, time steps and rule rates. It is crucial that the analysis phase always follows the simulation, so that several analyses can be carried out efficiently and independently of the simulation process.

5 Analysis

The previously described output files are subjected to further processing and analytical examination. The analytical procedure is divided into a processing section, in which relevant data is carefully filtered out of the `trajectory.json` file depending on the analysis objectives, and a visualisation section, in which the extracted data is converted into graphical representations for better interpretation.

5.1 Processing

For the processing step the `trajectory_analysis.py` file is used to extract data from the simulation output files. Therefore following command line options are necessary for the data extraction:

```
### Required Command line options ###
--trajectory_file: JSON file directory containing events' info

--pool_file: TXT file with initial molecule from systems' pool

### Not Required Command line options ###
--target_cycle: type=int, default= 0, help="target cycle as integer"

--target_time: type=int, default= 0, help="target time as integer"

--pattern_smarts_string: type=str, default= "",
help="pattern smarts string for a subgraph to find a subgraph or
multiple subgraphs over a periode of time as SMARTS"

--smiles_string: type=str, default= "",
help="reactant smiles string (isomeric e.g. PubChem or All Hs Explicit)
to find a mol in the mixture"
```

An example command for searching adenine molecules over 650 cycles from the simulation could be formulated as follows, emphasising that the command is a single command line:

```
python trajectory_analysis.py
--trajectory_file trajectory.json
--pool_file mapped_molecules.txt
--smiles_string '[n]1[c]([H])[n]([H])[c]2[n][c]([H])[n][c]([N]([H])[H])[c]12'
--target_cycle 650
```

After the data processing is completed several files will appear in your directory, which are used for graphical representation:

- `updated_molecule_pool.txt`
- `molecule_search.json`
- `subgraph_search.json`

The file `updated_molecule_pool.txt` contains the molecular units of the system intended for the target cycle. The file `molecule_search.json`, on the other hand, archives data on the specific molecular structure in `SMILES` notation to be included in the resulting mixture. This file also provides information about the synthetic derivation of the mapped molecule and describes the previous transformations that led to the formation of the molecule under investigation.
If you work with the above command, the file `subgraph_search.json` remains without content because the command line argument `-pattern_smarts_string` is missing. Entering a `SMARTS` string in this parameter will cause the `subgraph_search.json` file to reflect the information structure of the `molecule_search.json` file. In cases where neither the `SMILES` representation of the molecule nor the `SMARTS` pattern is integrated into the reaction frame, the resulting files remain empty after processing. The next step is the visualisation of the generated data.

5.2 Visualisation

The visualisation of the reaction paths is done by generating DOT files. The required data for these paths are obtained from the files `subgraph_search.json` or `molecule_search.json`, with the current discourse focusing on the latter. For the visualisation the script `reaction_path_search.py` is executed in a Python environment, creating a directory named `single_pathways`. Within this directory, single synthesis pathways are described, consisting of nodes corresponding to different reactions and edges representing the cycle number in which a particular reaction occurs for a particular product molecule (Fig. 5.1). In this way, the time course of a single reaction motif can be tracked across different cycle numbers. Both the resulting PDF and DOT files are saved in the `single_pathways` directory. In addition, a separate directory called `aggregated_single_pathways` is created. This directory also contains the synthesis pathways of the molecules mentioned above. However, it differs in that it uses a cumulative approach to represent the edges. Here, the frequency of occurrence of a reaction is visually coded by the increased thickness of the edges, allowing a comparative quantification of reaction prevalence within the aggregated data (Fig. 5.2).
In order to determine potential synthesis pathways for a particular molecular species, the reaction pathways of all individual molecules of a particular species are aggregated. This aggregation results in the creation of files labelled `aggregated_reactions.dot` and `aggregated_reactions.pdf`. This figure includes the confluence of synthetic pathways derived from individual molecular units. This visual representation helps to recognise

reaction clusters and recurring synthetic sequences in the data. It also provides a perspective on the underlying synthetic motifs and the possible mechanistic pathways of the molecular species involved. (Fig. 5.3). In order to reduce the influence of molecules resulting from reactions, which are not contributing to the backbone of the target molecule in the first place, e.g. small molecules and small ions, these molecular entities were not involved into the DOT-files to reduce the dependencies and simplify the visualisation. However, there are reaction networks, which could be interested in these molecules and ions so this need to be changed in the `trajectory_analysis.py` file in the `relevant molecules` section.

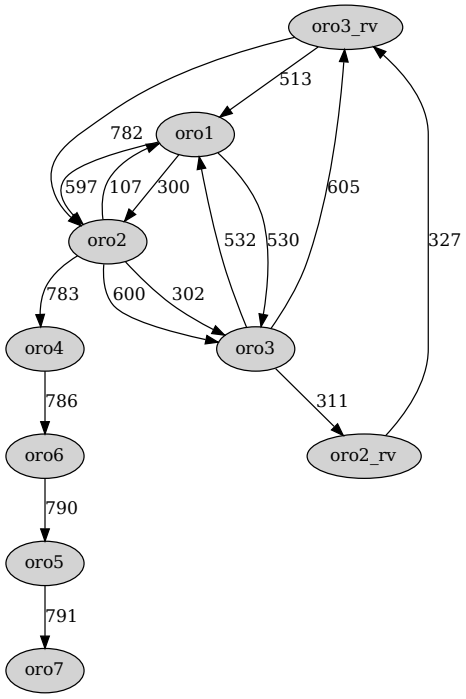

Figure 5.1: The synthesis pathway for a single molecular species is represented by a directed graph in which nodes represent different reaction events and directed edges indicate progression through successive cycle numbers.

In order to observe the distribution of the rule rates during the simulation, the file `simulation_plots.pdf` is created. This file contains plots in which the total rule rates are plotted against the respective cycle numbers, including all individual reaction rule rates that occur during the simulation. These plots specifically represent the values of the reaction rule rates, deliberately excluding cases where these rates are zero. A notable aspect of this visualization lies in its ability to demonstrate the impact on the rule rates of specific reactions as a result of the introduction of molecules and rules at different stages of the simulation described in Section 4.4 `atom_mapped_reactions.py`. Such introduction of molecules and rules serves as a potential regulatory mechanism that

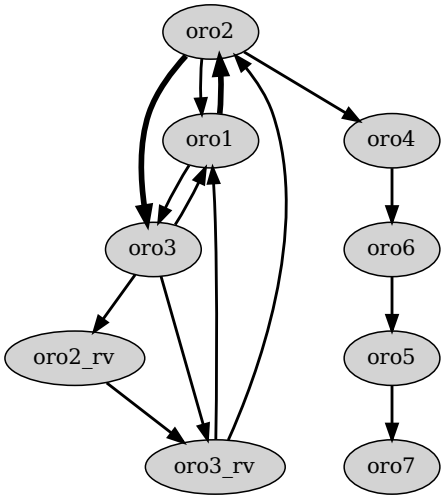

Figure 5.2: The synthesis pathway for a single molecular species is depicted by a cumulative approach to represent the edges. The frequency of occurrence of a reaction is visually coded by the increased thickness of the edges, allowing a comparative quantification of reaction prevalence within the aggregated data

facilitates either up- or down-regulation of reaction rates. The impact of these molecule and rule introductions is further discussed in Section 5.3 Results.

5.3 Results

The investigation of the dynamic reaction of a system is performed by computational simulation under a variety of initial parameters. These include the variation of initial molecule numbers, the systematic adjustment of estimated reaction constants within appropriate ranges and the inclusion of modulating entities such as inhibitors or accelerators (`input_molecules, input_reactions`) to explore their effects on the kinetics of the system. One system is subjected to this computational analysis in order to investigate the behavior under these modified conditions:

· The Oro-Orgel Adenine-Synthesis

Basic simulations are carried out without input molecules or input reactions, using theoretically derived reaction constants that define the probabilistic development of the system. Since reaction constants are basically invariant - or at least cannot be changed arbitrarily - the investigation of system modulation by changing these constants is not methodologically meaningful.

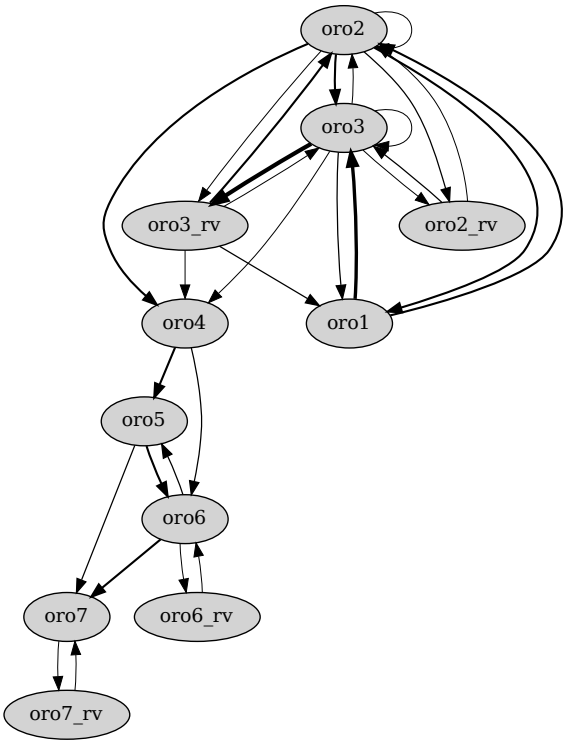

Figure 5.3: The synthesis pathway for all molecules of interest from the simulation depicted as a cumulative approach to represent the edges. The frequency of occurrence of a reaction is visually coded by the increased thickness of the edges, allowing a comparative quantification of reaction prevalence within the aggregated data

5.3.1 Standard Simulation and Simulation with Molecule Introduction

A more informative method for studying the modulation of system kinetics involves the strategic introduction of critical molecular species and regulative reactions that can significantly influence the reaction pathways and thus the overall system behavior.
The first reaction network investigated is the Oro-Orgel adenine synthesis, a process characterized by the formation of adenine through the interaction of five hydrogen cyanide (HCN) molecules. For this particular synthesis, the simulation is configured to run over a period of 1500 - 2000 cycles. Initially, the simulation is run without external molecule or rule introductions. This approach is chosen to allow a more comprehensive understanding of the system's intrinsic behavior under the predefined conditions determined by the established rules and the molecular entities involved. Following the simulation, the aggregation of the reactions and the distribution of the rule rates are visualized. This visual representation (see Fig. 5.4 & 5.7) provides an insight into the development of the system as determined by the simulated data. In addition, an examination of the DOT file, which details the reaction sequence, shows the dependencies between the different

reactions within the network. This analysis is crucial for elaborating the dynamics and sequential dependencies inherent in the Oro-Orgel adenine synthesis process.

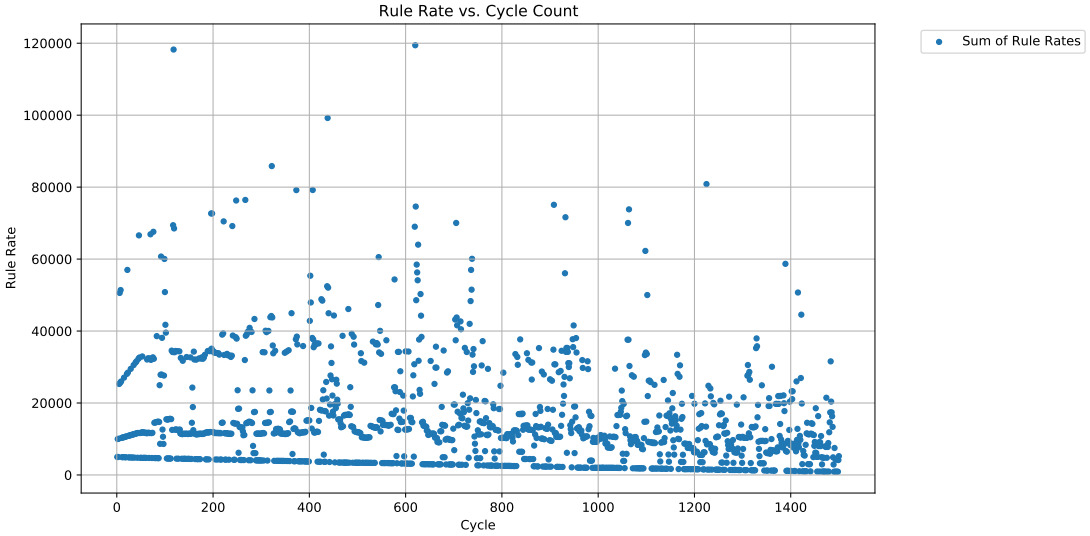

Figure 5.4: In the analysis of the Oro-Orgel adenine synthesis, which is characterized by a predefined set of rule rate constants and an initial mixture with 500 HCN molecules, the sum of the rule rates is observed in the `simulation_plots.pdf` file. However, it is noted that the aggregate of these rule rates does not significantly reveal the dependencies between the reactions under these specific conditions.

This scenario illustrates a baseline simulation of Oro-Orgel adenine synthesis performed without external molecular inputs, rule implementations or constraints on the existing rules. In subsequent iterations of the simulation, these elements are gradually integrated to evaluate their impact on the system behavior. The initial modification involves the systematic introduction of molecules in predetermined cycles. Specifically, molecular inlets are planned for cycles 500 to 525, 750 to 775 and 1250 to 1275, with the inlet frequency set to every 5 cycles. The molecular inflow per inlet consists of 5 molecules of the HCN tetramer associated with Oro-Orgel synthesis and 2 hydrogen ions. This step-wise approach aims to highlight the effects of controlled molecule introduction on the dynamic course of the synthesis simulation.
The introduction of molecules into the system can be seen in the graphical representation of the cumulative rule rates (Fig. 5.6). An upward trending trajectory in the diagram shows the effect of the introduced molecules on the dynamics of the system. The effect on certain reactions varies; some show a direct response, especially those with direct pattern matches with the introduced molecules (Fig. 5.7; oro4). In contrast, others show an indirect, time-delayed influence that manifests itself in a subsequent increase in their rule rates (Fig. 5.7; oro7). For example, the rule rate of the reaction labeled `oro4` shows an immediate increase at the same time as the molecular inlet. In turn, the rule rate of the reaction `oro7` shows a delay in its increase due to its involvement in molecular patterns corresponding to the products generated from the `oro6` reaction. This pattern

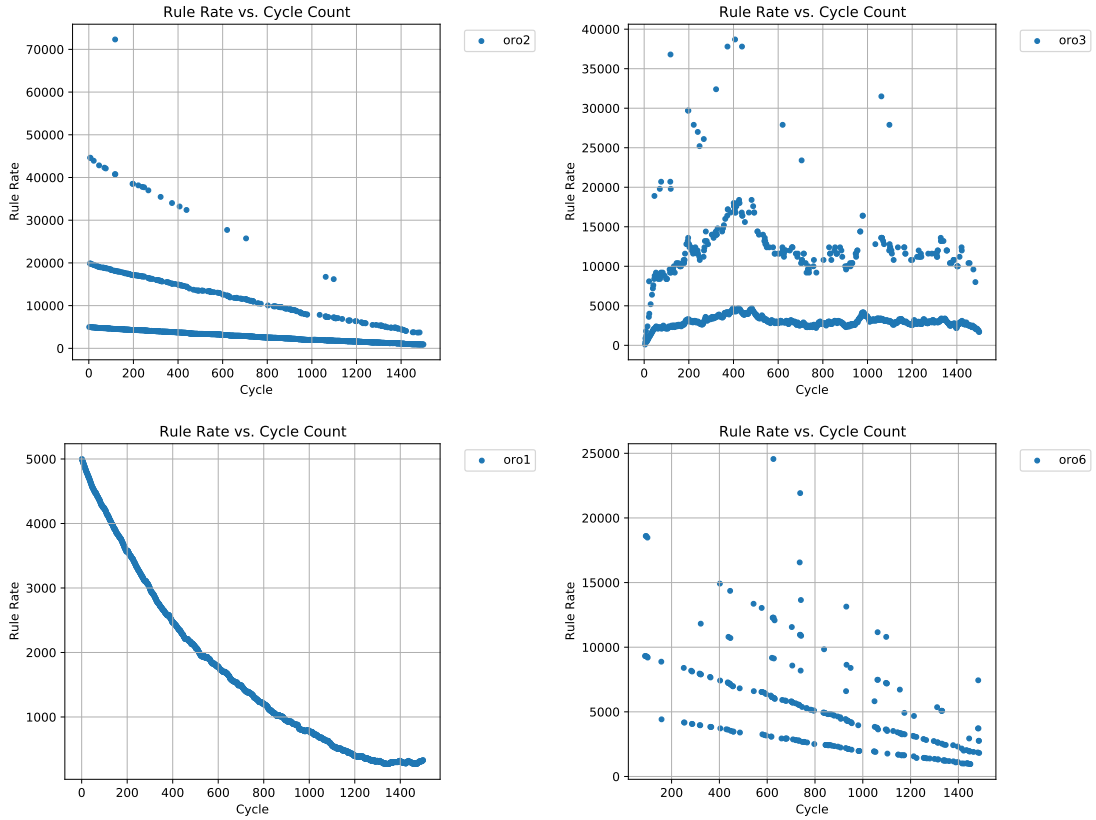

Figure 5.5: In the analysis of the Oro-Orgel adenine synthesis, which is characterized by a predefined set of rule rate constants and an initial mixture with 500 HCN molecules, the rule rate distribution for selected reactions is observed in the `simulation_plots.pdf` file. This analysis allows monitoring the evolution of the rule rates throughout the simulation. It also provides a better insight into the courses and dependencies between the individual reactions within the system.

illustrates the interplay between direct and indirect reaction pathway influences within the system, which is emphasized by the timing and nature of the molecular interactions. In reactions where the rate of rules increases following the introduction of molecules and does not decrease subsequently, it can be assumed that the reaction is unlikely to occur. Furthermore, the molecules involved in this reaction do not undergo transformations that would alter the pattern of the reaction (see Fig. 5.7; oro7_rv).

5.3.2 Simulation with Input Reactions and Limitation of Reactions

To explore the dynamics and interrelationships within a chemical system, an effective approach is to strategically implement new reactions while modulating existing ones. However, it must be recognised that in practical chemistry, initiating a reaction at a specific time or directly limiting a reaction is not feasible. Instead, this concept is usually realised through the introduction of regulating agents such as metal ions, which form

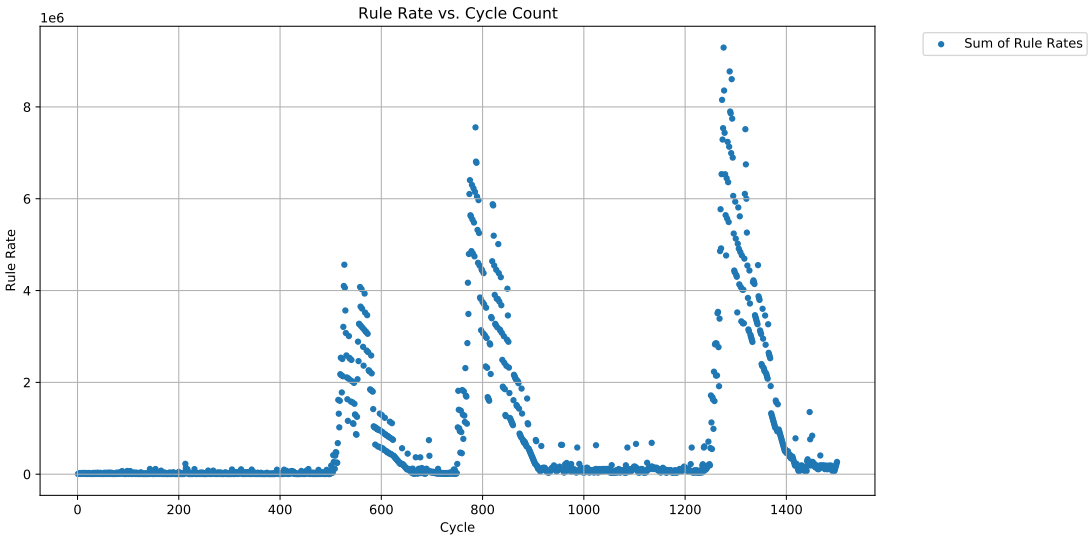

Figure 5.6: In the analysis of the Oro-Orgel adenine synthesis, which is characterized by a predefined set of rule rate constants and an initial mixture with 500 HCN molecules, the sum of the rule rates is observed in the `simulation_plots.pdf` file with introduction of molecules within the simulation. The contribution of the input molecules is seen as peaks in the plot of rule rate sum

complexes with certain molecules within the system and thereby influence the reaction pathways. While the mere introduction of new reactions expands the potential reaction network, it does not necessarily reduce the total amount of regulation sets, as it increases the total number of possible reactions in a stochastic simulation approach. Consequently, it is crucial to simultaneously modulate existing reactions, especially those affected by the newly introduced regulatory pathways. This dual strategy of introducing new reactions and modulating existing ones allows for more nuanced and effective control over the chemical system and facilitates a deeper understanding of its underlying dynamics and interactions.

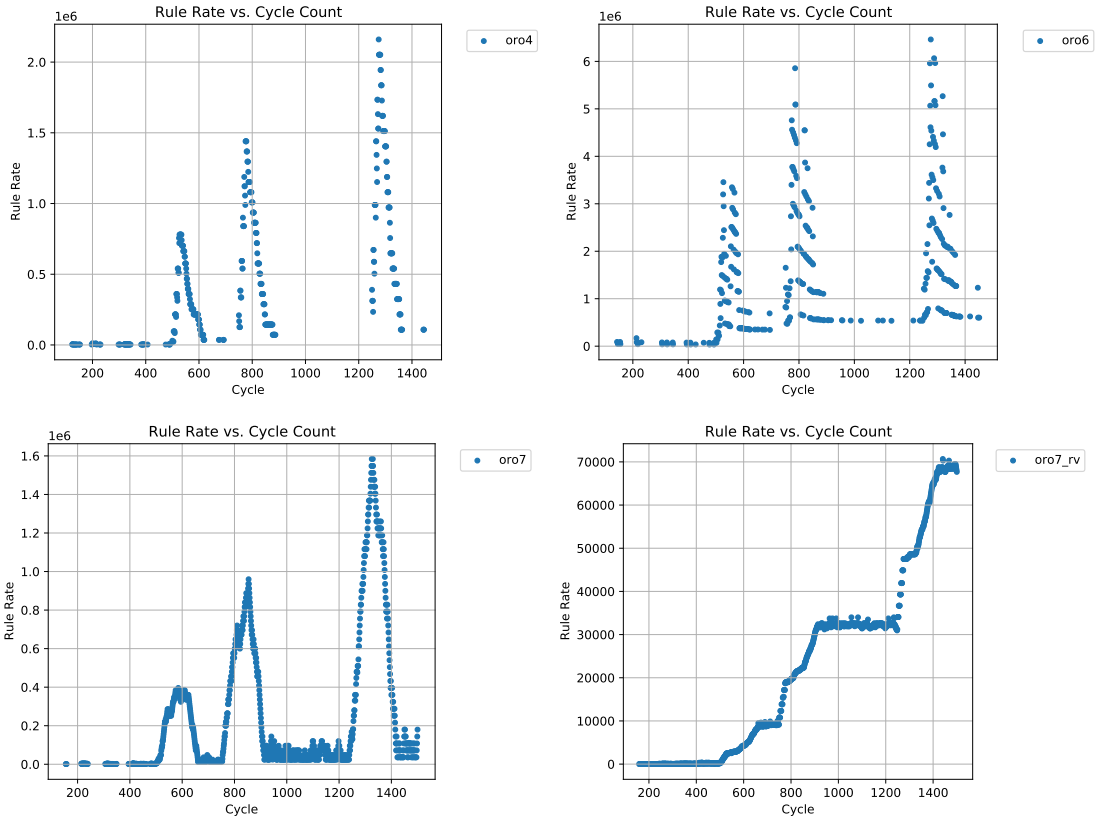

Figure 5.7: In the analysis of the Oro-Orgel adenine synthesis, which is characterized by a predefined set of rule rate constants and an initial mixture with 500 HCN molecules, the rule rate distribution for selected reactions with introduction of molecules is observed in the `simulation_plots.pdf` file. This analysis allows monitoring the evolution of the rule rates throughout the simulation. It also provides a better insight into the courses and dependencies between the individual reactions within the system and the influence on the input molecules.

The methodology of inhibition of a specific reaction with simultaneous introduction of an alternative masking reaction is shown in Figure 5.8 and Figure 5.9. The parameters controlling the introduction of molecules in this scenario are the same as those previously described for Figure 5.7 and Figure 5.6. A comparative analysis of the rule rates in these scenarios provides valuable insight into the effects of the masking response and the imposed constraints on the dynamics and stochastic distribution of the system. This comparison is important to understand the changes in the behaviour of the system as a result of these strategic interventions compared to the dynamics observed in the original simulation.

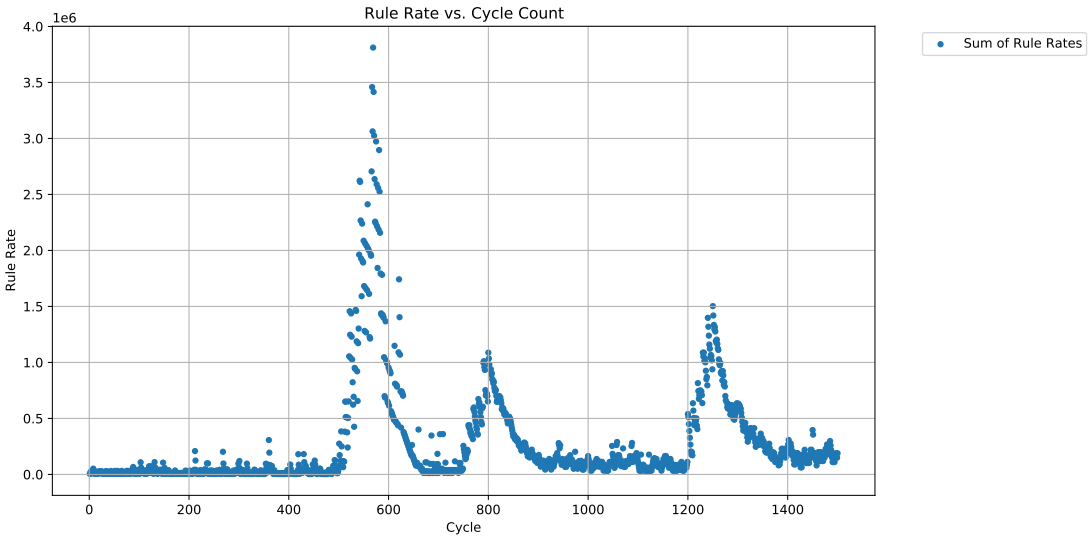

Figure 5.8: In the analysis of the Oro-Orgel adenine synthesis, which is characterized by a predefined set of rule rate constants and an initial mixture with 500 HCN molecules, the sum of the rule rates is observed in the `simulation_plots.pdf` file with introduction of molecules within the simulation and limitation of reaction `oro6` at cycle 750. The contribution of the input molecules and the limitation of this certain reaction is seen as peaks in the plot of rule rate sum

5.3.3 Conclusion

The software's capabilities allow the user to simulate finite chemical or biochemical systems, incorporating features such as the introduction of molecules, the restriction of certain reactions or the establishment of new reaction rules. This functionality facilitates the theoretical investigation of system behaviour and reduces the need for only experimental procedures and the subsequent generation of empirical data. However, a major challenge is the determination of suitable rule rate constants for the desired simulation results. In this study, the calibration of these constants was performed manually, which required numerous iterations to increase the yield of synthesised adenine molecules. Despite these efforts, it remains uncertain whether the determined conditions are optimal for adenine synthesis.
To address this limitation, the development of an iterative simulation methodology within the software is proposed. This advanced approach would systematically adjust the rule rate constants to optimise the production of adenine, potentially circumventing the need for extensive manual calibration. The details of this proposed methodology and its potential impact will be discussed in more detail in the Section 5.4.

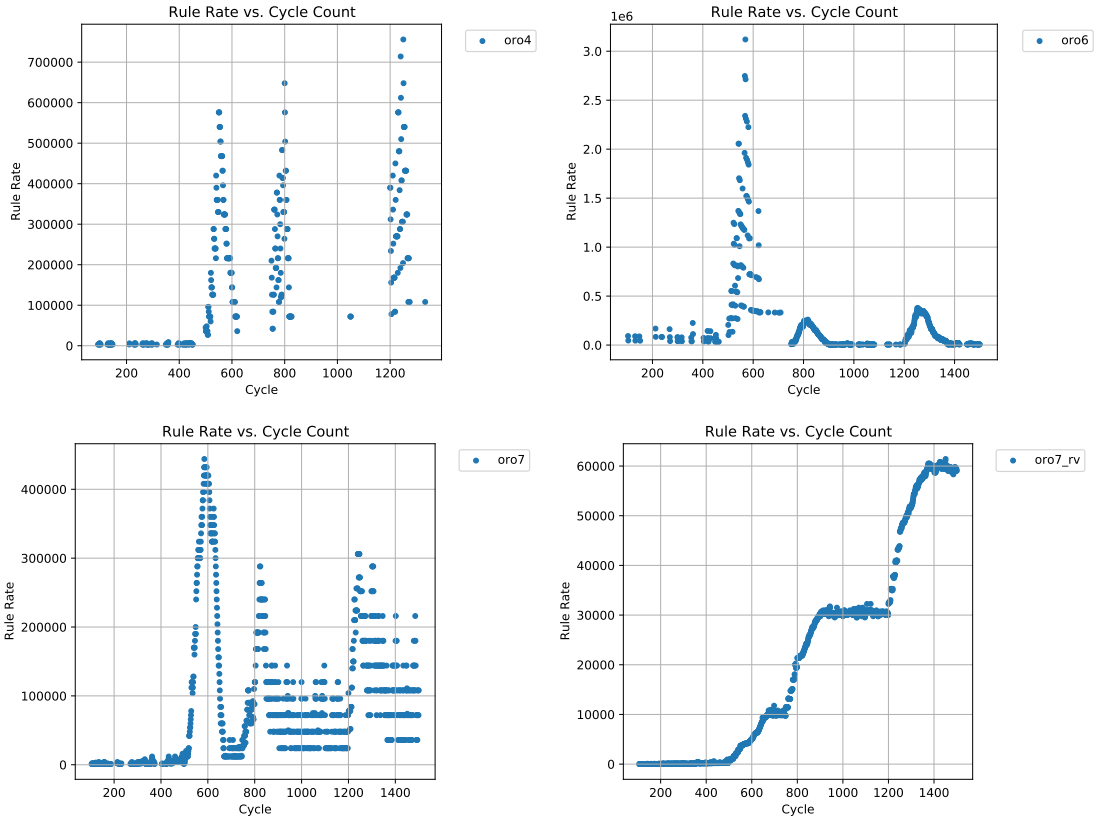

Figure 5.9: In the analysis of the Oro-Orgel adenine synthesis, which is characterized by a predefined set of rule rate constants and an initial mixture with 500 HCN molecules, the rule rate distribution for selected reactions with introduction of molecules is observed in the `simulation_plots.pdf` file. This analysis allows monitoring the evolution of the rule rates throughout the simulation. It also provides a better insight into the courses and dependencies between the individual reactions within the system and the influence on the input molecules. Here the limitation of reaction `oro6` can be seen in the corresponding diagram.

5.4 Outlook

Optimisation of rule rate constants is a crucial aspect in the simulation of chemical systems, especially given the complexity of pattern-based rules that can vary between different molecules. Traditional reaction constants from chemical reactions may not be directly applicable due to the variability in pattern-based reactions. One approach that may prove beneficial is the investigation of ratios between certain reaction constants as a basis for estimating rule rate constants. However, the integration of energetic considerations into the rule rate constants within a stochastic simulation algorithm poses a major challenge. To overcome these challenges, the implementation of an iterative process to modify the constants is proposed. This process aims to systematically adjust the constants to achieve a desired change in the concentration of specific molecular entities within the simulation

interval. Initially, the simulation would start with a randomly selected set of rule rate constants. Subsequent iterations would involve adjustments to these constants, with each iteration evaluating the impact of these changes on the simulation outcome. This iterative process would continue until further changes no longer have a significant impact on the simulation results. This methodology provides a systematic approach to fine-tuning the simulation parameters, improving the accuracy and relevance of the simulation results.

6 Discussion

Network-free simulation has emerged as a key tool in the field of dynamic systems biology over the last two decades, proving its lasting utility and relevance. The main objective here was to provide a basis for the development of a kinetic Monte Carlo simulation engine specifically designed for rule-based models. This work goes beyond a superficial understanding of the generalisation of Gillespie's direct method for rule-based models by addressing the creation of a basic framework and explaining several intricate implementation details that are essential for the construction of a network-free simulation algorithm. While this approach is applicable to a wide range of scenarios, its explicit implementation is not universally applicable and further development is required to capture the more complex cases. The method, called network-free simulation, is a comprehensive adaptation of Gillespie's direct method, with implications that go beyond its original application in modelling stochastic variations in biochemical systems with finite populations. Rule-based modelling combined with network-free simulation and atom tracking facilitates the definition of reaction classes based on minimal interaction data. This approach allows the exclusion of molecular interactions that are either irrelevant or not yet known to significantly influence reaction rates for specific molecular entities. Consequently, it eliminates the need to enumerate all possible species and reactions within a given interaction network, a task that is often impractical or computationally prohibitive. Explicitly recognising combinatorial complexity within this framework allows for a more nuanced and accurate investigation of system dynamics and avoids oversimplifications in the model. This aspect is particularly important when studying systems with complex metabolic reaction networks. Research using network-free simulation methods focuses primarily on the modelling and simulation of complicated biochemical systems, which are characterised by considerable combinatorial complexity. Network-free simulation algorithms provide a comprehensive and effective framework for the accurate simulation of such complex systems. The widespread adoption of software packages with network-free simulation capabilities has not only brought to light software bugs and inefficiencies, but has also fuelled the demand for new functionalities. It is important to recognise that the field of network-free simulation algorithm development is still in its early stages compared to the more mature field of stochastic simulation algorithms (SSA). Ongoing efforts are aimed at improving the performance of network-free simulations, indicating that significant progress still needs to be made. In parallel with the development of traditional SSA based on Gillespie's direct method, it is expected that the increasing application of rule-based modelling and network-free simulation will lead to advances in simulation algorithms and promote the development of robust, efficient software solutions.

Bibliography

1. Gillespie, 2006: Stochastic Simulation of Chemical Kinetics
2. Deuflhard, Röblitz, 2015: ODE models for systems biological networks. A guide to numerical modelling in systems biology
3. Elowitz et al., 2002: Stochastic gene expression in a single cell
4. Gillespie et al., 1977: Exact stochastic simulation of coupled chemical reactions
5. Cox, Miller, 1965: The theory of stochastic processes
6. Bortz et al., 1975: A new algorithm for Monte Carlo simulation of Ising spin systems
7. Gillespie, 1976: A general method for numerically simulating the stochastic time evolution of coupled chemical reactions
8. Gibson, Bruck, 2000: Efficient exact stochastic simulation of chemical systems with many species and many channels
9. Ramaswamy, Sbalzarini, 2010: A partial-propensity variant of the composition-rejection stochastic simulation algorithm for chemical reaction networks
10. Chylek et al., 2014: Rule-based modeling: a computational approach for studying biomolecular site dynamics in cell signaling systems
11. Blinov et al., 2006: A network model of early events in epidermal growth factor receptor signaling that accounts for combinatorial complexity
12. Yang et al., 2008: Kinetic Monte Carlo method for rule-based modeling of biochemical networks
13. Sneddon et al., 2011: Efficient modeling, simulation and coarse-graining of biological complexity with NFsim
14. Zhang et al., 2005: Implementation of a discrete event simulator for biological self-assembly systems
15. Suderman et al., 2018: Generalizing Gillespie's Direct Method to Enable Network-Free Simulations

16. Su et al., 2016: Phase separation of signaling molecules promotes T cell receptor signal transduction

17. Aitken et al., 2013: A rule-based kinetic model of RNA polymerase II C-terminal domain phosphorylation

18. Köhler et al., 2014: A rule-based model of base excision repair

19. Kühn, Hillmann, 2016: Rule-based modeling of labor market dynamics: an introduction

20. Suderman, Deeds, 2013: Machines vs. ensembles: effective MAPK signaling through heterogeneous sets of protein complexes

7 Appendix

Relevant Links for digital-only resources:

1. SMILES: http://opensmiles.org/opensmiles.html
2. SMARTS: https://www.daylight.com/dayhtml/doc/theory/theory.smarts.html
3. RDKit Cookbook: https://www.rdkit.org/docs/Cookbook.html
4. RDKit: https://www.rdkit.org/docs/Install.html
5. RDChiral: https://github.com/connorcoley/rdchiral
6. GitHub Repository: https://github.com/connorcoley/rdchiral