University of New Mexico
UNM Digital Repository

Computer Science ETDs | Engineering ETDs

12-1-2012

# Multivalent Random Walkers: A computational model of superdiffusive transport at the nanoscale

Mark Joseph Olah

Follow this and additional works at: https://digitalrepository.unm.edu/cs_etds

Recommended Citation
Olah, Mark Joseph. "Multivalent Random Walkers: A computational model of superdiffusive transport at the nanoscale." (2012).
https://digitalrepository.unm.edu/cs_etds/24

This Dissertation is brought to you for free and open access by the Engineering ETDs at UNM Digital Repository. It has been accepted for inclusion in Computer Science ETDs by an authorized administrator of UNM Digital Repository. For more information, please contact disc@unm.edu.

www.manaraa.com

Mark J. Olah

*Candidate*

Computer Science

*Department*

This dissertation is approved, and it is acceptable in quality and form for publication:

*Approved by the Dissertation Committee:*

Darko Stefanovic , Chairperson

Cris Moore

Lance Williams

Milan Stojanovic



www.manaraa.com

# Multivalent Random Walkers:
# A computational model of superdiffusive transport at the nanoscale

by

**Mark J. Olah**

B.S., Computer Science, Carnegie Mellon University, 2004
B.S., Mathematics, Carnegie Mellon University, 2004

DISSERTATION

Submitted in Partial Fulfillment of the
Requirements for the Degree of

Doctor of Philosophy
Computer Science

The University of New Mexico

Albuquerque, New Mexico

December, 2012



www.manaraa.com

# Acknowledgments

I would like to acknowledge the help and support I have received from my committee members over the years. Lance Williams originally gave me the idea to model the motion of walkers at a mechanical equilibrium using a Boltzmann distribution. Cris Moore's inspiring lectures on the use of randomness as a tool for computation have led me towards my current approach in this work. I would also like to acknowledge Milan Stojanovic for his visionary approach to chemistry, and insight into the nature of DNA nanotechnology. Without his guidance I would never have arrived at my present understanding of the complexity of behaviors possible in seemingly simple chemical systems. Finally, I'm grateful to have had Darko Stefanovic as an advisor since the first day I arrived at UNM. Darko has created a research environment where education and science are always the most important considerations, and he has helped me to develop from a student into a scientist. None of this work would have been possible without Darko's help and the help of other students within our Molecular Computing Group. In particular, I would like to thank my co-authors Oleg Semenov and David Mohr, who have been instrumental in much of the work which supports this dissertation. I would also like to thank Tom Hayes for sharing his insight into the Metropolis-Hastings algorithm and other Markov chain Monte Carlo techniques. This work was generously supported by the National Science Foundation under grants 0533065, 0829896, and 1028238.



www.manaraa.com

# Multivalent Random Walkers: A computational model of superdiffusive transport at the nanoscale

by

**Mark J. Olah**

B.S., Computer Science, Carnegie Mellon University, 2004
B.S., Mathematics, Carnegie Mellon University, 2004

Ph. D., Computer Science, University of New Mexico, 2012

## Abstract

We present a stochastic model and numerical simulation framework for a synthetic nanoscale walker that can be used to transport materials and information at superdiffusive rates in artificial molecular systems. Our *multivalent random walker* model describes the motion of a walker with a rigid, inert body and flexible, enzymatic legs. A leg can bind to and irreversibly modify surface-bound chemical substrate sites arranged as nanoscale tracks. As the legs attach to, modify, and detach from the sites, the walker moves along these tracks. Walkers are symmetrical and the tracks they walk on are unoriented, yet we show that under appropriate kinetic constraints the walkers can transform the chemical free energy in the surface sites into directional motion, and can do ordered work against an external load force. This shows that multivalent random walkers are a new type of molecular motor, useful for directional transport in nanoscale systems.

We model the motion of multivalent random walkers as a continuous-time discrete-state Markov process. States in the process correspond to the chemical state of the legs



www.manaraa.com

and surface sites, and transitions represent discrete chemical changes of legs binding to, unbinding from, and modifying the surface sites. The Markov property holds because we let the mechanical motion of the body and unattached legs come to equilibrium in between successive chemical steps, thus the transitions depend only on the current chemical state of the surface sites and attached legs. This coarse-grained model of walker motion allows us to use both equilibrium and non-equilibrium Markov chain Monte Carlo simulation techniques. The Metropolis-Hastings algorithm approximates the motion of a walker's body and legs at a mechanical equilibrium, while the kinetic Monte Carlo algorithm simulates the transient chemical dynamics of the walker stepping across the surface sites. Using these numerical techniques, we find that MVRWs move superdiffusively in the direction of unmodified substrate sites when there is a residence time bias between modified and unmodified sites. This superdiffusive motion persists when opposed by external load forces, showing that multivalent random walkers are *molecular motors* that can transform chemical free energy into ordered mechanical work.

To produce these results we devised a distributed object-oriented framework for parallel simulation and analysis of the MVRW model. We use an object-relational mapping to persistently maintain all simulation-related objects as tuples in a relational database. We present a new object-relational mapping technique called the *natural entity framework* which disambiguates the semantics of object identity and uniqueness in the relational and object-oriented programming models. Using the natural entity framework we are able to guarantee the uniqueness of mappings between data stored as objects in the relational database and external data stored in non-transactionally-secured HDF5 data files.



www.manaraa.com

# Contents





www.manaraa.com

www.manaraa.com

www.manaraa.com

www.manaraa.com

www.manaraa.com

www.manaraa.com

# List of Figures





www.manaraa.com

www.manaraa.com

# List of Tables





www.manaraa.com

# Chapter 1
# Introduction

Nature at the nanoscale is different from the familiar macroscopic experience in many ways, the most fundamental of which is the stochastic character of motion and events. At this scale, objects have such tiny mass that continual bombardment by other molecules effectively randomizes momentum, leading to slow, uncontrolled diffusive motion. Without a source of energy, any system will eventually come to thermodynamic equilibrium, after which it loses all capacity to process material or information in useful ways. Motion does not cease at equilibrium, only net motion does. The probability of any action is exactly balanced by its opposite, a property called *detailed balance*.

For living systems, equilibrium is death. Cells are the most sophisticated molecular machines known to science, and they need to transport materials and information in directed, purposeful, and prescriptive ways requiring the expenditure of energy. A *molecular motor* is a nanoscale device that can transform chemical free energy into directed motion and mechanical work—it produces order from disorder. Cells use molecular motors to control internal structure [50] and regulate internal distribution of materials and information [62]. A class of natural cellular motors called *translational molecular motors* move directionally on oriented 1D tracks [122]. These natural molecular motors have evolved to be incredibly efficient, making them highly specialized for their particular cellular environment. Adapting these motors to move over arbitrary tracks and to use arbitrary chemical substrates as fuel without fundamentally altering their functionality or efficiency is not feasible, as natural molecular motors rely on complex, non-local kinetic coupling between their walking heads to coordinate their rigid hand-over-hand walking gait [91, 120].

In this work we investigate simpler mechanisms by which a translational molecular motor can be constructed. We show it is possible to generate directed motion and mechanical work from the random thermal noise without the sophisticated and specialized conformational coupling employed by natural motors. We introduce the *multivalent ran-*



www.manaraa.com

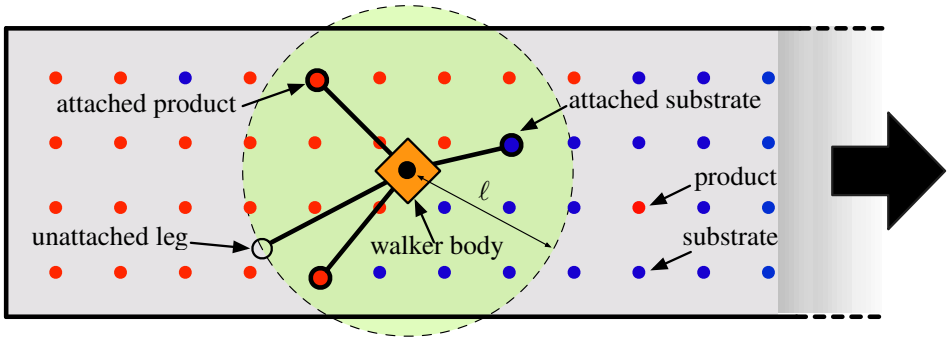


**Figure 1.1:** A multivalent random walker (MVRW) has a rigid body and $k$ flexible legs of length $\ell$. A leg can attach to and detach from fixed chemical sites on a surface that are within distance $\ell$ from the body. The enzymatic action of a leg irreversibly transforms a substrate site into a product, changing the subsequent binding kinetics for that site. As the legs attach and detach from sites, the walker moves over the surface.

*dom walker* (MVRW) model, which describes the motion of walkers with a rigid, inert body and several flexible enzymatic legs (Figure 1.1). The legs walk over a surface of immobile chemical binding sites, modifying the surface as they move. Unlike cellular molecular motors, MVRWs do not rely on oriented tracks, rigid walking gaits, chemo-mechanical coupling, or coordinated conformational changes. The legs are chemically and conformationally uncoupled, other than the passive constraint imposed by the connection to a common body. Yet, under appropriate kinetic conditions, these walkers can be made to move directionally and processively, even in opposition to a force. By modeling and understanding these simple walker systems, we learn which chemical and mechanical properties of walker-based motors are sufficient for superdiffusive motion, and which properties are not necessary.

## 1.1 Molecular spiders

The MVRW model is inspired by the chemical and mechanical features of a type of synthetic DNA-based molecular walker, called a *molecular spider* [98]. A molecular spider



www.manaraa.com

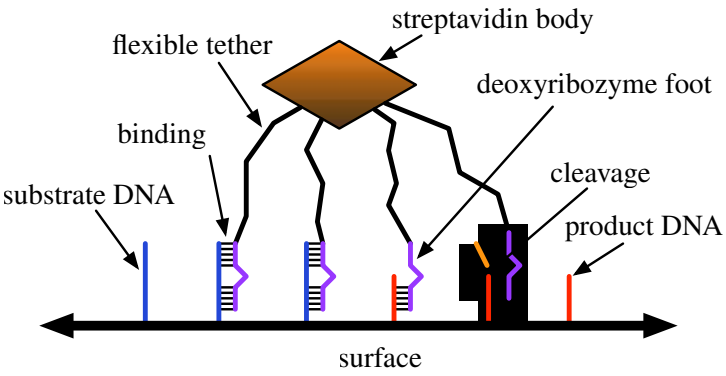


**Figure 1.2:** A molecular spider moves over a surface of DNA substrates by binding and enzymatically cleaving them, leaving behind shorter product strands. Because the lower part of the leg is complementary to the cleaved product sites, it can still bind and unbind those sites but at a different rate.

has a small rigid body with several binding sites for flexible, enzymatic legs (Figure 1.2). The legs are deoxyribozyme phosphodiesterases—enzymatic sequences of single-stranded DNA that can bind to and cleave complementary single-stranded DNA sequences. A molecular spider moves over a surface coated with complementary DNA substrates, cleaving them and leaving behind shorter product strands (Figure 1.3). As a spider moves, it leaves behind a path of cleaved sites that have been irreversibly modified. Subsequently, legs can reversibly bind to product sites, but at different rates than for substrate sites. Thus, the cleaved sites a spider leaves behind affect its future actions and the actions of other spiders that encounter these tracks. The full details of the chemical design and properties of molecular spiders and their environments are discussed in detail in Section 2.3.

Molecular spiders allow prescriptive (i.e., programmable) control over molecular motion in nanoscale systems. In recent experiments researchers observed individual molecular spiders following a self-assembled track of DNA substrates from a start site to a finish site [82]. The track in these experiments was algorithmically designed using a DNA self-assembly procedure called DNA origami [109]. Using these methods, a mathematical description of a track can be compiled to a set of DNA sequences that will spontaneously self-assemble into a chemical system that allows controlled, directed transport of information-carrying molecular cargo in an environment that is otherwise dominated by uncontrolled Brownian motion. This algorithmic control over the motion of individual molecules is a powerful tool that can be used for nanoscale assembly and computation,



www.manaraa.com

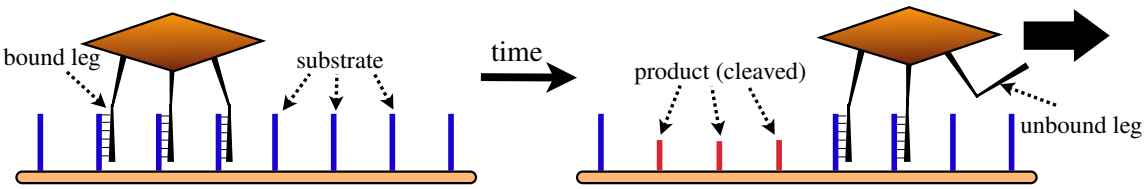


**Figure 1.3:** A molecular spider moves over a surface of fixed chemical substrate sites as the legs bind to, modify, and unbind from the sites.

and inspires our approach to the simulation and analysis of spider-like walker systems.

## 1.2 Molecular motors and nanoscale transport

Potential nanoscale applications for this type of algorithmic control of molecular motion are emerging from several directions as the design of synthetic walkers and manipulation of natural walkers become more sophisticated [2]. Applications are often inspired by observing how cellular systems utilize natural protein-based translational molecular motors such as kinesin, dynein, and myosin [72, 122] for fundamentally important cellular functions. Just as kinesin and other cellular motors are used to transport signaling molecules in neurons [50,63], synthetic molecular motors can act as molecular cargo shuttles [5, 32], cooperatively distributing materials over nanoscale transport networks [21]. If the cargo deliveries are information-carrying molecules, these networks can be thought of as controlled molecular communication systems [34], or even as molecular computational systems [89,90]. Furthermore, the ability of the molecular motors to do mechanical work by moving in opposition to a force allows them to mechanically manipulate bundles of nanowires [58], similar to the natural function of kinesin in mechanical division of the cell during mitosis [38, 128]. When these force-generating motors are hierarchically arranged in a cooperative manner, their collective actions can generate forces that move large internal cellular structures [59] and, on an even larger scale, are the force-generating mechanisms of muscle contractions [31, 65].



www.manaraa.com

Despite the increasing understanding of the biological importance and ubiquitous nature of molecular motors, there is not as yet a definitive understanding of exactly which properties are necessary for a molecular device to function as a translational molecular motor. While there are general thermodynamic and kinetic design principles and constraints [7, 69], there is no single encompassing theory or design that can definitively enumerate necessary and sufficient conditions for designing synthetic walkers that move directionally, processively (i.e., without detachment), and in opposition to a force. In the absence of a general theory, progress in the theoretical design and understanding of molecular motor systems proceeds through the analytic and numerical investigation of models for specific motor designs. Molecular spiders are a particularly interesting type of molecular walker, as they lack many of the properties that are essential to the functionality of other classes of molecular motors. Unlike kinesin and other natural motors, spiders move over arbitrarily arranged 2D tracks, and are able to do so without inherent orientation, structural asymmetry, or chemomechanical coupling between the legs. The gaits of a molecular walker are uncoordinated and acyclic, yet the irreversible modification of surface sites causes an emergent asymmetry in local substrate concentrations that is able to bias the motion of spiders, allowing them to move directionally along prescriptive landscapes. One of the most useful properties of molecular spiders is this structural and chemical simplicity, as it means that the conceptual functionality of a molecular spider is independent of the specific chemical composition of the leg and sites. Molecular spiders are by their nature adaptable to different chemistries, unlike the highly specialized natural walking motors.

## 1.3 The multivalent random walker model

Our multivalent random walker (MVRW) model is a stochastic description of the motion of spiders and spider-like walkers, which uses abstraction to take advantage of the structural simplicity and chemical generality of spider systems. A multivalent random



www.manaraa.com

walker is two-dimensional (2D) with a rigid body and $k$ flexibly tethered legs of length $\ell$. We abstract away the specific DNA chemistry of the spiders to arrive at a general kinetic description whereby a leg can bind to and modify arbitrary chemical species arranged as paths and tracks over a 2D landscape. Each leg is either attached to a site or unattached. A leg can modify an attached site and subsequently detach, leaving behind a different chemical species.

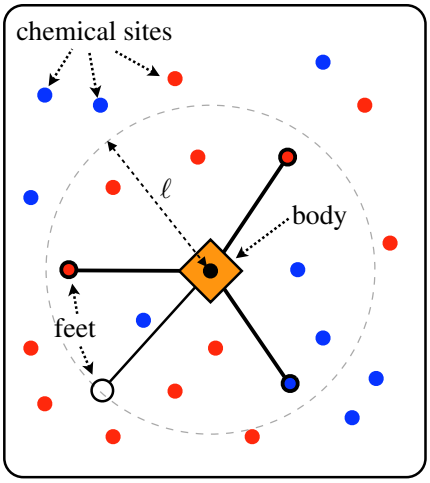


**Figure 1.4:** A multivalent random walker (MVRW). Differently colored chemical sites represent different species.

Mathematically, the MVRW takes the form of a continuous-time, discrete-space Markov process, where each state represents a discrete chemical state of the chemical sites and the walker legs. Transitions in the model correspond to chemical reactions of the legs binding to, unbinding from, or modifying the surface sites. For both biophysical and computational reasons we assume that all other non-chemical (i.e., mechanical) processes come to an equilibrium quickly after each chemical step. From a biophysical viewpoint, the assumption of mechanical equilibrium is plausible because the molecular vibrations and physical bombardment that control the mechanical state of a walker system occur on much faster timescales than the relatively slow chemical reactions. Most molecular motors are best described as near mechanical equilibrium at all times, even though they must operate far from chemical equilibrium to perform ordered work [7]. From a computation viewpoint, equilibrium assumptions are desirable because they greatly simplify the mathematical and algorithmic description of walker motion. The unique mechanical equilibrium positions of the body and unattached legs depend only on the discrete chemical configuration of the attached legs and local chemical sites, and not on any previous chemical states. Thus, the body's continuously parameterized location is not part of the chemical state, so the process state space remains discrete, and the Markov



www.manaraa.com

property holds. Furthermore, as we show in Section 4.3, the translational invariance of the mechanical equilibrium can be exploited to allow precomputation of transition rates for every possible chemical state when walkers move over surfaces with sites arranged on a regular lattice.

## 1.4 Kinetic Monte Carlo simulations

Designing a model for a physical system requires the choice of a particular *coarse-graining* that defines the mapping of the true physical states of the system into a smaller set of states corresponding to the physical properties of interest for predicting the behavior of the system. The particular coarse-graining used in the MVRW model was chosen partly for the scientific reason that it provides the appropriate level of detail to observe the individual chemical reactions that drive the motion of the system. But, an equally important consideration in choosing a coarse-graining is the computational requirements of analyzing the model. The advantage of the MVRW model is that it takes the form of a continuous-time Markov process (CTMP)—a ubiquitous structure in computer science and statistical physics.

A discrete-state, continuous-time Markov Process is defined over a countable state space $\Omega$. The process $X = \{X(t)\}_{t\in\mathbb{R}^+}$ is a time-indexed collection of random variables. Each random variable $X(t)$ is a probability distribution over $\Omega$ at time $t$. The process $X$ must also obey the Markov property, which for continuous-time systems implies that there is a transition rate function $R : \Omega \times \Omega \to \mathbb{R}^+$, such that a transition from state $u$ to state $v$ occurs with constant probability $R(u, v)$ per unit time. $R$ is invariant in time, so we can estimate $X(t)$ given only $X(0)$ and $R$. One way to do this is to approximate $X(t)$ with an ensemble average of many independent realizations of the process. Each realization $x$ is a function from $\mathbb{R}^+$ to $\Omega$, and for each $t$, $x(t)$ is a sampling from the distribution of $X(t)$. If we collect many independent realizations or samples $x_1(t), \ldots, x_n(t)$ we can estimate the



www.manaraa.com

distribution of the random variable $X(t)$ for any time $t$.

We generate a realization $x(t)$ of the process by starting with the initial value $x(0)$ and iteratively using the transition probability rate function $R$ to select a next state and a time until that state transition occurs. We take advantage of the convenient property that the time until the next transition will be exponentially distributed with rate equal to the sum of the rates of all the possible transitions. By repeatedly selecting a next state and incremented time, we can generate a sample of the process up to any desired time, $t_{\max}$. Repeating this procedure from $x(0)$ with a new random seed allows us to generate independent realizations which can be used to estimate various properties of interest about the CTMP $X$. Collecting runs for an ensemble estimate of $X$ is highly parallelizable, as each run is totally independent. This simulation and sampling procedure is called the kinetic Monte Carlo (KMC) method, and was initially developed for simulation of CTMPs in statistical physics models such as the Ising model [87], but is now commonly used to simulate CTMPs in many other fields, including in Gillespie's model of stochastic chemical kinetics [49]. We use the translational invariance of the model to allow precomputation of transition rates based on the finite number of feasible leg gaits for a given spider. Chapter 4 describes the KMC algorithm and other CTMP simulation techniques in detail, along with considerations of the efficiency of such methods.

## 1.5 Multivalent random walkers are molecular motors

Chapter 6 summarizes the major results of our KMC-based numerical simulation of the MVRW model. We use statistical measures of walker motion that are appropriate for analyzing the motion of non-ergodic systems, such as MVRWs. Details of measurements are given in Section 6.1. We investigate the motion of walkers along finite-width tracks of substrates as shown in Figure 1.5. Our results show that MVRWs are able to move superdiffusively, directionally, and processively along molecular substrate tracks when there



www.manaraa.com

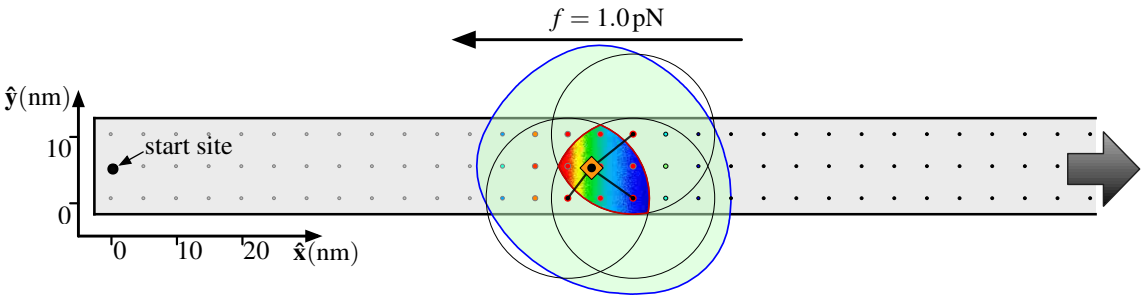


**Figure 1.5:** A MVRW simulation of a walker moving over a semi-infinite track. The walker starts at the origin and must move to the right as there are no chemical sites to the left. A conservative load force applied to the walker body opposes this motion. Walkers can move superdiffusively against this force, using the chemical free energy in surface sites to bias their motion and generate mechanical work.

is a residence time bias between modified and unmodified sites. If leg-product bindings are much longer lived than leg-substrate bindings, the collective constraints imposed by the limited leg lengths cause a directional bias towards the local substrate concentration gradient. In Section 7.1 we explain how this bias, combined with emergent anisotropy in substrate distribution at the boundary between visited and unvisited sites and the irreversibly of substrate catalysis, leads to the prolonged superdiffusive behavior observed in the MVRW model.

Furthermore, we show that the superdiffusive motion persists even when opposed by a conservative load force. By modeling the mechanical motion of spiders at equilibrium, we can directly model the effect of a load force on the walker body and unattached legs. This allows us to directly calculate the mechanical work done by the walker, demonstrating that MVRWs are molecular motors, capable of sustained superdiffusive and directional motion even under load.



www.manaraa.com

## 1.6 Simulation of MVRW systems

We have developed an object-oriented distributed simulation and analysis framework to take advantage of the ensemble-level parallelization opportunities of KMC simulations. Relying on distributed resources for simulation and analysis greatly complicates the management of simulation data and program state. We rely on a relational database and the features of its transactional system to protect against incompatible concurrent access patterns to simulation data. To make the relational data easy to work with we use object-relational mapping to map persistent objects in an object-oriented program to tuples in a relational database. Our MVRW simulation framework is written in Python and uses the SQLAlchemy object-relational mapping package. The simulation framework is summarized in Chapter 8.

**Computational efficiency.** Because the MVRW model is a CTMP, we can use the KMC algorithm as the basis for the simulation of independent realizations of MVRW traces under many different parameter settings. The KMC algorithm and its efficiency, as well as other techniques for analyzing CTMPs, are discussed in detail in Chapter 4. Unlike other CTMP models of chemical reactions on the molecular scale, such as the Gillespie model [49], our MVRW system does not possess a regular structure of state transitions that can be exploited for efficiency. Thus, our KMC algorithm will be specialized for the MVRW system and cannot take advantage of the many recent improvements on the Gillespie KMC simulation algorithm (Section 4.1.2). Systems like the Ising model have spatial extent, allowing many essentially independent transitions in a single KMC simulation trace to be parallelized by blocking regions of system space [17]. However, a similar parallelization opportunity is not present in MVRW simulations because chemical reactions only occur in the region directly adjacent to the walker. In any case, the statistical analysis of the MVRW model requires an ensemble of independent sample traces, and the natural ensemble-level parallelization leads to efficient linear scaling, at least for moder-



www.manaraa.com

ately sized simulations and computational resources.

**The natural entity object-relational mapping framework.** In object-relational mapping (ORM), objects are stored persistently as tuples in a relational database, but manipulated as in-memory objects in an object-oriented (OO) language. There are fundamental semantic differences between OO data models and relational data models that make this mapping difficult to implement consistently and correctly. The concepts of object identity and uniqueness are a particular concern for MVRW simulations, but they are represented differently in the two data models. Relational databases use a value-based concept of identity for tuples, while OO languages treat object identity as explicitly independent of value. Our concern for the MVRW simulations is that this discrepancy can result in concurrent processes representing the same conceptual object in more than one object or tuple, leading to duplication or loss of data. To resolve the ambiguities of persistent object identities in ORM, we introduce the *natural entity framework* [92], which is described in Chapter 9. This tool works on top of the SQLAlchemy ORM, defining a new `NaturalEntity` base type for persistent objects that allows the OO environment to directly enforce value-based object uniqueness.

**Data management.** In Section 8.1 we discuss storage and access times for large computational datasets. Ultimately, we found that large numerical datasets are best kept outside of the relational database, as the overhead associated with database access is too expensive for practical use. To meet our need for large numerical array storage, we use the HDF5 hierarchical data format to store simulation state and measurement data for MVRW simulation runs. However, the database and the object-relational mapping remain essential for maintaining the referential consistency for these external datasets because the HDF5 library lacks any built-in concurrency protection or transactional semantics.



www.manaraa.com

**Random number generation.** Monte Carlo simulations are only as good as the random number generators they depend on [39]. In single-threaded contexts, many good pseudo-random number generators exist that come with mathematical guarantees regarding correlation and other statistical measures of randomness for the single-threaded stream of random numbers generated [71, 84]. However, random number generation becomes more complicated in a parallel context. In our parallel KMC and Metropolis-Hastings simulations, we maintain these mathematical guarantees by using the leapfrogging method [11] to divide a single random number generator stream into an arbitrary number of parallel streams, giving each simulation an independent slice of random numbers from a single master stream (Section 8.2). In this way, each individual simulation from the ensemble of $N$ can be executed concurrently in arbitrary order, yet the entire set of simulations remains *exactly* equivalent to a single process iteratively computing a single step for each of the $N$ simulations in turn using a single master stream.

## 1.7 Dissertation overview

This dissertation comprises a comprehensive mathematical and computational analysis of multivalent random walker motion. In the process we have made contributions from the chemical and biophysical modeling level to the software engineering level. Each of these contributions builds up to the overall result, but for clarity we have divided the material into two main parts. The first part covers the modeling, simulation, and biophysical aspects of the model; the second covers the software engineering aspects. A shorter third part collects concluding thoughts and future work.

### 1.7.1 Part I: The multivalent random walker model

In our first publication [93] we addressed the MVRW model as it relates to molecular spiders. This publication introduces a simplified version of MVRW model which is given



www.manaraa.com

in expanded detail in Chapter 3, and discusses the KMC simulation of spiders and the use of Metropolis-Hastings sampling for equilibrium estimates. We expand and elaborate on that KMC material in Chapter 4, and the Metropolis-Hastings sampling material in Chapter 5. Our method of using the translational invariance of the equilibrium body distribution to pre-compute transition rates for walkers moving over regular lattices is given in Section 4.2, and is the focus of a document in preparation [95].

Our central biophysical result is that multivalent random walkers are molecular motors, capable of transforming chemical free energy into biased superdiffusive motion [94]. In support of this hypothesis we present the simulation results and numerical analysis in Chapter 6. Chapter 7 explains how leg kinetics lead to an effective residence time bias between unvisited substrates and visited products. An emergent anisotropy in substrate concentrations allows this kinetic bias to break symmetry and generate an effective directional bias in the direction of unvisited sites (Section 7.1). This result has relevance to our preliminary research into simple 1D spider models which revealed similar mechanisms of superdiffusive motion for both single spiders [112] and cooperative swarms of spiders [113, 114].

### 1.7.2 Part II: The multivalent random walker simulation framework

In Part II, we address the complexities of distributed simulation and analysis of the MVRW model. It is often difficult to write about simulation software because many interesting ideas and methods are application-specific and not of general interest. Instead, we focus on implementation concerns and design choices that are cross-cutting through all levels of the simulation program structure and organization.

At a high level our simulation framework is an object-oriented distributed simulation and analysis framework, which manages simulation data and objects concurrently using object relational mapping (ORM) to store simulation objects persistently in a relational database. Chapter 8 gives an overview of the simulation framework and the associated



www.manaraa.com

concerns, including large data storage (Section 8.1), and parallel random number generation (Section 8.2).

In order to resolve the object identity and uniqueness ambiguities that arise in object-relational mapping, we devised an ORM tool called the *natural entity framework* that is particularly useful in managing simulation objects consistently and correctly given the complexities of concurrent data access. Chapter 9 presents this framework based on our previously published work [92].

### 1.7.3 Part III: Conclusions

To give perspective to these results, in Chapter 10 we present our outlook on the concepts involved in computer simulation of chemical and biological systems. We make the case that Markov process models like the MVRW model are particularly intriguing as they represent a *computational understanding* of the chemical structure and mechanisms of the system. Through the KMC simulation technique, the computational model essentially executes an abstract stochastic experiment. In our case this means that at the same time we are running simulations to calculate abstract statistical measurements, we are actually step-by-step following a virtual walker along its unique random trajectory.



www.manaraa.com

# Part I

# The Multivalent Random Walker Model: Simulation and Results



www.manaraa.com

# Chapter 2
# Molecular Spiders

The concept of a *molecular spider* introduced in Section 1.1 is inspired by a need for simple synthetic molecules that follow prescriptive tracks on nanoscale landscapes, and hence provide a means of controlled locomotion for artificial nanoscale systems. Recent experiments [82] have demonstrated that molecular spiders follow short paths, but it remains nearly impossible to observe spiders on the space and time scales necessary to discern the individual chemical and mechanical actions of the legs and body (Section 2.6). To understand how these individual stochastic reactions lead to the overall motion of spiders or other multivalent random walkers, we must rely on mathematical models and computer simulations that derive from a microscopic understanding of the chemistry of the molecular spiders and the associated kinetics. This chapter reviews the relevant biological and chemical background necessary to understand the structural and kinetic properties of molecular spiders and how the multivalent random walker model abstracts these details to arrive at the appropriate coarse-graining of walker dynamics.

## 2.1 Chemical kinetics and enzymatically controlled reactions

Chemical reactions describe the transformation of matter from one chemical species to another, and are represented by reaction schemes, such as

$$\mathrm{A} + \mathrm{B} \rightarrow \mathrm{C} + \mathrm{D}. \tag{2.1}$$

In this familiar convention, *reactants* A and B are transformed into *products* C and D. A reaction proceeds at a rate proportional to the concentrations of the reactants. Due to conservation of mass, the rate of increase in the products is exactly equal to the rate of decrease of the reactants. A reaction scheme is a high-level representation of a very



www.manaraa.com

detailed molecular process, and need not correspond to the actual sequence of chemical events that take place to produce the net reaction effect. In order to understand how the kinetics of a reaction are related to concentrations of the product species, we need to represent a reaction in terms of *elementary reactions*, which correspond directly to the making and breaking of chemical bonds at the molecular level.

An elementary reaction is either *unimolecular*, if it involves a single reactant species, or *bimolecular*, if it involves two species. While reaction schemes allow forms with more than two reactants, such reactions are in reality carried out by a sequence of elementary uni- and bimolecular reactions [57]. Unimolecular reactions occur when a molecule spontaneously breaks or rearranges internal chemical bonds. A bimolecular reaction occurs when two reactants collide with enough energy to react by breaking or forming bonds. The difference in the free energy of the reactant species and the product species determines if the reaction will occur spontaneously. In solution chemistry, this difference is given by the *Gibbs free energy*, $\Delta G$, which quantifies the change in free energy between reactants and products under constant temperature and pressure, while correcting for changes in entropy caused by the reaction. If

$$\Delta G = \Delta H - T\Delta S = H_{\text{products}} - H_{\text{reactants}} - T\Delta S < 0, \tag{2.2}$$

the reaction will proceed towards the products. In Equation 2.2, $\Delta H$ is the change in enthalpy, which at constant pressure is the change in internal energy when the reactants are transformed into the products; and $\Delta S$ is the change in entropy, which is a function of the concentrations of the various species.

While $\Delta G$ determines the direction of a reaction, it does not determine how fast it will happen. The *reaction rate* is determined by the energy necessary to make and break internal chemical bonds as atoms are rearranged through a series of higher-energy intermediate states. At the atomic level a chemical reaction is a continuously parameterized transformation of the 3D atomic arrangement. Each arrangement $x$ corresponds to a particular energy $E(x)$. One way to visualize a reaction is as a random walk in the high-dimensional



www.manaraa.com

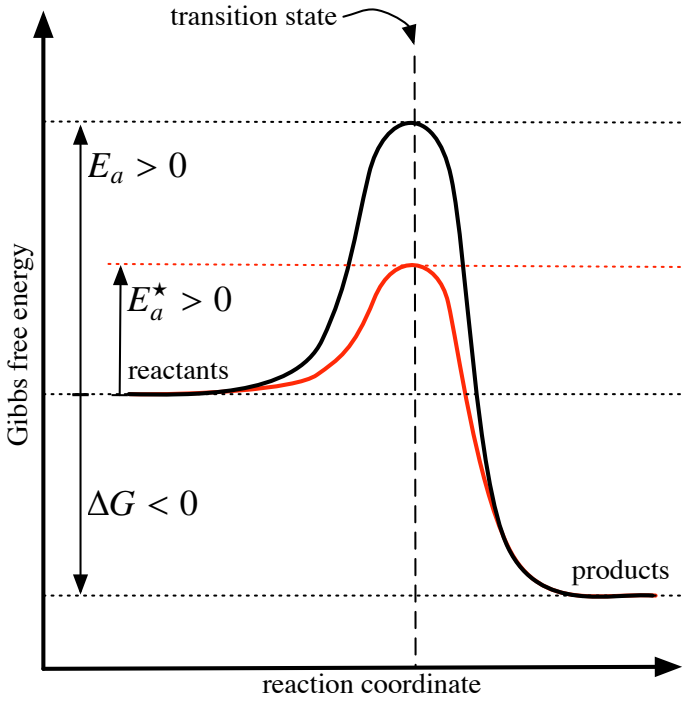


**Figure 2.1:** A chemical reaction is parameterized by a reaction coordinate, which represents the lowest energy path from reactants to products. If $\Delta G < 0$, the reaction proceeds spontaneously from reactants to products. The activation energy $E_a$ controls the rate at which the reaction proceeds, as reactants must gain at least this much energy to react. A catalyst lowers the activation energy to $E_a^\star$ (shown in red), hence speeding up both the forward and reverse reactions.

space of atomic positions. Any continuous path of atomic arrangements leading from reactants to products is a *reaction pathway*. Most microscopic models of kinetics make the *transition state hypothesis* [57], which assumes that the reaction pathway with the smallest energy maximum is dominant. The position along this parameterized pathway is called the *reaction coordinate*. Figure 2.1 shows a typical Gibbs free energy profile parameterized by the reaction coordinate. The *transition state* is the state at the energy maximum. The difference between the reactants' energies and the transition state energy is called the *activation energy*, $E_a$, because any reactants must pass over this *energy barrier* before they can assume the lower-energy product conformation. The dependence of the reaction rate $k$ on the activation energy $E_a$ and absolute temperature $T$ is given by the Arrhenius equation,

$$k \propto e^{-E_a/k_B T}. \tag{2.3}$$

Here $k_B$ is Boltzmann's constant, and the ratio of $E_a$ to $k_B T$ gives the ratio of the activation energy to the background thermal energy. The rate of the reaction becomes exponentially small as $E_a$ increases. This can make reactions very slow when the activation barrier is



www.manaraa.com

much higher than $k_BT$.

### 2.1.1 Catalysis

A *catalyst* is a chemical that increases the rate of a reaction by enabling a new reaction pathway with a lower-energy transition state. Figure 2.1 shows the effect of a catalyst on a reaction, lowering the activation energy from $E_a$ to $E_a^\star$. Because of microscopic reversibility, a catalyst that lowers the activation energy of the forward reaction from reactants to products also lowers the activation energy for the reverse reaction pathway from product to reactants. However, if $\Delta G/k_BT \ll 0$, the rate of reverse reaction can be negligibly small, making the catalyzed transformation essentially irreversible.

In a biological context a catalyst is called an *enzyme*; most known enzymes are proteins. However, certain single-stranded DNA sequences have been shown to function as catalysts [19, 22]. Such a DNA sequence is called a *deoxyribozyme* to emphasize its enzymatic function. In typical biological settings the reactants that an enzyme helps to convert to products are called *substrates*. Consider the unimolecular reaction in which a substrate S is converted into two products $P_1$ and $P_2$:

$$\mathrm{E} + \mathrm{S} \underset{k_-}{\overset{k_+}{\rightleftarrows}} \mathrm{ES} \xrightarrow{k_{\mathrm{cat}}} \mathrm{E} + \mathrm{P}_1 + \mathrm{P}_2. \tag{2.4}$$

An enzyme E catalyzes this reaction by reversibly binding to S to form an enzyme-substrate complex ES. The rates $k_+$ and $k_-$ control how fast the enzyme binds to and unbinds from the substrate. The ES complex can also undergo a catalyzed reaction to form products $P_1$ and $P_2$. There are in fact several elementary reaction steps in this process: the actual catalysis, and the subsequent unbinding of both products. For simplicity we represent these combined steps by a single step with rate $k_{\mathrm{cat}}$. While catalysis is in theory reversible, allowing $E$ to bind $P_1$ and $P_2$ and transform back into an ES complex, this reaction becomes negligible when $\Delta G/k_BT \ll 0$, which must be the case for a molecular motor to prevent reverse motion [23]. This kinetic description, summarized in Equation 2.4, is



www.manaraa.com

used to model the kinetics of the legs in a molecular spider. We revisit Equation 2.4 in the context of modeling walker leg reactions in Section 3.4.

## 2.2 DNA chemistry

Of all biomolecules, DNA has been the most studied molecule for the design of computational chemical reaction networks, due to its predictability in Watson-Crick base pairing, its intrinsic information storage ability, and its long-term stability at room temperature. DNA has been used to design logic gates [116], game-playing automata [83, 117], finite automata [12], combinatorial search algorithms [1], and other computational media [118]. Our previous work has also shown that DNA has the ability to act as an aptamer, recognizing and signaling the presence of other small molecules [53, 97]. The existent structural and computational DNA technology can potentially be combined with the perspective motility of molecular spiders to create programmable sensing, computing, transport, and communication in synthetic nanoscale systems.

A single strand of DNA is a polymer of individual units called *nucleotides*. Each nucleotide consists of a phosphate, a deoxyribose sugar, and one of the four bases A, T, C, or G. Polymerization attaches the nucleotides covalently to form a backbone of alternating phosphates and sugars. Two single strands of DNA or two parts of the same strand can bind to form a double strand by pairing bases according to the Watson-Crick rules (A $\leftrightarrow$ T and C $\leftrightarrow$ G). This base pairing process is called *hybridization* and occurs through the formation of low-energy hydrogen bonds, which are individually much weaker than the covalent bonds within a strand, but the effective binding strength is increased as the number of paired bases (also known as the *hybridization length*) increases. Hybridization and dissociation reactions allow the legs of molecular spiders to attach and detach from substrate DNA, and the weak hydrogen bonds are easily made and broken ensuring the legs can move and rebind quickly. More elaborate hybridization reactions also allow DNA



www.manaraa.com

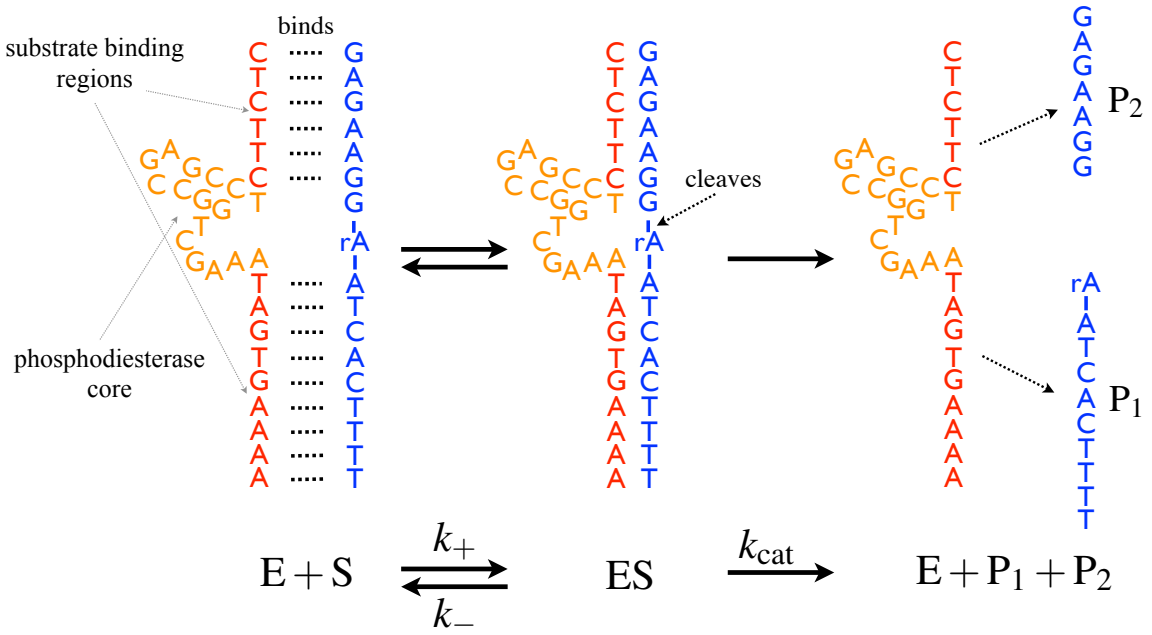


**Figure 2.2:** The foot of a molecular spider is an 8-17 deoxyribozyme (E) that binds reversibly with a complementary oligonucleotide substrate (S). The enzymatic core of the deoxyribozyme can catalyze the cleavage of the substrate backbone at a ribose impurity directly across from the enzymatic core. This creates two product oligonucleotides ($P_1$ and $P_2$), which subsequently dissociate, leaving the enzyme free to react again. A free enzyme remains complementary to the products, so they can rebind, but not as strongly since they are shorter. For molecular spider systems $P_1$ remains attached to the surface and is frequently rebound by spider legs, while $P_2$ is lost to solution and the effective concentration of free $P_2$ is assumed to be low enough that rebinding does not occur.

to form the 2D and 3D rigid structures [101] and these are used to create the substrate-covered surfaces the spiders move across [82, 109].

The deoxyribozyme legs of a molecular spider catalyze the cleavage of *oligonucleotides*—other short single-stranded DNA sequences [20]. Because they break the substrate oligonucleotide backbone at a phosphate bond they are called *phosphodiesterases*. The catalytic mechanism requires that the substrate have a substitution of a ribose sugar for a deoxyribose along the backbone, allowing the desired cleavage site to be programmed into the substrate sequence. The 8-17 phosphodiesterase deoxyribozyme acts as the legs of the molecular spiders. Its kinetics are given by Equation 2.4, and shown graphically in



www.manaraa.com

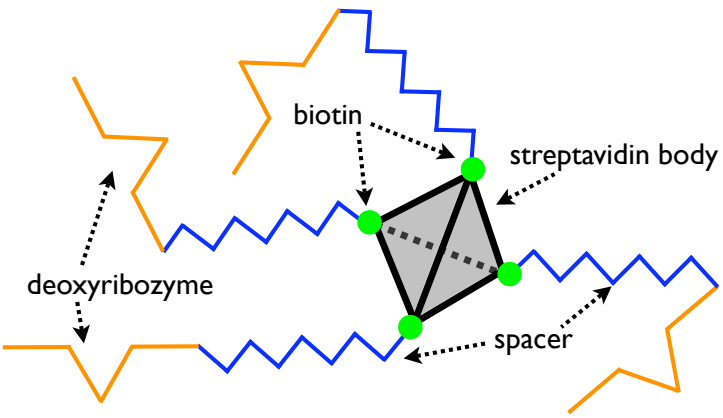


**Figure 2.3:** A molecular spider has a streptavidin body, which accepts up to four legs. Each leg consists of a biotin which binds to streptavidin, followed by a flexible chain-like spacer and a deoxyribozyme foot.

Figure 2.2.

## 2.3 Molecular spiders

The body of a molecular spider [98] is a tetrahedral protein, called *streptavidin*, with four binding sites (Figure 2.3). The body acts solely as a rigid scaffold to bind the individual legs together and has no chemical activity of its own. At the hip end of each leg is a biotin—a small molecule that attaches irreversibly to one of the four binding sites on the streptavidin body. The foot end of the leg is a deoxyribozyme, and is the only chemically active part of the spider. The length of the leg can be varied by adding inert, flexible spacer molecules. The deoxyribozyme binding and unbinding rates ($k^+$ and $k^-$) can be changed to a limited extent by altering the length and sequence of the substrate recognition regions.

When a spider is released on a surface covered with immobilized oligonucleotide substrates, the deoxyribozyme legs attach to, cleave, and detach from the substrates, according to the kinetics shown in Figure 2.2. Upon cleavage and dissociation, only the lower product $P_1$ remains bound to the surface. The upper product is lost to solution where bulk



www.manaraa.com

concentration is essentially 0, and we assume it has no further effect on the kinetics of walker motion. There are five reactions controlling the chemical kinetics of a leg (L) reversibly binding to surface-bound substrate (S) and product (P) sites:

$$\begin{aligned} \mathrm{L} + \mathrm{S} &\underset{k_{\mathrm{S}}^{-}}{\overset{k_{\mathrm{S}}^{+}}{\rightleftarrows}} \mathrm{LS} \xrightarrow{k_{\mathrm{cat}}} \mathrm{L} + \mathrm{P} \\ \mathrm{L} + \mathrm{P} &\underset{k_{\mathrm{P}}^{-}}{\overset{k_{\mathrm{P}}^{+}}{\rightleftarrows}} \mathrm{LP} \end{aligned} \tag{2.5}$$

As the legs attach and detach, the entire spider structure moves over the surface in a process controlled by the chemical reactions of the attached legs and the physical process of the constrained diffusion of the body and unattached legs. As long as at least one leg of the spider is attached to a chemical site on the surface, the spider will remain tethered to the surface. If all the legs detach, the spider is free to float away in solution. Thereafter, it may diffuse over the surface and rebind to other sites, or be washed away from the surface by a flow designed to prevent rebinding.

## 2.4 Spiders with DNA tile bodies

Although the streptavidin-based spiders are the most experimentally studied structural variety of molecular spiders, researchers at Arizona State University, Kyle Lund and Hao Yan, have developed a more configurable and expandable scaffold for molecular spiders using self-assembled DNA tiles [81]. Each tile has an attachment point for the leg, and the tiles can be programmed to self-assemble into any polyomino [51] shape. We call these types of spiders *segmented spiders*. Figure 2.4a shows the detailed 3D structure of a segmented spider, and Figure 2.4b is our abstract representation of the same spider. Whereas the streptavidin spiders have all the hip locations essentially at the same point, the segmented spiders have bodies with spatial extent and hip locations are arranged throughout the body. Segmented spiders may be able to achieve more directional bias in their motion,



www.manaraa.com

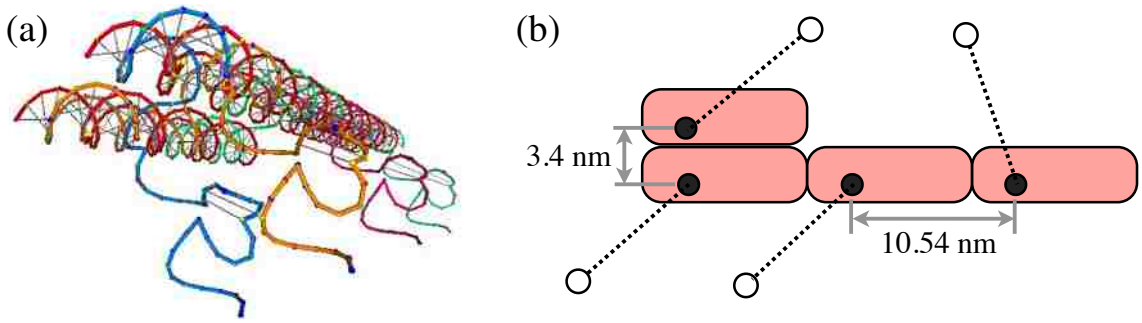


**Figure 2.4:** (a) A 3D structural image of a segmented molecular spider with 4 segments (Kyle Lund, Arizona State) (b) An abstract representation of the same spider.

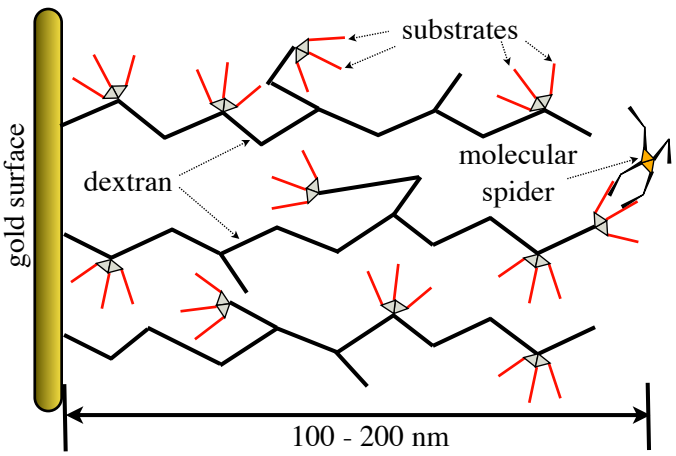


**Figure 2.5:** Pei et al. used SPR to measure spider cleavage rates for systems where multiple streptavidin spiders move through a pseudo-3D, forest-like environment of dextran-bound substrates [98]. Substrates are bound to streptavidin molecules which in turn are bound to the dextran strands at random junctions. This experiment shows that individual spiders on average cleave more than 3800 substrates without dissociating—in other words that they are *processive*.

since their asymmetry prevents them from rotating freely while multiple legs are attached. Our MVRW model (Chapter 3) and simulation framework are designed to allow the simulation of segmented and other general 2D walker bodies, but further work is needed to make these simulations efficient enough to admit results for these more complicated walkers.

## 2.5 Spider environments

A molecular spider can step over substrate sites laid out either at random, or as prescriptive tracks and paths. In the original spider experiments (Figure 2.5), substrates were



www.manaraa.com

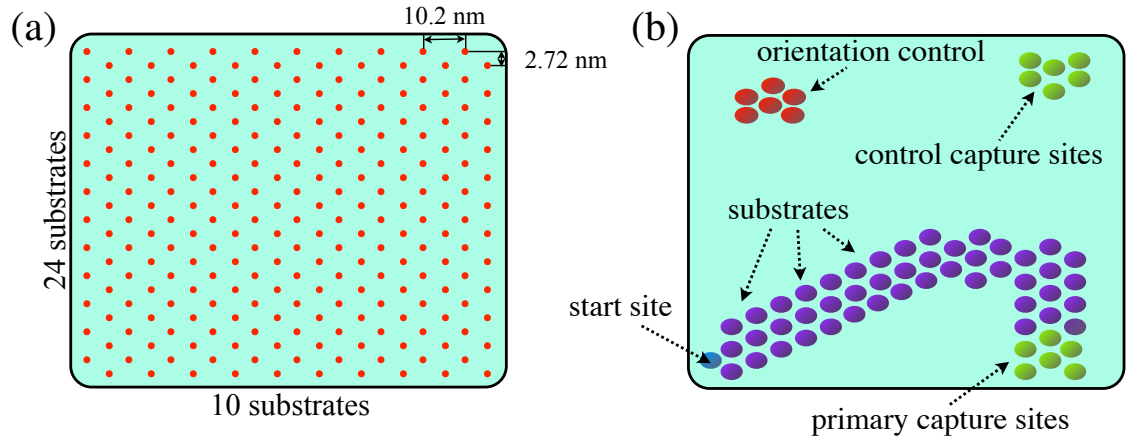


**Figure 2.6:** (a) An addressable lattice of chemical sites can be built on a DNA origami scaffold. The substrate spacing is 10.2 nm × 2.72 nm in a hexagonal lattice pattern. Due to the technological constraints of DNA origami, these lattices are currently limited to 10×24 sites. (b) Lund et al. used AFM to observe streptavidin spiders moving over the track in this DNA origami environment [82]. Walkers moved from the start site to the primary capture site without moving onto the control capture site. This shows walkers move processively (i.e., without detachment), and they can follow prescriptive paths and tracks. (This image is adapted from material provided by Kyle Lund, Arizona State.)

displayed on a pseudo-3D matrix of dextran strands [98]. This experiment used surface plasmon resonance (SPR) [61] to measure the mass decrease of this matrix as a swarm of spiders moved through it, cleaving substrates and releasing product $P_2$. The mass was observed to decrease linearly, corresponding to the spider swarm consuming substrate at a constant rate. The spiders consumed close to 100% of the available substrate, eventually slowing their rate of consumption as substrate supplies were exhausted. Significantly, this demonstrates that spiders are processive, (i.e., they have large turnover of substrates before detaching), as each spider on average cleaved at least 3800 substrates without dissociating.

If, however, spiders are to be used for directional transport they require environments with prescriptive paths and tracks, guiding the walker's motion in useful ways. Lund et al. [82] have created addressable 2D surface environments for molecular spiders (Figure 2.6a) using the *DNA origami* [109] self-assembly technique, where a long single-stranded DNA sequence is folded using algorithmically designed sequences of DNA as



www.manaraa.com

stable strands, creating a rectangular region with regularly spaced and individually addressable binding sites. Using atomic force microscopy (AFM) [16], spiders have been observed following non-linear tracks of substrates over origami surfaces [82] (Figure 2.6b). The size of the origami surfaces limits the length and complexity of the tracks; however, nano-lithography [100] or other more sophisticated DNA self-assembly mechanisms may be able to produce arbitrarily large and complex networks of paths for walkers to move on and interact with each other.

## 2.6 Limitations of experimental observations

The typical size of a single-stranded DNA polymer is 2.2–2.6 nm wide, with a length of 0.33 nm per base. The streptavidin molecule in the body of a DNA walker is approximately 4.5 nm × 4.5 nm × 5.0 nm, and the spacing of substrates on a DNA origami surface is 10.2 nm × 2.72 nm. Observing the individual chemical actions of the molecular spider legs binding and unbinding would require sub-nanometer measurement precision.

When viewing light-emitting objects at very high resolutions, the fundamental limits of diffraction come into play. An object like a nanometer-scale fluorophore emitting light will appear as a blurry disk of light, called an *Airy disk*, which is well approximated by a Gaussian. The limit of resolution in microscopy is defined by the ability of a microscope to distinguish two adjacent Airy disks. The resolution distance for light of frequency $\lambda$ viewed through a lens with numerical aperture $A$ is given by $d = \lambda/2A$. With typical values of fluorophore emissions in the green part of the spectrum ($\lambda \approx 500$ nm) and an upper bound on practical aperture numbers around 1.5, this gives an effective resolution of 167 nm [110]—much too large for effectively tracking the individual actions of molecular spiders. Super resolution techniques can use maximum likelihood techniques to estimate the center of a point emitter to much higher accuracy [64]. These techniques have been used to observe individual spiders moving over a random surface of substrate sites [82].



www.manaraa.com

However, even super-resolution fluorescence microscopy has a spatio-temporal resolution much coarser than the level of individual chemical reactions that the MVRW model describes. The same is true for atomic force microscopy (AFM). While AFM has very high spatial resolution, it has extremely low temporal resolution, and has at present only yielded very coarse time series data for spiders [82].

In the future, more sensitive microscopes and more sophisticated experimental techniques may provide the type of resolution needed to discern the individual chemical actions of molecular spiders. For the present, however, constraints on spatio-temporal resolution for direct imaging of spider systems prevent us from observing spiders at the level necessary to validate individual chemical aspects of a spider model. Hence, in this work we focus on a more abstract and general class of multivalent random walkers that focuses on the minimal chemical and mechanical features that are necessary for spider-like systems to behave as molecular motors. As detailed structural and kinetic information is made available, more concrete and sophisticated molecular spider models can be individually validated through experimental agreement, but we expect these future models to share the same mathematical and structural underpinnings established in the MVRW model as described in detail in Chapter 3.



www.manaraa.com

# Chapter 3
# The Multivalent Random Walker Model

The molecular spider system described in Section 2.3 was the original motivation for developing computational models of nanoscale walker motion. However, by abstracting away the detailed structural and chemical specifics of the molecular spider systems, we arrive at a model that is simultaneously more general and more powerful. Our multivalent random walker (MVRW) model describes the motion of a general class of enzymatic walker molecules and allows us to investigate the minimal geometrical, structural, and chemical constructs sufficient to control these walkers for directional, prescriptive transport. Our central scientific contribution is to show that multivalent random walkers behave as a new type of molecular motor. We developed our model to not only simulate the motion of the walkers, but also to explain the molecular mechanisms through which the chemical energy in substrate sites can be transformed into directional motion. To generate such a detailed molecular understanding, we account for both the high-frequency mechanical vibration and Brownian motion of the walker body and legs, as well as low-frequency chemical reactions corresponding to stochastic transitions between discrete chemical states. To demonstrate the walker's characterization as a molecular motor we directly calculate the work exerted by each walker as it moves against an external load force. Using an intermediate level of coarse-graining, we assume the walker comes to a mechanical equilibrium in between chemical steps. From a biophysical perspective, this level of representation allows us to accurately model the effect of external load on the walker's body through its effect on the equilibrium mechanical position of the walker, yet still maintain a discrete state space. From a computation perspective, equilibrium estimates of leg and body distributions are Markovian and translationally invariant, leading to opportunities for precomputation and optimization.

The MVRW model is a continuous-time, discrete-state Markov process (CTMP) that describes the action of a MVRW as a sequence of stochastic state transitions. From any



www.manaraa.com

state, the mechanical equilibrium of the body and legs determines a finite set of possible state transitions corresponding to the feasible chemical reactions of the legs with local sites. By assuming that the body and unattached legs are at equilibrium, the state transitions occur at rates dependent only on the current chemical state of the system and not on any previous chemical states; this gives the system the Markov property.

In order for the MVRW model to accurately represent the physical and chemical events of a walker system, we must make several abstractions and assumptions, and we must carefully define the state of the system to encompass all of the variables that would affect the transition rates under these assumptions. Section 3.1 gives background on the different levels of modeling granularity that can be used to describe chemical systems, as well as the abstractions and assumptions we make to arrive at a discrete stochastic model for the walkers. In Sections 3.2 and 3.3, we consider in detail the two components of the discrete chemical states of MVRW model: the state of the environment, and the state of the walker. The transition rates between these chemical states are derived from simple kinetic and mechanical assumptions in Sections 3.4, 3.5, and 3.6, allowing us to model the uni- and bimolecular reactions of any enzymatic leg/substrate chemistry. Building on these derivations, Section 3.7 shows how the attachment rates and mechanical equilibrium state of the body are affected by the presence of a constant load force. Finally, in Section 3.8 we consider how the irreversibility of substrate cleavage reactions serves to partition the state space and how this affects the equilibrium system dynamics.

## 3.1 Modeling chemical reaction systems

The kinetics of chemical systems like molecular spiders can be modeled at many different scales from fine- to coarse-grained, depending on the level of detail one is interested in and the nature of the assumptions one makes about the system.



www.manaraa.com

**Macroscale kinetics and mass action.** Historically, chemical kinetics has been studied from a macroscale viewpoint, in which chemical species are represented by their concentrations per unit volume. Provided the solutions are well mixed, the number of molecules is very large, and the reactions occur slowly compared with the diffusion rate of the chemical species, it is justified to assume that the concentration of any species is uniform over the volume within the reaction chamber and can be represented by a single positive concentration value at each time. Under such conditions, the rate of an elementary reaction (Section 2.1) is observed to be proportional to the product of the concentrations of the reactants. This is known as *mass action kinetics*, and gives rise to a system of differential equations whose solution is the time-varying concentrations for each species. Mass action kinetics is a deterministic model representing molecules as continuous concentration functions in time. In reality, molecules are discrete objects and reactions are discrete events that occur stochastically. These realities have increasing importance as the size of the reaction volume and the number of molecules become smaller, as is the case for single walker molecules moving over a surface of chemical sites.

**Molecular dynamics.** At the opposite end of the size and time scales, a *molecular dynamics* model is a detailed representation of a system that tracks the position and momentum of every individual atom—those in the reactive species as well as those in the solvent molecules. The dynamics of the atoms is then modeled using electromagnetic forces derived from quantum-mechanical approximations [15]. In such a model it becomes clear how collisions between solvents and reactants are responsible for the Brownian motion of molecules and the random energy fluctuations that enable reacting molecules to pass over energy barriers. Such detailed models are necessary to understand the intra-molecular mechanisms by which enzymes attach to substrates and catalyze their transformation. There is, however, a huge computational price to pay for such detail. Every collision and internal molecular vibration must be simulated and these events occur at intervals as short as $10^{-14}$ s. For these reasons, molecular dynamics simulations of DNA reactions span only



www.manaraa.com

tens or hundreds nanoseconds of simulated time [36, 130]. The MVRW model only needs to describe the long-term transport properties of the system as the walkers take many steps over the surface; there is no need to understand how electromagnetic potentials allow a leg to bind with a substrate, only how long it is likely to take for the reaction to occur from a given molecular state. Nor is it necessary to track the solvent species explicitly; although the solvent interactions are truly what drives the motion of the walker and legs, they occur orders of magnitude more frequently than actual reactive collisions and can be approximated as at equilibrium on the timescale of chemical reactions.

### 3.1.1 Stochastic chemical kinetics

In between the extremes of very coarse grained mass action kinetics and very fine grained molecular dynamics, there is an intermediate level of granularity, in which individual chemical events are explicitly represented, but other mechanical processes are assumed to be at equilibrium. This level of modeling is known as *stochastic chemical kinetics*, and it describes a system as a continuous-time stochastic process over the individual discrete chemical states of the system. For appropriate assumptions and state spaces these processes are Markovian, making them particularly simple to simulate using KMC techniques (Chapter 4).

The Gillespie model of chemical kinetics [48, 49] is one such CTMP model. It describes the stochastic kinetics of well-mixed, dilute solutions of reacting molecules in a fixed volume, at constant temperature and pressure. Under these conditions, the reactive species are always near physical equilibrium, and distributed uniformly over the volume. Thus, the probability of any particular reaction occurring is simply proportional to the number of reactant molecules for that reaction. For unimolecular reactions, this is just the number of molecules of the reactant, and for bi-molecular reactions, this is the number of distinct pairs of reactant molecules. The proportionality constant for each reaction can be determined with the Arrhenius relation between reaction rate and the activation energy at



www.manaraa.com

a constant temperature (Equation 2.3).

However, MVRWs are by construction the opposite of well-mixed. The chemical surface sites have fixed locations, and when walkers are attached to these sites they have a bounded region of feasible sites they might attach to. Outside of this region, no reactions are possible. Thus, enumerating the possible reactions is much more complicated and model-specific than the general Gillespie stochastic model. In developing the MVRW model, we use the position-independent unimolecular reaction kinetics from the Gillespie model, but we develop our own theory to describe the rates of the tethered chemical kinetics of leg–site binding (Section 3.6.1).

## 3.2 The environment

A walker moves over a 2D space of chemical sites, called the *environment*. The chemical composition of the sites is modified by the actions of the walker, but the site locations are stationary. We define the environment mathematically with the following objects.

- $S \subset \mathbb{R}^2$ — The set of fixed chemical sites can be arranged as arbitrary tracks or paths on the surface.
- $\Sigma$ — The set of chemical species is normally taken to be $\{\mathrm{S}, \mathrm{P}\}$ for the substrate-product chemistry in Equation 2.5.
- $\pi : S \to \Sigma$ — The species function is a mapping from a site to the species displayed at that site.

We assume $S$ and $\Sigma$ are fixed initial conditions, while the mutable state of the environment is represented solely by the species function $\pi$. The set of all possible states of the environment is then all the possible mappings $\pi$ from sites to species:

$$\mathbb{E} = \{\pi_i : i = 1, \ldots, |\Sigma|^{|S|}\} = \Sigma^S. \tag{3.1}$$



www.manaraa.com

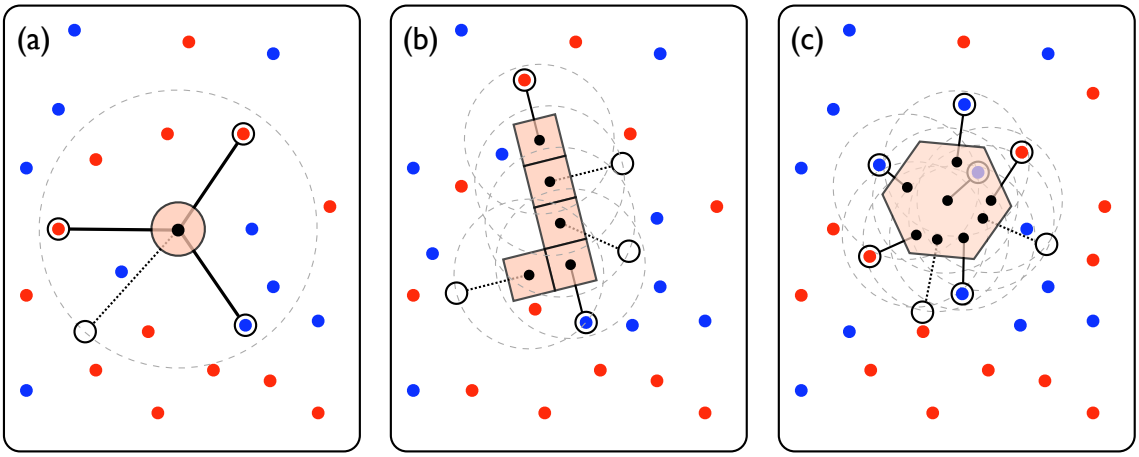


**Figure 3.1:** Walker types are differentiated by the body shape and location of hips on the body. (a) A *point-bodied walker* has all hip locations at the same point. (b) A *segmented walker* has a body composed of segments with a hip location at the center of each segment. (c) A *general walker* has arbitrary hip locations, and an arbitrary body shape.

## 3.3 Walkers

The body of a multivalent random walker is a rigid, inert 2D scaffold with a set of $k$ attachment points for legs, called *hip locations*. Each of the $k$ legs has a reactive *foot* that is flexibly tethered to a hip location with maximum extension length $\ell$. We avoid any detailed structural description of the legs or their intra-molecular interactions. Instead the legs remain independent and uncoordinated, constrained only by the limited tether extension length.

### 3.3.1 Walker body shapes

Several different types of walker bodies are possible depending on the layout of the hip locations on the body. Figure 3.1 illustrates the three main classes of multivalent walker bodies. Point-bodied walkers are motivated by the streptavidin body of molecular spiders (Section 2.3). The symmetry of point-bodied walkers makes them more straightforward to model and simulate. The segmented walkers mimic the design of DNA tile spiders (Sec-



www.manaraa.com

tion 2.4), and allow design of non-symmetric walker body shapes. While our results in Chapter 6 exclusively focus on point-bodied walkers, we define the model to also encompass the parameters associated with segmented and general walkers. Our future work will investigate the transportation advantages that can be gained by controlling body shape and the interaction with substrate spacing.

### 3.3.2 Walker parameters

Assuming that the legs of a walker are all of the same length and chemical composition, we can describe a particular multivalent walker with the following parameters.

- $k \geq 2$ — The number of legs.
- $\ell > 0$ — The length of each leg.
- $C_b = \mathbb{R}^2$ — The space of coordinates in the body's reference frame, with which hip locations are defined.
- $C_e = \mathbb{R}^2$ — The space of coordinates in the environment's reference frame, with which the sites $S$ are defined.
- $\boldsymbol{H} = [\boldsymbol{h}_i \in C_b]_{i=1}^{k}$ — The vector of hip locations in the body's coordinates. Hip locations can be coincident as is the case for point-bodied walkers.
- $\boldsymbol{A} = [\boldsymbol{a}_i \in S \cup \{\odot\}]_{i=1}^{k}$ — The vector of attached feet locations a foot $i$ is either attached to a site $\boldsymbol{a}_i \in S$, or it is detached, which is represented by the symbol $\odot$.

To ensure that at most one foot is attached to any site, the set of attached legs $\boldsymbol{A}$ must obey the condition

$$(i \neq j) \wedge (\boldsymbol{a}_i = \boldsymbol{a}_j) \implies \boldsymbol{a}_i = \boldsymbol{a}_j = \odot. \tag{3.2}$$



www.manaraa.com

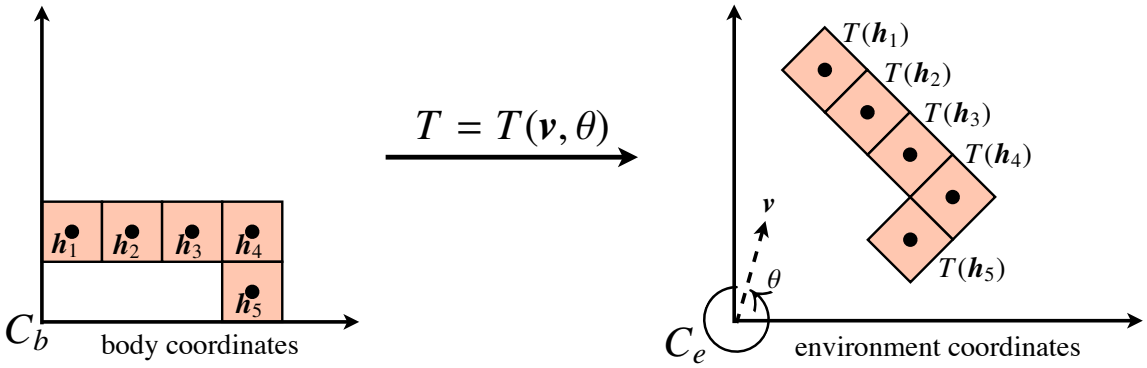


**Figure 3.2:** A 2D rigid body transform $T$ maps the body's reference frame $C_b$ to the environment's reference frame $C_e$. The transform can be specified by a rotation $\theta$ and translation $\boldsymbol{v}$.

### 3.3.3 Walker position is defined by a rigid body transform

The hip locations are in the space $C_b$ of body coordinates, while the foot locations are represented as sites in the space $S$, which has reference frame $C_e$. Therefore, we need a mapping $C_b \to C_e$ to relate these spaces. Because the body is a rigid 2D object, this mapping will be a 2D rigid body transform $T = T(\boldsymbol{v}, \theta)$ which is a rotation about the origin by angle $\theta$ followed by a translation by $\boldsymbol{v} \in \mathbb{R}^2$. The mapping $T$ will determine the location and orientation of the body in the environment space, and hence the coordinates of the hip locations in the space $C_e$ (Figure 3.2). In the simplified case of the point-bodied walkers, rotational symmetry implies the body location can be defined solely by the translation $\boldsymbol{v}$.

### 3.3.4 Walker state in the Markov process

The Markov process state encompasses only the properties of the walker that change over time; this is completely defined by $\boldsymbol{A}$, the attached state of the $k$ feet. The set of possible states for the walker is[1]

$$\mathbb{W} = (S \cup \{\odot\})^k . \tag{3.3}$$

[1]The actual space of walker states is much smaller, because the legs cannot be attached to sites that would violate the maximum length constraint. For notational simplicity we will include these impossible states in the state space, but the transition rates to these states will always be 0.



www.manaraa.com

Each of the $k$ feet can be either attached to a site in $S$ or detached ($\odot$). The state space of the entire walker system, $\Omega$, is the Cartesian product of the environment's state space $\mathbb{E}$ (Equation 3.1) and the walker's state space $\mathbb{W}$ (Equation 3.3):

$$\Omega = \mathbb{E} \times \mathbb{W} = \Sigma^S \times (S \cup \{\odot\})^k . \tag{3.4}$$

Any state $\omega = \Omega$ can be represented by the current species function, $\pi$, and the attached leg state, $\boldsymbol{A}$,

$$\omega = (\pi, \boldsymbol{A}). \tag{3.5}$$

## 3.4 State transitions

The three types of state transitions correspond to the three types of chemical reactions that can take place: binding (association), unbinding (dissociation), and catalytic transformation.[2] As discussed in Section 2.3, the interactions of a leg (L) with a substrate (S) or product (P), are given by five reactions:

$$\begin{aligned} &\mathrm{L} + \mathrm{S} \underset{k_\mathrm{S}^-}{\overset{k_\mathrm{S}^+}{\rightleftarrows}} \mathrm{LS} \xrightarrow{k_\mathrm{cat}} \mathrm{L} + \mathrm{P} \\ &\mathrm{L} + \mathrm{P} \underset{k_\mathrm{P}^-}{\overset{k_\mathrm{P}^+}{\rightleftarrows}} \mathrm{LP}. \end{aligned} \tag{3.6}$$

### 3.4.1 General leg chemistries

Beyond the substrate-product chemistry, we can model the reactions of a leg with any set of chemical species $\Sigma$ using the following functions:

[2] Some chemical systems may involve non-leg reactions at the surface sites, such as sites spontaneously cleaving, or a replenishment of substrates from solution. As long as these reactions are time-independent, they can be worked into the Markov process transition rates. We do not concern ourselves with such specialized systems, other than to note that they are possible to model without fundamentally changing the simulation strategy.



www.manaraa.com

- $k^+ : \Sigma \to \mathbb{R}^+$ — The rate of the leg binding reaction for each species.
- $k^- : \Sigma \to \mathbb{R}^+$ — The rate of the leg dissociation reaction for each species.
- $k_{\text{cat}} : \Sigma \to \mathbb{R}^+$ — The rate of leg catalysis for each species.
- $\chi : \Sigma \to \Sigma$ — The species transformation resulting from catalysis for each species.

These reaction rates determine which reactions are possible and how fast they occur. A reaction rate of 0 indicates a reaction that never occurs. If $k_{\text{cat}}(\sigma) = 0$, the leg cannot catalyze the transformation of $\sigma$ and the value of $\chi(\sigma)$ is not relevant. Restrictions on $\chi$ enforce chemically plausible models; $\chi$ is a function which enforces that there is at most one transformation a leg may induce on a particular species. To ensure that all catalytic transformations have $\Delta G<0$, the relation $\chi$ should be antisymmetric. The general set of reactions for leg $L$ can be described for each $\sigma \in \Sigma$ as

$$L + \sigma \underset{k^-(\sigma)}{\overset{k^+(\sigma)}{\rightleftarrows}} L\sigma \xrightarrow{k_{\text{cat}}(\sigma)} L + \chi(\sigma). \tag{3.7}$$

### 3.4.2 Transition rates and chemical kinetics

Dissociation ($k^-$) and catalysis ($k_{\text{cat}}$) are first-order reactions, so their rate is independent of the local environment. However, the association reactions are second-order and thus rates $k^+(\cdot)$ are actually pre-rates that when multiplied by the local concentration-dependent propensity for that reaction give a true rate. We assume that the second-order reaction of the leg associating with a chemical site is dependent on the leg colliding with the chemical site while also having enough kinetic energy to pass over the activation energy barrier $E_a$ (Section 2.1). Several factors affect this rate: the constrained diffusion of the body between reactions, the diffusion of the leg in the time between reactions, and the height of the energy barrier relative to $k_B T$.



www.manaraa.com

## 3.5 Body position distribution

In between each chemical reaction or step in the Markov process the body is assumed to be at *mechanical equilibrium*, as determined by the attached leg locations $\boldsymbol{A}$. Let random variable $\boldsymbol{B}$ represent the position of the walker's body over rigid-body transform parameters $(\boldsymbol{v}, \theta)$.

**Definition 3.5.1.** We say that a body position $\boldsymbol{b} = (\boldsymbol{v}, \theta)$ is a *feasible body position* if for each $i = 1, \ldots, k$,

$$\boldsymbol{a}_i = \odot \ \vee \ \|\boldsymbol{a}_i - T(\boldsymbol{v}, \theta)(\boldsymbol{h}_i)\| \leq \ell.$$

**Definition 3.5.2.** Let $\mathcal{F} \subset \mathbb{R}^2 \times [0, 2\pi)$ be the set of all feasible body positions. Then, for attached leg locations $\boldsymbol{A}$, we have

$$\mathcal{F} = \mathcal{F}(\boldsymbol{A}) = \left\{ (\boldsymbol{v}, \theta) \in \mathbb{R}^2 \times [0, 2\pi) \;\middle|\; \ell \geq \max_{\substack{i=1 \\ \boldsymbol{a}_i \neq \odot}}^{k} \{\|\boldsymbol{a}_i - T(\boldsymbol{v}, \theta)(\boldsymbol{h}_i)\|\} \right\}.$$

Figure 3.3 shows $\mathcal{F}$ for a point-bodied walker and the constraints implied by the attached legs. A point-bodied walker has rotational symmetry with all $\boldsymbol{h}_i = (0, 0)$, so we can describe a body position simply by a 2D vector $\boldsymbol{p}$. Segmented walkers are more complicated to illustrate as $\mathcal{F}$ is three-dimensional when the rotation $\theta$ must also be considered. For simplicity of presentation, we illustrate only the point-bodied walkers (Figure 3.1), but our model also describes the equilibrium body distribution for segmented and general walkers.

### 3.5.1 The walker body position as a Boltzmann distribution

At equilibrium, $\boldsymbol{B}$ will take on a Boltzmann distribution over $\mathcal{F}$ according to the energy $\Delta U(\boldsymbol{v}, \theta)$, so that at position $(\boldsymbol{v}, \theta)$,

$$\mathbf{P}\,[\,\boldsymbol{B} = (\boldsymbol{v}, \theta)\,] = p_{\boldsymbol{B}}(\boldsymbol{v}, \theta) = \frac{e^{-\beta \Delta U(\boldsymbol{v}, \theta)}}{\int_{\mathcal{F}} e^{-\beta \Delta U(\boldsymbol{v}, \theta)} d\boldsymbol{v} d\theta} = \frac{1}{Z} e^{-\beta \Delta U(\boldsymbol{v}, \theta)}. \tag{3.8}$$



www.manaraa.com

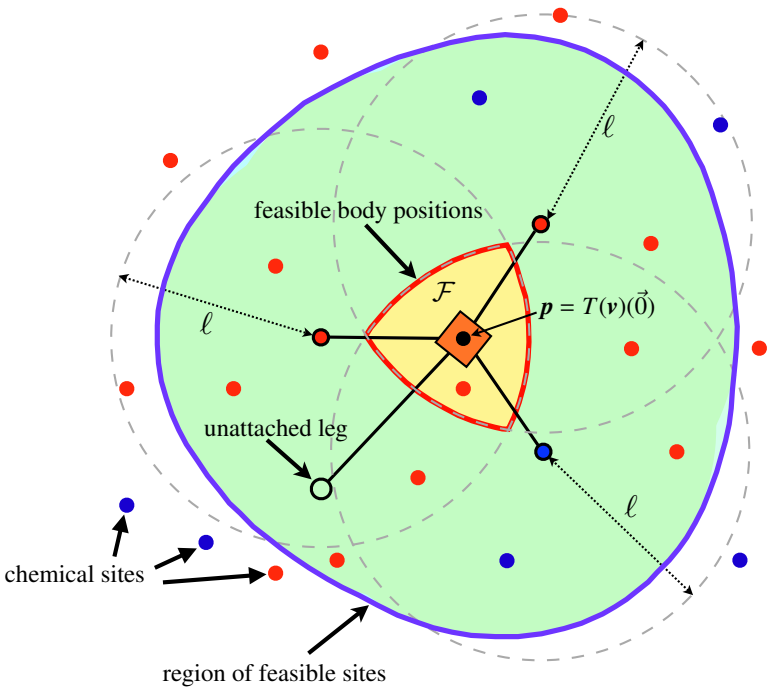


**Figure 3.3:** The feasible body positions $\mathcal{F}$ for a point-bodied walker are indicated in yellow. Each attached leg imposes a circular constraint on the body's location. Free surface sites are blue for substrates and red for products. The feasible sites $S_{\mathcal{F}}$ are all sites contained within the green region. These are the sites that can be reached by a leg of length $\ell$ from *some* body position in $\mathcal{F}$. Any site $s \notin S_{\mathcal{F}}$ has zero probability of attachment.

In Equation 3.8, $\beta = 1/k_B T$ where $k_B$ is Boltzmann's constant and $T$ is absolute temperature, so that $1/\beta$ represents the average amount of energy available at temperature $T$. We consider only isothermal systems where $T$ is fixed at 300 K, in which case $k_B T = 4.14$ pN nm, and the partition function,

$$Z = \int_{\mathcal{F}} e^{-\beta \Delta U(\boldsymbol{v},\theta)} d\boldsymbol{v} d\theta, \tag{3.9}$$

is a constant that serves only to normalize the probabilities in Equation 3.8. In general the partition function is inconvenient or impossible to compute analytically. However, as explained in Chapter 5, the Metropolis-Hastings algorithm can be used to sample from distribution $p_{\boldsymbol{B}}$ without the need to normalize and hence we can avoid the computation of $Z$ entirely.

### 3.5.2 The energy of the walker body position

For any particular chemical implementation of a multivalent random walker, an accurate determination of the walker's internal energy as a function of position, $\Delta U(\boldsymbol{v}, \theta)$, would



www.manaraa.com

require detailed structural modeling of internal degrees of freedom in the walker legs and body. To keep our model general and simple, we do not attempt such detailed analysis. In fact, $\Delta U(\boldsymbol{v}, \theta)$ can be treated as a free parameter, and there are several natural choices that lead to a range of different body distributions, but that maintain our assumptions of minimal internal structure and mechanical coordination between the legs.

**Uniform body distribution.** Assuming the legs are totally uncoupled and free from internal structure, the energy should be uniform across all feasible positions,

$$\Delta U(\boldsymbol{v}, \theta) = \begin{cases} 0 & (\boldsymbol{v}, \theta) \in \mathcal{F} \\ \infty & \text{otherwise} \end{cases}. \tag{3.10}$$

The uniform body distribution models the minimal mechanical coupling between legs. Only the restriction of finite leg length constrains the motion of the legs, and the body's constrained diffusion moves with equal likelihood over all feasible locations. We use the uniform body distribution for results in Chapter 6 to show that even with the weakest possible coordination constraints between legs, multivalent random walkers can still function as molecular motors.

**Elastic legs body distribution.** A more restrictive model assumes that legs are elastic and the energy of a body position is the sum of the squared leg lengths from the hip joints to the attached feet:

$$\Delta U(\boldsymbol{v}, \theta) = \begin{cases} \sum_{\substack{i=1 \\ \boldsymbol{a}_i \neq \odot}}^{k} \mu \|\boldsymbol{a}_i - T(\boldsymbol{v}, \theta)(\boldsymbol{h}_i)\|^2 & (\boldsymbol{v}, \theta) \in \mathcal{F} \\ \infty & \text{otherwise} \end{cases}. \tag{3.11}$$

The free parameter $\mu$ in Equation 3.11 is the spring rate of the legs. When $\mu > 0$, an energy minimum and therefore probability maximum forms around the position that minimizes squared leg length. At high values of $\mu$, $\Delta U$ will be less affected by external forces than walkers with uniform body distribution based on Equation 3.10.



www.manaraa.com

**Deterministic body position—elastic spring model.** As free parameter $\mu \to \infty$ in Equation 3.11, the body's distribution becomes a delta function at the minimal feasible body position $(\boldsymbol{v}^*, \theta^*)$, where

$$(\boldsymbol{v}^*, \theta^*) = \min_{(\boldsymbol{v},\theta)\in\mathcal{F}} \sum_{\substack{i=1 \\ \boldsymbol{a}_i \neq \odot}}^{k} \|\boldsymbol{a}_i - T(\boldsymbol{v}, \theta)(\boldsymbol{h}_i)\|^2. \tag{3.12}$$

## 3.6 Leg–site interactions

The bimolecular kinetics of leg–site binding is controlled by two factors: (I) a second-order process by which the leg and the site come into contact with each other, and (II) a first-order process wherein the leg and the site undergo conformational changes to move to a strongly bound state [69]. Process I is controlled by the constrained diffusion of the body and the unattached legs, while process II is controlled by the activation energy barrier of the reaction as described in Section 2.1. Depending on which of these processes is rate-limiting, there are two different types of kinetics for the leg–site binding reactions.

**Diffusion-limited kinetics.** If the reaction energy barrier is low, process I is limiting; the leg is likely to react with one of the first few substrates it comes in contact with. Thus, the leg will be more likely to react with sites closer to where it had previously been attached. Because the diffusion to new sites is the limiting factor in the reaction this situation is called *diffusion-limited*.

**Reaction-limited kinetics.** However, if the energy barrier is high, process II is limiting; the leg and substrate have to move through one or more weakly-bound conformational states before they enter the low-energy conformation of the strongly-bound state. This situation is called *reaction-limited*. The leg will diffuse around the local environment



www.manaraa.com

of sites, encountering and interacting weakly with many sites until by chance it spends enough time at a high enough energy near a certain site to react.

For the MVRW model the reaction-limited kinetics are simultaneously more practical from a computational viewpoint, and more realistic from a biophysical viewpoint. When kinetics are reaction-limited, they are slow enough to allow the separation of timescales between the much faster mechanical motion and the slower chemical reactions. The legs, like the body, reach a mechanical equilibrium before they react, and unlike in the case of diffusion-limited reactions, the leg's action is Markovian—independent of the previous state of that leg. Thus, for the computationally important reasons of maintaining the Markovian property and avoiding simulation of the constrained leg diffusion, we assume reaction-limited kinetics in the MVRW model. Reaction-limited kinetics are also a biophysically plausible assumption, as deoxyribozyme binding kinetics depend on the DNA polymer weakly binding at multiple base pairs in order to form a strong enough bond to remain attached, which means that a deoxyribozyme leg is likely to interact with many local substrates before it finally attaches strongly.

### 3.6.1 Modeling reaction limited leg binding

The reaction-limited kinetic model implies that the rate of the bimolecular reaction of the leg binding to a site should be proportional to the number of leg–site pairs that may potentially interact and bind [49]. Consider the case of the constrained motion of walker leg $i$ as it moves about hip location $\boldsymbol{h}_i$ while the walker body is fixed at position $\boldsymbol{b} = (\boldsymbol{v}, \theta) \in \mathcal{F}$. A leg cannot reach sites farther than distance $\ell$, but any site closer than $\ell$ is a potential candidate for attachment.



www.manaraa.com

## Binding kinetics from a fixed body position.

A site $\boldsymbol{s}$ is feasible for leg $i$ from body position $\boldsymbol{b}$ if it is not already occupied ($\boldsymbol{s} \notin \boldsymbol{A}$) and

$$\|\boldsymbol{s} - T(\boldsymbol{b})(\boldsymbol{h}_i)\| < \ell. \tag{3.13}$$

**Definition 3.6.1.** The *feasibility indicator function* for leg $i$ binding to site $\boldsymbol{s} \notin \boldsymbol{A}$ from fixed body position $\boldsymbol{b}$, is

$$I_{\boldsymbol{b}}^{i}(\boldsymbol{s}) = \begin{cases} 1 & \|\boldsymbol{s} - T(\boldsymbol{b})(\boldsymbol{h}_i)\| < \ell \\ 0 & \text{otherwise} \end{cases}.$$

From the reaction-limited kinetic viewpoint, the feasibility indicator $I$ determines if leg $i$ with hip location $\boldsymbol{h}_i$ is close enough to unoccupied site $\boldsymbol{s}$ that the leg and site may potentially interact while the leg is diffusing in its constrained environment. The attachment reaction rate for site $\boldsymbol{s}$ depends on the species $\pi(\boldsymbol{s}) \in \Sigma$ at the site and the associated kinetic rate $k^{+}_{\pi(\boldsymbol{s})}$. From fixed body position $\boldsymbol{b} = (\boldsymbol{v}, \theta)$ the attachment rate is

$$r_{\boldsymbol{b}}^{i}(\boldsymbol{s}) = k^{+}_{\pi(\boldsymbol{s})} I_{\boldsymbol{b}}^{i}(\boldsymbol{s}). \tag{3.14}$$

Equation 3.14 implies the total rate of a single leg binding to any site, from position $\boldsymbol{b}$, is

$$R_{\boldsymbol{b}}^{i} = \sum_{\boldsymbol{s} \in S} r_{\boldsymbol{b}}^{i}(\boldsymbol{s}). \tag{3.15}$$

For reaction-limited kinetics, $R_{\boldsymbol{b}}^{i}$ is proportional to the total number of feasible sites.

## Binding kinetics at mechanical equilibrium.

Now we take into account the assumption that in between reaction steps the body is not at a fixed position $\boldsymbol{b}$, but in an equilibrium distribution $\boldsymbol{B}$ over positions.

**Definition 3.6.2.** We integrate the feasibility indicator function (Definition 3.6.1) over the probability distribution $\boldsymbol{B}$ to define the *feasibility probability* of site $\boldsymbol{s} \notin \boldsymbol{A}$ for leg $i$:

$$f_{\boldsymbol{B}}^{i}(\boldsymbol{s}) = \int_{\mathcal{F}} p_{\boldsymbol{B}}(\boldsymbol{v}, \theta) I_{(\boldsymbol{v},\theta)}^{i}(\boldsymbol{s}) \, d\boldsymbol{v} \, d\theta.$$



www.manaraa.com

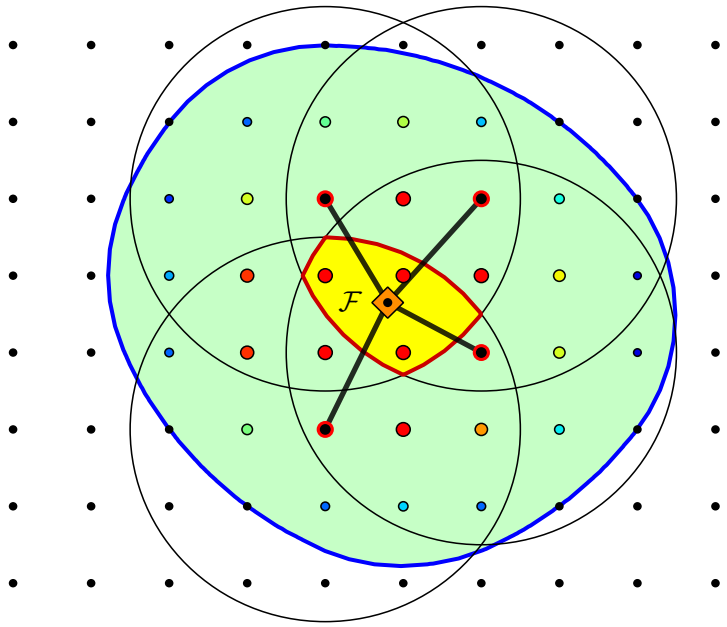


**Figure 3.4:** The function $f_{\boldsymbol{B}}^{i}(\boldsymbol{s})$ represents the probability that a leg $i$ is close enough to site $\boldsymbol{s}$ to react, given the body's position as random variable $\boldsymbol{B}$. For simplicity, we illustrate this with point-bodied walkers, so leg index $i$ is not necessary to specify. Here we assume $\boldsymbol{B}$ is uniform over $\mathcal{F}$, shown in yellow. The region of feasible sites is shown in green; the set of unattached sites within this region is $S_{\mathcal{F}}$. The color and size of each unattached site shows $f_{\boldsymbol{B}}(\boldsymbol{s})$ for that site. Red sites have feasibility probability 1, cooler colors are less probable, and black sites have probability 0.

The feasibility probability $f_{\boldsymbol{B}}^{i}(\boldsymbol{s})$ is a pre-rate for site $\boldsymbol{s}$. It is bounded, $0 \leq f_{\boldsymbol{B}}^{i}(\boldsymbol{s}) \leq 1$, so we interpret it as the probability that a leg is close enough to site $\boldsymbol{s}$ to feasibly bind at mechanical equilibrium (Figure 3.4).

**Definition 3.6.3.** The feasibility probability combined with Equation 3.14 allows us to define the *attachment transition rate* for leg $i$ attaching to site $\boldsymbol{s}$, given body distribution $\boldsymbol{B}$:

$$r_{\boldsymbol{B}}^{i}(\boldsymbol{s}) = k_{\pi(\boldsymbol{s})}^{+} f_{\boldsymbol{B}}^{i}(\boldsymbol{s}).$$



www.manaraa.com

### 3.6.2 The set of feasible sites

Any site with non-zero rate of attachment is called a *feasible site*; the region of feasible sites is shown in green in Figures 3.3 and 3.4. A site is feasible for leg $i$ if it is within distance $\ell$ of $\boldsymbol{h}_i$ in *some* body position in $\boldsymbol{b} = (\boldsymbol{v}, \theta) \in \mathcal{F}$ (Equation 3.13). Let the distance of leg $i$ from a site $\boldsymbol{s}$ under feasible body positions $\mathcal{F}$ be

$$d^i(\boldsymbol{s}, \mathcal{F}) = \min_{\boldsymbol{b} \in \mathcal{F}} \{\|\boldsymbol{s} - T(\boldsymbol{b})(\boldsymbol{h}_i)\|\}. \tag{3.16}$$

**Definition 3.6.4.** Given the set of attached leg locations $\boldsymbol{A}$, we define the *set of feasible sites* for leg $i$ as

$$S^i_{\mathcal{F}} = \{\boldsymbol{s} \in S \setminus \boldsymbol{A} \mid d^i(\boldsymbol{s}, \mathcal{F}) \leq \ell\}.$$

Knowing $r^i_{\boldsymbol{B}}(\boldsymbol{s})$ for each site in $S^i_{\mathcal{F}}$ for each unattached leg $i$ gives the transition rates for all association reactions.[3] Together with the much simpler rates for the unimolecular dissociation and cleavage reactions, which are independent of body and leg diffusion, this enables us to model all of the reactions that lead to state transitions in the model.

## 3.7 Effect of external load forces

If the walker body is subjected to an external load force, either as the result of the viscous drag of a cargo or as the result of a measurement apparatus, such as those used to probe kinesin [119], the walker can do work by moving in opposition to the force. We assume the force is applied to a cargo tether on the walker's body at location $\boldsymbol{c}$ in body coordinates $C_b$. The change in potential energy of the walker when moving from the original position $\boldsymbol{b}_0 = (\boldsymbol{v}_0, \theta_0)$ to a new position $\boldsymbol{b} = (\boldsymbol{v}, \theta)$ under the load of conservative force $\boldsymbol{f}$ is

$$\Delta E_f(\boldsymbol{b}) = \Delta E(\boldsymbol{b}) = -\boldsymbol{f} \cdot (T(\boldsymbol{b})(\boldsymbol{c}) - T(\boldsymbol{b}_0)(\boldsymbol{c})). \tag{3.17}$$

[3] In Section 4.2 we specialize these kinetic rates to the point-bodied walkers, where all $k$ legs are identical and body rotation $\theta$ is not necessary to consider.



www.manaraa.com

When $\Delta E > 0$ the walker is doing work moving against the force; when $\Delta E < 0$ the force is doing work on the walker.

The MVRW model can capture the effect of a conservative load force on the walker body through the effect on the body's energy function $\Delta U$. The new energy of position $\boldsymbol{b} = (\boldsymbol{v}, \theta)$, under force $\boldsymbol{f}$ is

$$\Delta U_f(\boldsymbol{b}) = \Delta U_0(\boldsymbol{b}) + \Delta E_f(\boldsymbol{b}). \tag{3.18}$$

In Equation 3.18, $\Delta U_0(\boldsymbol{b})$ represents the energy under zero force, i.e., from Equation 3.10. The effect of force on a point-bodied walker is shown in Figure 3.5.



www.manaraa.com

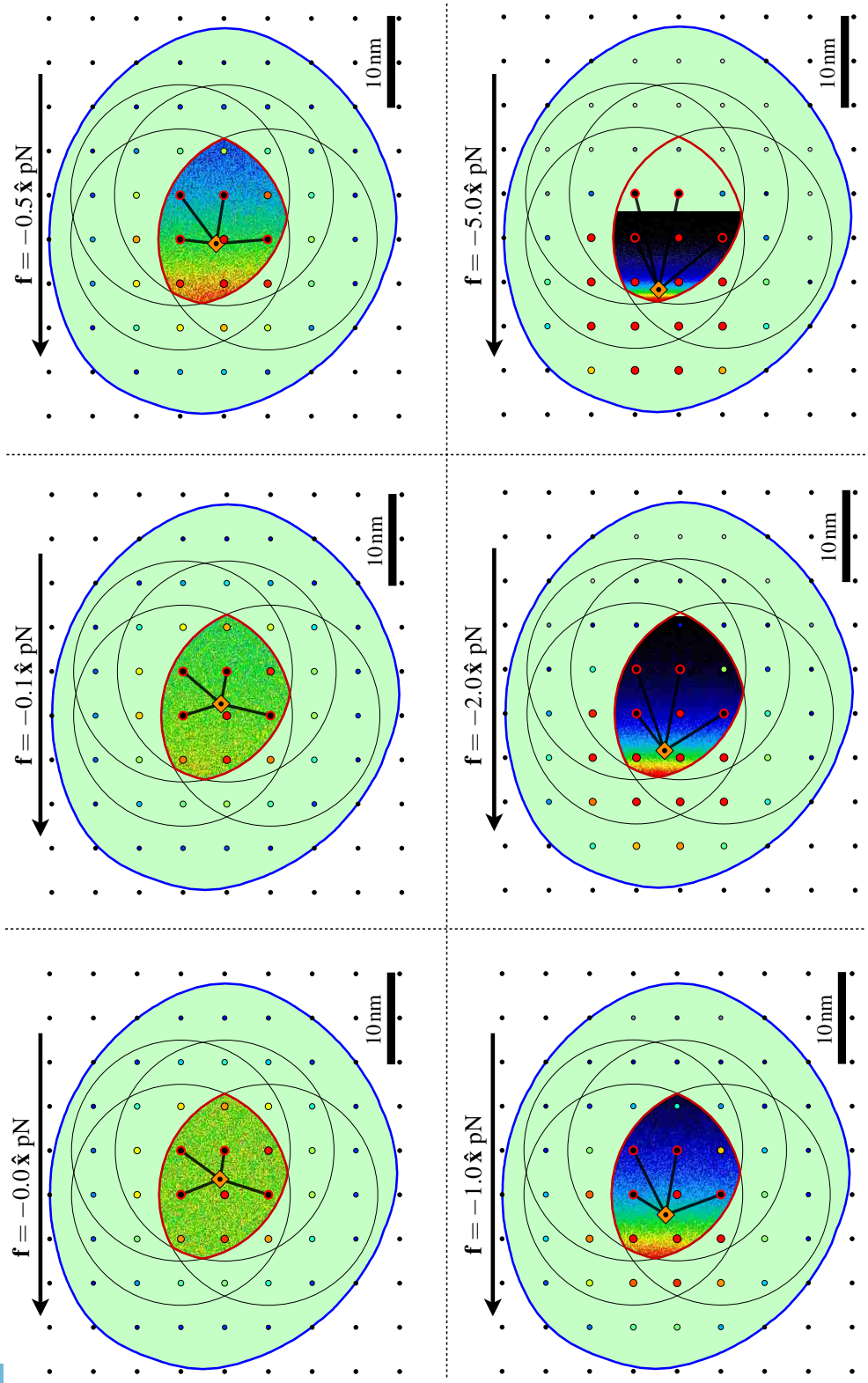


**Figure 3.5:** A conservative load force $f$ is applied in the $-\hat{\mathbf{x}}$ direction to a point-bodied walker with cargo attachment point $\boldsymbol{c} = \vec{0}$. The body's equilibrium distribution $p_{\boldsymbol{B}}$ is displayed as a 2D histogram over $\mathcal{F}$. Warmer colors represent higher probability. The region of feasible sites is drawn in green, and the feasibility of each site, $f_{\boldsymbol{B}}(\boldsymbol{s})$, is shown by the color and size of the site. The body is drawn at its mean equilibrium location, $\langle \boldsymbol{B} \rangle$.



www.manaraa.com

## 3.8 The state space of multivalent random walker systems

The MVRW model has many irreversible state transitions, specifically those that are represented by a catalyzed transformation of a chemical site from species $\pi(s)$ to species $\chi(\pi(s))$. The presence of irreversible transitions has implications for the nature of the CTMP underlying the model.

In a Markov process, a state $j$ is *accessible* from state $i$ if there is a time $t$ such that

$$\mathbf{P}\left[\, X(t) = j \,|\, X(0) = i \,\right] > 0.$$

We can write this $i \to j$. Two states *communicate* if $i \to j$ and $j \to i$, which we can write as $i \leftrightarrow j$. Furthermore, a state $i$ is said to be *transient* if there is a non-zero probability of never returning to state $i$ again, otherwise it is called *recurrent*; it is *positive recurrent* if the expected time to return is finite.

We can use these definitions to examine the state space $\Omega = \mathbb{E} \times \mathbb{W}$. Consider the example of the walkers using the substrate/product chemistry (Equation 3.6) where $\Sigma = \{\mathrm{S}, \mathrm{P}\}$. Initially all sites are substrates, but over time substrates are cleaved, resulting in a net decrease in $\Delta G$. If there is a finite number of sites, eventually all sites will be cleaved to products, and the system will be at an energy minimum. No further cleavages are possible. We can divide the states into equivalence classes based on their free energy. If the initial energy is 0, and a cleavage of a substrate to a product decreases the energy by 1 unit and there, are $n$ sites in total, then the energy equivalence classes are $\mathbb{E}_0, \mathbb{E}_{-1}, \ldots, \mathbb{E}_{-n}$. Each $\mathbb{E}_{-i}$ contains all states of the walker and environment where exactly $i$ sites have been cleaved. Sites in different energy classes do not communicate since the energy-releasing reactions are irreversible.

We can further classify the states by looking at the walker's state space. Let $\mathbb{W}_P$ be all the states of the system where the sites in $P \subseteq S$ are products, and all other sites are substrates. The states $\mathbb{W}_P$ correspond to the walker moving without cleaving any sites. Thus, the states in $\mathbb{W}_P$ all communicate. However, if site $s$ is then cleaved, the system will



www.manaraa.com

make an irreversible transition from a state in $\mathbb{W}_P$ to a state in $\mathbb{W}_{P\cup\{s\}}$, and will never be able to return to any state in $\mathbb{W}_P$. Figure 3.6 illustrates this state space structure. Each $\mathbb{W}_P$ contains the same number of states and they have the same connectivity amongst themselves; however, the rates of transitions will be different because of the different species at the sites.

All states except for those in $\mathbb{E}_{-n}$ are transient because of the irreversibility of substrate catalysis. These considerations are important if one is interested in equilibria. A Markov process will have a unique equilibrium distribution over the recurrent states. This will be over all the states in $\mathbb{E}_{-|S|} = \mathbb{W}_S$, the set of all states where all sites have been cleaved to a product. This is the same distribution as for a walker system that starts in an all-product environment. This equilibrium corresponds to uninteresting, diffusive motion. Therefore, we see that for any finite environment our walkers are eventually doomed to use up their store of chemical energy and enter a diffusive type of behavior. Any interesting, non-diffusive behavior must therefore occur as the walker moves through the high-energy states on its way to equilibrium. After all, it is really only at the irreversible parts of the process that any control over the system evolution can occur. This point will remind us that while the long-term behavior of the system will become diffusive, there is still a potential for useful, non-diffusive motion for shorter times where not much of the energy in the chemical sites has been expended.

In Chapter 7, we revisit these concepts in the case of walkers moving over semi-infinite 1D tracks of substrates. While these infinite state spaces have unlimited substrate chemical free energy, it is a locally-limited resource, and we argue that any such system that moves less than ballistically is doomed to consume its local substrate reserves near the origin, and eventually move as if it is also in a zero-free-energy, all-product environment.



www.manaraa.com

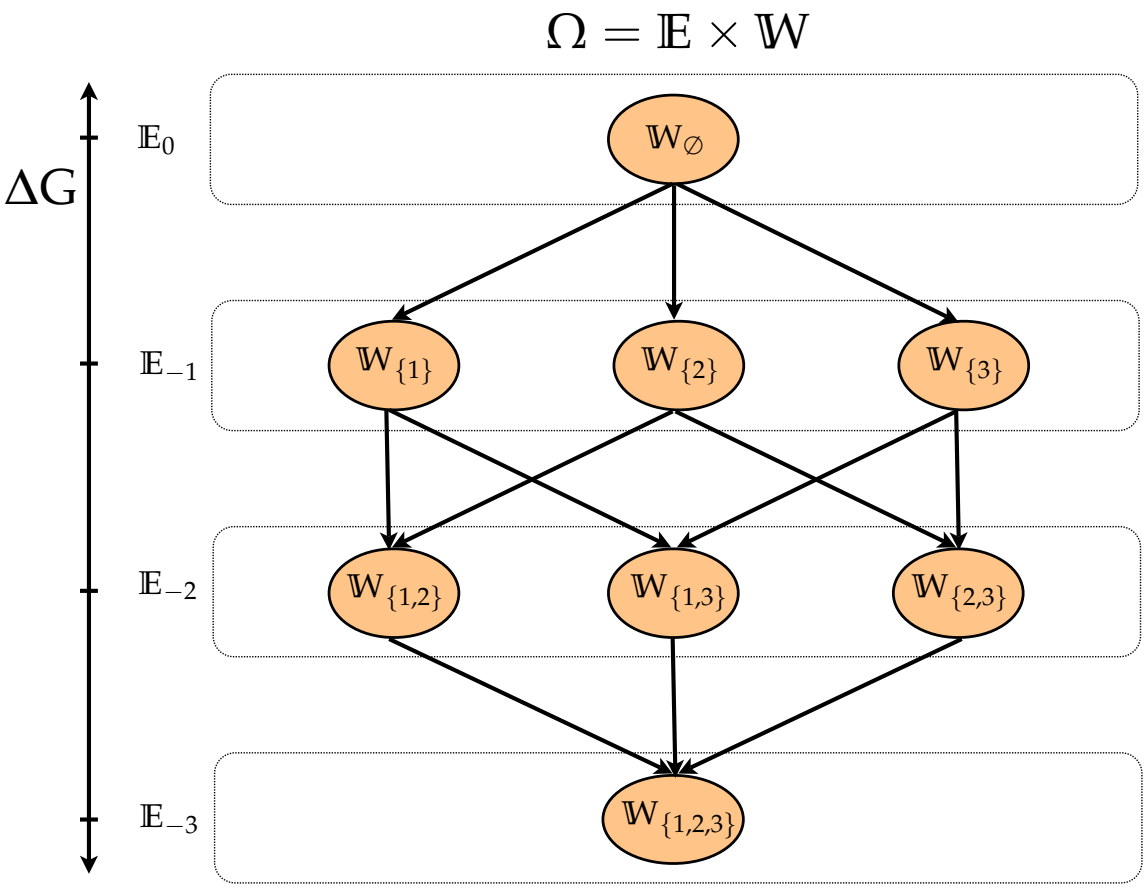


**Figure 3.6:** The state space of a walker system with an environment of three sites, $S = \{1, 2, 3\}$. Irreversible changes to sites in the environment create a partial order on equivalence classes of the state space. Each $\mathbb{W}_*$ represents the equivalence class of states with a fixed environment and the internal transitions which correspond to legs binding and unbinding are not shown. Each equivalence class $\mathbb{E}_*$ is all states with the same free energy $\Delta G$. The unique equilibrium distribution of the walker Markov process is over the recurrent states $\mathbb{W}_{\{1,2,3\}}$; all other states are transient.



www.manaraa.com

# Chapter 4
# Kinetic Monte Carlo Simulations

The MVRW continuous-time Markov process as developed in Chapter 3 can be directly simulated using the kinetic Monte Carlo (KMC) method. Section 4.1 presents a generic description of the KMC method, which uses the Markov process transition rates to iteratively simulate a single trace of a walker moving through a series of discrete states, separated by exponentially distributed step times $\tau$. Kinetic Monte Carlo is not a prepackaged algorithm, but rather a general numerical technique. The mechanism by which the next step and step time are chosen is simple; the actual difficulty in simulating the multivalent random walkers is the enumeration of all potential attachment transitions $S^i_{\mathcal{F}}$ (Definition 3.6.4) and the calculation of the transition rates $r^i_{\boldsymbol{B}}(\boldsymbol{s})$ (Definition 3.6.3). In the worst case, the Metropolis sampler would need to be run on every KMC step in order to get a sample from $\boldsymbol{B}$ which is necessary for the computation of $r^i_{\boldsymbol{B}}(\boldsymbol{s})$.

However, there are opportunities for optimization if we focus on the specialization of the general MVRW model to the specific case of spider-like point-bodied walkers with deoxyribozyme substrate/product kinetics. Section 4.2 shows that the rotational symmetry of point-bodied spiders reduces the computation of the set of feasible body positions, $\mathcal{F}$, and the set of feasible sites, $S_{\mathcal{F}}$, to the problem of finding the intersection of 2D balls (discs) of radius $\ell$. Furthermore, in Section 4.3 we show that when a point-bodied walker moves over surfaces on which the chemical sites are arranged as a regular rectangular lattice, the translational symmetry leads to a restricted set of possible *canonical* leg configurations, for which the computationally expensive values of $S_{\mathcal{F}}$ and $r_{\boldsymbol{B}}(\boldsymbol{s})$ can be precomputed using a single set of samples for each unique $\boldsymbol{B}$ up to translation. In Section 4.4 we explain how the MVRW kinetic Monte Carlo algorithm uses the precomputed values to efficiently simulate the motion of point-bodied walkers on regular lattices of sites.



www.manaraa.com

## 4.1 Simulation of continuous-time Markov processes

Models based on CTMPs are ubiquitous in statistical physics, and many physical processes at the nanoscale are modeled well by the Markovian paradigm. Therefore, there has been extensive work in the statistical physics community exploring efficient methods—both exact and approximate—of numerically sampling trajectories of CTMPs through simulation. The same is also the case for the newer field of stochastic chemical kinetics using the Gillespie approach. Our MVRW simulation algorithms build upon the methods used in these fields.

The approach of numerically simulating Markov chains was developed by Metropolis et al. in 1953 at Los Alamos laboratories [87]. Metropolis used a Monte Carlo approach to numerically sample from the equilibrium distribution of CTMPs describing physical systems. Over the following decades, physicists started using Monte Carlo approaches to study the kinetics of non-equilibrium systems. These approaches became collectively known as kinetic Monte Carlo (KMC) methods by the 1990's [124].

There are two major classes of KMC algorithms: those that use rejection sampling, and the modern rejection-free methods. The efficiency of rejection-based methods is highly dependent on the relative ratios of transition rates, while the rejection-free methods are independent of rate ratios. Rejection-free approaches were introduced in the physics community in 1975 by Bortz, Kalos, and Libowitz [18] as what is now known as the *BKL algorithm*. They were introduced in the stochastic chemistry community in 1976 with Gillespie's *stochastic simulation algorithm (SSA)* [48]. Most modern approaches draw from these works.

### 4.1.1 The generalized kinetic Monte Carlo method

The basic idea of the KMC method is to iteratively simulate a sample trajectory of the CTMP, starting at some fixed initial state and evolving the system state and time stochas-



www.manaraa.com

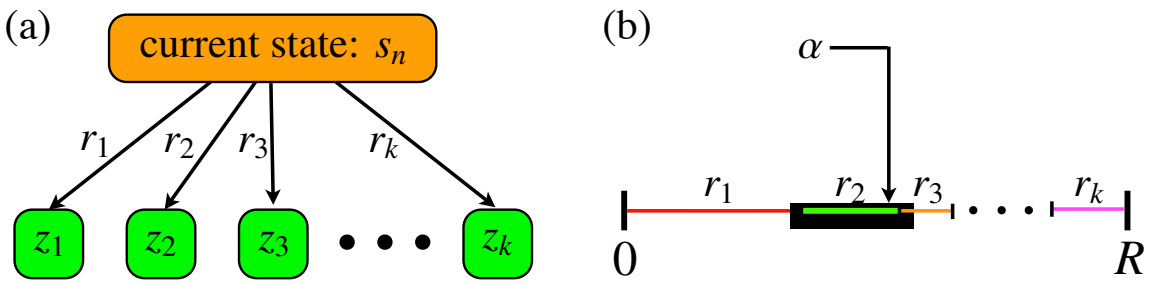


**Figure 4.1:** (a) At step $n$ of the KMC algorithm, the system is in state $s_n$, and we must choose $s_{n+1}$ from amongst the $k$ possible next states $\{z_i\}_{i=1}^k$ according to their respective transition rates $\{r_i\}_{i=1}^k$. (b) We can select the next state using a single random number $\alpha \sim \text{Uniform}(0, R)$, where $R = \sum r_i$ is the total rate. This example shows the next state chosen to be $z_2$.

tically with probabilities derived from the transition rates of the model. The result is a single sample trace of a system. Many traces can be grouped as an ensemble to estimate the state distribution of the process as it evolves in time.

Consider a CTMP $\{X(t)\}$ defined over a countable state space $S = \{s_i\}$. Associated with this process is a transition rate function $Q : S \times S \to \mathbb{R}^+$, where $Q(s_i \to s_j)$ gives the transition rate from state $s_i$ to state $s_j$. Given an initial start state, $s_0$, the KMC algorithm evolves the system state through time, stochastically choosing a next state and an elapsed time. The result is a function $x : \mathbb{R}^+ \to S$, where $x(t)$ was the state of the simulation at time $t$, which represents a *realization* or *trace* of the Markov process. More concretely, we expect $x(t) \sim X(t)$ for all $t \geq 0$.

The core of the KMC algorithm is a procedure for determining the next transition and the transition time starting from a given state. The state of the system is $s_0$ at time $t_0 = 0$. If the KMC algorithm is run for $N$ steps, we produce a sequence $\{s_i\}_{i=0}^N$ of states and a sequence $\{t_i\}_{i=0}^N$ of times when the process enters those states; together these define the realization $x(t)$.

After the $n$-th step of the algorithm, the system will be in state $s_n$ at time $t_n$. The task of the KMC algorithm is to stochastically choose $s_{n+1}$ and $t_{n+1}$ according to the Markov



www.manaraa.com

process's transition rates. Consider the set of transitions from state $s_n$ with positive rate,

$$Z = \{s' \in S \mid Q(s_n \to s') > 0\}. \tag{4.1}$$

We assume that $|Z| = k$ is finite, and thus we can enumerate it as $Z = \{z_i\}_{i=1}^k$. The transition rates are $\{r_i\}_{i=1}^k$, with $r_i = Q(s_n \to z_i)$. Let the total rate of all transitions be $R = \sum_{i=1}^k r_i$. This situation is illustrated in Figure 4.1a.

The probability of the process moving to state $z_i$ at time $n+1$ is given by the ratio $r_i/R$. We can choose a next state $z^\star \in Z$ by selecting a random number $\alpha \sim \text{Uniform}(0, R)$ and choosing $z^\star = z_j$, where $j$ is the smallest integer satisfying $\sum_{i=1}^j r_i > \alpha$. This process is depicted in Figure 4.1b.

Finally, we must choose how much time will elapse until the transition to $z^\star$. From our current state, all of the possible transitions in $Z$ occur stochastically with constant rate per unit time. Thus, the time $\tau_i$ until the transition to $z_i$ will be exponentially distributed: $\tau_i \sim \text{Exp}(r_i)$. We are interested only in the probability distribution for the minimum of these variables, $\tau^\star = \min\{\tau_1, \ldots, \tau_k\}$. The exponential distribution has the convenient property that $\tau^\star$ will also be exponentially distributed:

$$\begin{aligned}
\mathbf{P}[\,\tau^\star > t\,] &= \mathbf{P}[\,\min\{\tau_1, \ldots, \tau_k\} > t\,] \\
&= \mathbf{P}\left[\,\bigwedge_{i=1}^k \tau_i > t\,\right] \\
&= \prod_{i=1}^k \mathbf{P}[\,\tau_i > t\,] \\
&= \prod_{i=1}^k e^{-tr_i} \\
&= e^{-t\sum r_i} = e^{-tR}.
\end{aligned} \tag{4.2}$$

Thus, we see that $\tau^\star \sim \text{Exp}(R)$, and we only need to sample a single exponential distribution to find the minimum time. The exponential distribution is also particularly easy to



www.manaraa.com

sample from; given $\beta \sim \text{Uniform}(0, 1)$, we can find $\tau^\star$ as

$$\tau^\star = \frac{-\ln \beta}{R}. \tag{4.3}$$

At this point the KMC algorithm records the next state $s_{n+1} = z^\star$ and the new time $t_{n+1} = t_n + \tau^\star$, and the process repeats until $N$ simulation steps have been made.

### 4.1.2 Efficiency of KMC methods

Typically, the KMC method is used to study long-term behavior of stochastic processes, so the number of steps, $N$, is quite large. Additionally, most statistical analyses will require an ensemble average over many simulation runs. The iterative KMC algorithm requires $\Omega(N)$ computation time as there is no general way around simulating each step of the process individually. The computational efficiency is thus normally considered per step.

While the mathematical description of the Markov process helps to formally define the state space and transition function, simulations can be more efficient and practical regarding the representation of states and state transitions as long as they remain equivalent to the mathematical description. In a computer simulation of an abstractly defined Markov process, the state space is rarely represented using the exact same variables that are used to describe it mathematically. The transitions and rates are computed as needed on each iteration, which is an application-specific task and often the most expensive part of a KMC step. This is certainly the case in the MVRW model.

Many clever optimizations of the KMC algorithm are possible for certain classes of CTMPs. Often the number $k$ of transitions from any state is very large and the process of selecting a next step from amongst the $k$ choices weighted by rates is a significant contribution to the run time. The standard algorithm (Figure 4.1) is $\mathcal{O}(k)$, as it traverses an unordered list of partial sums of the transition rates to find the selected transition, $z^\star$. If each new state brings an entirely new list of potential reactions and rates, one could not expect to do much better than this as the cost of inserting the transition rates into any data



www.manaraa.com

structure will still be $\Omega(k)$. However, many CTMPs have a certain amount of symmetry in their transition rate matrix, where neighboring states have nearly identical transitions and rates. Simulations of these Markov processes can be more clever in their choice of data structures. If the data structure allows fast searches and efficient updates of those few transition rates that do change then there is a possibility for the run time of single KMC steps to be $\mathcal{O}(\ln k)$ [47], or even $\mathcal{O}(1)$ [111, 115].

The CTMP describing the MVRW model does not have this special homogeneity property, because state transitions in the MVRW model correspond to leg attachment or detachment events and each of these two types of events changes the physical restrictions on the walker's position. At each step, the spider has a new equilibrium position distribution which causes nearly all the possible transitions and rates to change. This is not a major concern as the total number of transitions from any one state in the MVRW model is small, bounded by some function of the number of legs, leg length, and chemical site density. Therefore, even the simple unordered search of transition rates is sufficient for our purposes. We revisit the issue of computation of transition rates in Section 4.4 in the simplified context of point walkers (Section 4.2) moving over regular lattices (Section 4.3), where translational symmetry leads to opportunities for precomputation.

## 4.2 Simulation of point-bodied walkers

Consider a point-bodied walker with $k$ legs of length $\ell$ that moves over sites arranged on a regular lattice under the substrate/product kinetics of Equation 3.6, where $\Sigma = \{S, P\}$. We assume that all sites $S$ are initially substrates, but the action of the walker modifies sites $P \subset S$ to products.



www.manaraa.com

### 4.2.1 State representation

A state $\omega \in \Omega$ of this simplified MVRW Markov process can be described as

$$\omega = (P, A). \tag{4.4}$$

Equation 4.4 represents both the state of the surface and the state of the walker, and is a specialization of the general state $\omega = (\pi, \boldsymbol{A})$ as given by Equation 3.5. Since all sites are initially substrates, the set of sites that have been transformed into products, $P \subset S$, completely defines the species function for site $\boldsymbol{s} \in S$, with

$$\pi(\boldsymbol{s}) = \begin{cases} \mathrm{P} & \boldsymbol{s} \in P \\ \mathrm{S} & \text{otherwise} \end{cases}.$$

The identical nature of the $k$ legs for point-bodied walkers lets us write the vector of attached legs $\boldsymbol{A} = A$, where $A$ is now the (unordered) set of attached sites. It should obey the restrictions

$$A \subset S, \tag{4.5}$$

$$|A| \leq k, \text{ and} \tag{4.6}$$

$$\mathcal{F}(A) \neq \emptyset. \tag{4.7}$$

### 4.2.2 Feasible body positions

Instead of defining walker body position with the general 2D rigid body transform $T(\boldsymbol{v}, \theta)$, a point-bodied walker's position can be defined solely by its coordinates in 2D space, $\boldsymbol{p} \in \mathbb{R}^2$, which is equivalent to translating the hip coordinates $\boldsymbol{h} = (0, 0)$ by $\boldsymbol{p}$:

$$T(\boldsymbol{p})(\boldsymbol{h}) = \boldsymbol{h} + \boldsymbol{p} = (0, 0) + \boldsymbol{p} = \boldsymbol{p}.$$

Definition 3.5.2 gives feasible body positions for general walkers, but we can specialize this to point-bodied walkers, and for attached sites $A$ define

$$\mathcal{F} = \mathcal{F}(A) = \left\{ \boldsymbol{p} \in \mathbb{R}^2 \;\middle|\; \ell \geq \max_{\boldsymbol{a} \in A} \{ \|\boldsymbol{a} - \boldsymbol{p}\| \} \right\}. \tag{4.8}$$



www.manaraa.com

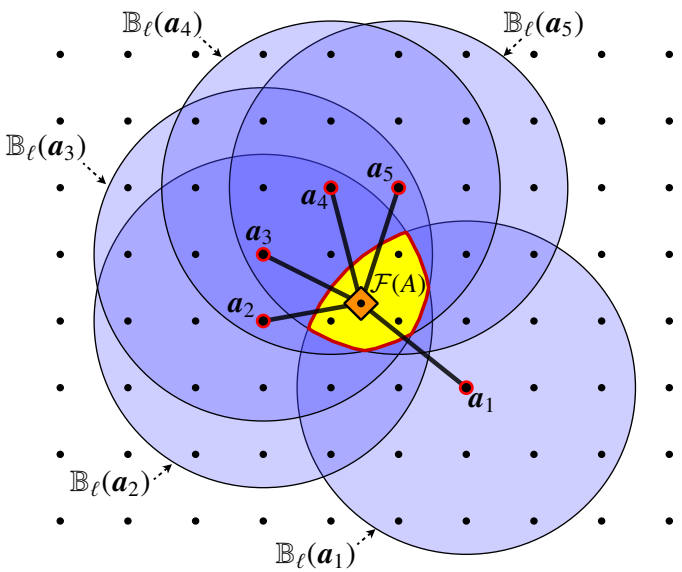


**Figure 4.2:** The feasible body positions $\mathcal{F}(A)$ for a point-bodied walker are indicated in yellow. Each attached leg $\boldsymbol{a}_i$ imposes a circular constraint on the body's location, so that $\mathcal{F} = \cap_{\boldsymbol{a}\in A}\mathbb{B}_\ell(\boldsymbol{a})$.

**Definition 4.2.1.** The (closed) *2D ball* around location $\boldsymbol{s}$ of radius $\ell$ is

$$\mathbb{B}_\ell(\boldsymbol{s}) = \left\{\boldsymbol{p} \in \mathbb{R}^2 \;\middle|\; \ell \geq \|\boldsymbol{s} - \boldsymbol{p}\|\right\}.$$

Thus, we have the equivalence

$$\boldsymbol{p} \in \mathbb{B}_\ell(\boldsymbol{s}) \iff \ell \geq \|\boldsymbol{s} - \boldsymbol{p}\|. \tag{4.9}$$

Using Equation 4.9 to simplify Equation 4.8, we find

$$\begin{aligned}
\mathcal{F} = \mathcal{F}(A) &= \left\{\boldsymbol{p} \in \mathbb{R}^2 \;\middle|\; \ell \geq \max_{\boldsymbol{a}\in A}\{\|\boldsymbol{a} - \boldsymbol{p}\|\}\right\} \\
&= \left\{\boldsymbol{p} \in \mathbb{R}^2 \;\middle|\; \bigwedge_{\boldsymbol{a}\in A} \ell \geq \|\boldsymbol{a} - \boldsymbol{p}\|\right\} \\
&= \left\{\boldsymbol{p} \in \mathbb{R}^2 \;\middle|\; \bigwedge_{\boldsymbol{a}\in A} \boldsymbol{p} \in \mathbb{B}_\ell(\boldsymbol{a})\right\} \\
&= \left\{\boldsymbol{p} \in \mathbb{R}^2 \;\middle|\; \boldsymbol{p} \in \bigcap_{\boldsymbol{a}\in A} \mathbb{B}_\ell(\boldsymbol{a})\right\} \\
&= \bigcap_{\boldsymbol{a}\in A} \mathbb{B}_\ell(\boldsymbol{a}).
\end{aligned} \tag{4.10}$$



www.manaraa.com

Thus, as illustrated in Figure 4.2, the problem of testing whether a set of attached sites $A$ is feasible is equivalent to testing whether the intersection of the balls around each site in $A$ is non-empty.

### 4.2.3 Feasible sites

The definition of feasible sites can be similarly specialized for point-bodied walkers. Consider that for point-bodied walkers, the distance to a site from feasible body positions $\mathcal{F}$, given by Equation 3.16 becomes

$$d(\boldsymbol{s}, \mathcal{F}) = \min_{\boldsymbol{p}\in\mathcal{F}}\{\|\boldsymbol{s}-\boldsymbol{p}\|\}. \tag{4.11}$$

Then, using Equation 4.11, we can specialize the definition of the set of feasible sites based on Definition 3.6.4, showing that

$$\begin{aligned} S_{\mathcal{F}} = S_{\mathcal{F}(A)} &= \{\boldsymbol{s}\in S\setminus A \mid \ell \geq d(\boldsymbol{s}, \mathcal{F}(A))\} \\ &= \left\{\boldsymbol{s}\in S\setminus A \;\middle|\; \ell \geq \min_{\boldsymbol{p}\in\mathcal{F}(A)}\{\|\boldsymbol{s}-\boldsymbol{p}\|\}\right\} \\ &= \left\{\boldsymbol{s}\in S\setminus A \;\middle|\; \bigvee_{\boldsymbol{p}\in\mathcal{F}(A)} \ell \geq \|\boldsymbol{s}-\boldsymbol{p}\|\right\} \\ &= \left\{\boldsymbol{s}\in S\setminus A \;\middle|\; \bigvee_{\boldsymbol{p}\in\mathcal{F}(A)} \boldsymbol{p}\in\mathbb{B}_\ell(\boldsymbol{s})\right\} \\ &= \{\boldsymbol{s}\in S\setminus A \mid \mathcal{F}(A)\cap\mathbb{B}_\ell(\boldsymbol{s})\neq\emptyset\}. \end{aligned} \tag{4.12}$$

As illustrated in Figure 4.3, we find for potential site $\boldsymbol{s}\notin A$,

$$\begin{aligned} \boldsymbol{s}\in S_{\mathcal{F}(A)} &\iff \mathbb{B}_\ell(\boldsymbol{s})\cap\mathcal{F}(A)\neq\emptyset \\ &\iff \mathbb{B}_\ell(\boldsymbol{s})\cap\bigcap_{\boldsymbol{a}\in A}\mathbb{B}_\ell(\boldsymbol{a})\neq\emptyset \\ &\iff \bigcap_{\boldsymbol{a}\in A\cup\{\boldsymbol{s}\}}\mathbb{B}_\ell(\boldsymbol{a})\neq\emptyset \\ &\iff \mathcal{F}(A\cup\{\boldsymbol{s}\})\neq\emptyset. \end{aligned} \tag{4.13}$$



www.manaraa.com

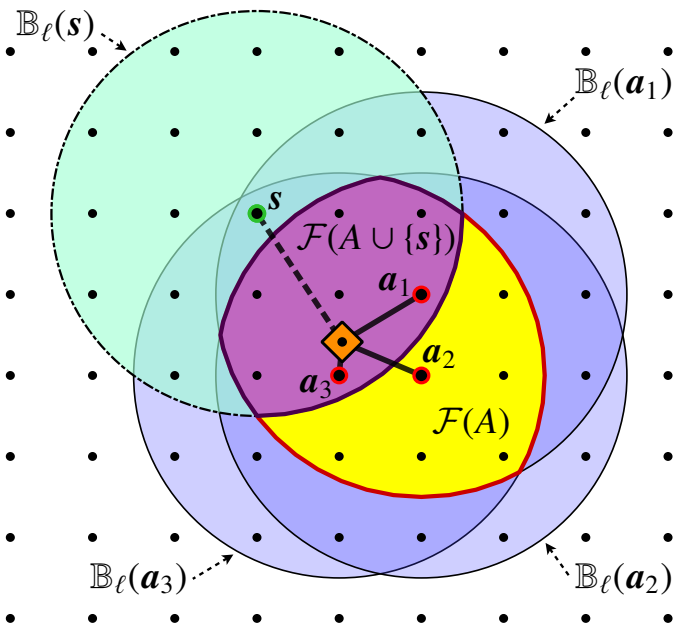


**Figure 4.3:** The feasible body positions $\mathcal{F}(A)$ for a point-bodied walker are indicated in yellow, as determined by the constraints of attached legs $A = \{\boldsymbol{a}_i\}$. A site $\boldsymbol{s} \notin A$ is feasible if $\mathcal{F}(A \cup \{\boldsymbol{s}\}) = \mathbb{B}_\ell(\boldsymbol{s}) \cap \mathcal{F}(A) \neq \emptyset$.

Equation 4.13 shows that the leg attachment process is consistent: from feasible configuration $A$, with $\mathcal{F}(A) \neq \emptyset$, any site $\boldsymbol{s} \in S_{\mathcal{F}}$ leads to a feasible configuration with $\mathcal{F}(A \cup \{\boldsymbol{s}\}) \neq \emptyset$, and any site $\boldsymbol{s} \notin S_{\mathcal{F}} \cup A$ leads to an infeasible configuration with $\mathcal{F}(A \cup \{\boldsymbol{s}\}) = \emptyset$.

## 4.3 Lattice surfaces

When spiders move over regular lattices of sites, it is possible to take advantage of the translational invariance to greatly simplify the computation of attachment rates.

### 4.3.1 Regular lattices

**Definition 4.3.1.** A lattice is a pair $(L, \mathfrak{L})$, where $L \subset \mathbb{Z}^2$ is a set of *valid coordinates*, and $\mathfrak{L}$ is the *lattice location transform*, $\mathfrak{L} = \mathfrak{L}(\boldsymbol{\delta}, \boldsymbol{o}) : \mathbb{Z}^2 \to \mathbb{R}^2$, with origin $\boldsymbol{o} = (o_x, o_y) \in \mathbb{R}^2$



www.manaraa.com

and spacing $\boldsymbol{\delta} = (\delta_x, \delta_y) \in \mathbb{R}^2$. We define for $\boldsymbol{c} = (c_x, c_y) \in \mathbb{Z}^2$,

$$\mathfrak{L}(\boldsymbol{\delta}, \boldsymbol{o})(\boldsymbol{c}) = \Delta\boldsymbol{c} + \boldsymbol{o} = \begin{bmatrix} \delta_x & 0 \\ 0 & \delta_y \end{bmatrix} \begin{bmatrix} c_x \\ c_y \end{bmatrix} + \begin{bmatrix} o_x \\ o_y \end{bmatrix}. \tag{4.14}$$

For simplicity we maintain $\boldsymbol{o} = (0, 0)$, throughout our analysis, so that

$$\mathfrak{L}(\delta, \boldsymbol{o}) = \mathfrak{L}(\delta) = \Delta. \tag{4.15}$$

Thus, $\mathfrak{L}$ is linear, and there is an inverse function $\mathfrak{L}^{-1} : \mathbb{R}^2 \to \mathbb{R}^2$, where $\boldsymbol{c} \in \mathbb{Z}^2$ implies $\mathfrak{L}^{-1}(\mathfrak{L}(\boldsymbol{c})) = \boldsymbol{c}$.

### 4.3.2 Leg configurations

Walkers moving over sites arranged as regular lattices have $S = \mathfrak{L}(L)$, so we can identify sites $\boldsymbol{s}$ with corresponding lattice coordinates $\boldsymbol{c} \in L$ where $\mathfrak{L}^{-1}(\boldsymbol{s}) = \boldsymbol{c}$.

**Definition 4.3.2.** We call the set of lattice coordinates of the attached leg sites, $C$, the *configuration* of the walker legs. This uniquely defines the attached locations in $S$ as $\mathfrak{L}(C) = A$.

### 4.3.3 Feasible configurations

Generalizing Equation 4.10 we define $\mathcal{F}(\cdot)$ not just for attached sites $A \subset S$, but also for attached leg configurations $C$,

$$\mathcal{F}(C) \equiv \mathcal{F}(\mathfrak{L}(C)) = \bigcap_{\boldsymbol{c} \in C} \mathbb{B}_\ell(\mathfrak{L}(\boldsymbol{c})). \tag{4.16}$$

Also in analogy to Equation 4.13, for $\boldsymbol{c} \in L$ we assert

$$\boldsymbol{c} \in S_{\mathcal{F}(C)} \iff (\boldsymbol{c} \notin C) \ \wedge \ (\mathcal{F}(C) \cap \mathbb{B}_\ell(\mathfrak{L}(\boldsymbol{c})) \neq \emptyset). \tag{4.17}$$

**Definition 4.3.3.** A configuration $C$ with $\mathcal{F}(C) \neq \emptyset$ is called a *feasible configuration*.



www.manaraa.com

**Definition 4.3.4.** Lattice coordinates $\boldsymbol{c} \in L$ are *feasible coordinates* and correspond to a *feasible site* if $\boldsymbol{c} \in S_{\mathcal{F}(C)}$.

**Definition 4.3.5.** With respect to Definition 3.6.2, let the feasibility probability of coordinates $\boldsymbol{c}$ from configuration $C$ be

$$f(\boldsymbol{c}) \equiv f_{\boldsymbol{B}(C)}(\mathfrak{L}(\boldsymbol{c})),$$

where $\boldsymbol{B}(C)$ is given by Equation 3.8 with $\mathcal{F} = \mathcal{F}(C)$.

### 4.3.4 Transformational invariance of coordinates

All of these definitions for lattice sites now allow us to take advantage of the regularity and translational invariance of the lattice structure.

**Definition 4.3.6.** A lattice *coordinate translation* is a mapping $\Phi : \mathbb{Z}^2 \to \mathbb{Z}^2$ which translates coordinates $c \in \mathbb{Z}^2$ by some translation vector $\varphi \in \mathbb{Z}^2$,

$$\Phi(c) = c + \varphi.$$

Now for any configuration $C \subset L$, and coordinate translation $\Phi$, if $\Phi(C) \subset L$ (i.e., all of the translated coordinates are still *valid*), then from Equation 4.16,

$$\begin{aligned} \mathcal{F}(\Phi(C)) &= \bigcap_{z \in \Phi(C)} \mathbb{B}_\ell(\mathfrak{L}(z)) \\ &= \bigcap_{\boldsymbol{c} \in C} \mathbb{B}_\ell(\mathfrak{L}(\boldsymbol{c}) + \mathfrak{L}(\varphi)) \\ &= \left[ \bigcap_{\boldsymbol{c} \in C} \mathbb{B}_\ell(\mathfrak{L}(\boldsymbol{c})) \right] + \mathfrak{L}(\varphi) \\ &= \mathcal{F}(C) + \mathfrak{L}(\varphi). \end{aligned} \tag{4.18}$$

Thus, the set of feasible sites is invariant under translation $\Phi$, except for a translation $\mathfrak{L}(\varphi)$.[1] Similarly from Equation 4.17, we maintain translational invariance of the concept

[1] For set $Z \subset \mathbb{R}^2$ and vector $\boldsymbol{v} \in \mathbb{R}^2$, we write $Z + \boldsymbol{v}$ to mean the translation of the entire set: $Z' = \{\boldsymbol{z} \in Z \mid \boldsymbol{v} + \boldsymbol{x}\} = Z + \boldsymbol{v}$.



www.manaraa.com

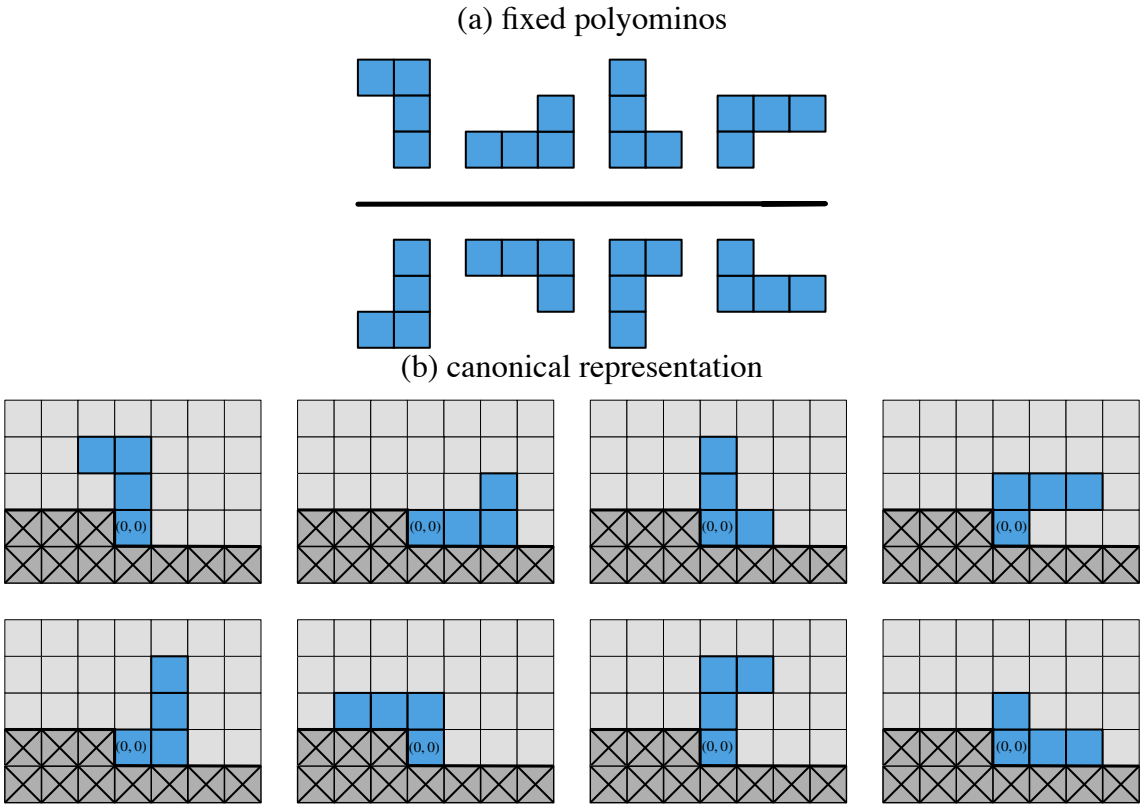


**Figure 4.4:** (a) A set of 8 distinct fixed polyominos. (b) The canonical representation for each polyomino, has a cell at the origin, and no cells in the crossed-out coordinates. Every fixed polyomino corresponds to a unique canonical representation.

of feasible sites so that for $\boldsymbol{s} \notin A$,

$$\begin{aligned}
\Phi(\boldsymbol{s}) \in S_{\mathcal{F}(\Phi(C))} &\iff \mathbb{B}_\ell(\mathfrak{L}(\Phi(\boldsymbol{s}))) \cap \bigcap_{\boldsymbol{c}\in C} \mathbb{B}_\ell(\mathfrak{L}(\Phi(\boldsymbol{c}))) \neq \emptyset \\
&\iff \mathbb{B}_\ell(\mathfrak{L}(\boldsymbol{s}) + \mathfrak{L}(\varphi)) \cap \bigcap_{\boldsymbol{c}\in C} \mathbb{B}_\ell(\mathfrak{L}(\boldsymbol{c}) + \mathfrak{L}(\varphi)) \neq \emptyset \\
&\iff \mathbb{B}_\ell(\mathfrak{L}(\boldsymbol{s})) \cap \bigcap_{\boldsymbol{c}\in C} \mathbb{B}_\ell(\mathfrak{L}(\boldsymbol{c})) \neq \emptyset \\
&\iff \boldsymbol{s} \in S_{\mathcal{F}(C)}.
\end{aligned} \tag{4.19}$$



www.manaraa.com

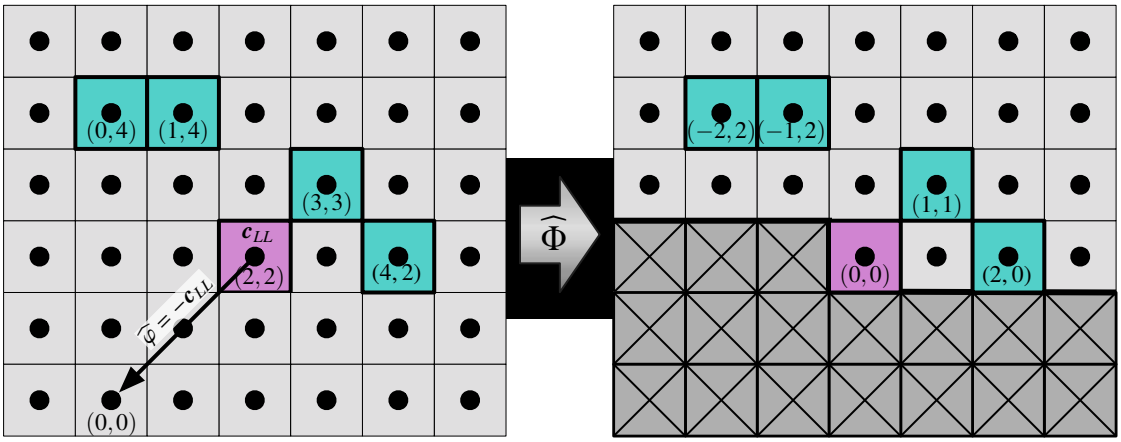


**Figure 4.5:** Any set of attached leg configurations has a canonical representation where the leftmost of the lowermost sites ($\boldsymbol{c}_{LL}$) is translated to the origin. This is the canonical transformation $\widehat{\phi} = -\boldsymbol{c}_{LL}$.

### 4.3.5 Canonical configurations

To determine the rate of attachment from configuration $C$, we need to compute $S_{\mathcal{F}(C)}$ (the set of all feasible sites), and the effective feasibility $f_{\boldsymbol{B}(C)}(\boldsymbol{s})$ for each site $\boldsymbol{s} \in S_{\mathcal{F}(C)}$, which in turn requires a Metropolis sampling (Chapter 5) of the Boltzmann distribution $\boldsymbol{B}(C)$ over sites in $\mathcal{F}(C)$.

We take advantage of the translational invariance of $\mathcal{F}$ (Equation 4.18) and $S_{\mathcal{F}}$ (Equation 4.19) to allow values computed for $S_{\mathcal{F}(C)}$ and $f_{\boldsymbol{B}(C)}(\boldsymbol{s})$ to be translated and reused for any other $C' \subset L$, where $C' = \Phi(C) = C + \varphi$. We define an equivalence class for configurations $C$,

$$E(C) = \left\{ C' \subset \mathbb{Z}^2 \;\middle|\; \exists \varphi \in \mathbb{Z}^2,\ (C' = C + \varphi) \right\}. \tag{4.20}$$

Any configuration in $E(C)$ can use the translated values for $S_{\mathcal{F}(C)}$ and $f_{\boldsymbol{B}(C)}$, so we only need to compute these values for a single configuration from the entire equivalence class.

Uniquely identifying a *canonical* configuration from the equivalence class as a representative could conceivably be done in many ways. We take inspiration from the algorithms used in counting polyominos. A *fixed polyomino* [51] is a connected set of lattice



www.manaraa.com

coordinates or cells (Figure 4.4a). In direct analogy to the (unconnected) coordinate sets for attached leg configurations, fixed polyominos are transitionally invariant and have an equivalence relation over translationally identical fixed polyominos. In order to count exactly one member from each equivalence class, enumeration algorithms [85, 104] define the concept of a canonical representation of a fixed polyomino (Figure 4.4b). All fixed polyominos correspond to some *canonical* fixed polyomino with one cell at the origin, no cells below the $x$-axis and no cells to the left of the origin with $y = 0$. We extend these concepts to walker leg configurations $C$.

**Definition 4.3.7.** The canonical mapping for attached leg configuration $C$ takes the leftmost of the lowermost sites, $\boldsymbol{c}_{LL}$, and translates it to the origin (Figure 4.5). The *canonical translation* of configuration $C$ is $\widehat{\Phi}_C$, where

$$\widehat{\Phi}_C(\boldsymbol{c}) = \boldsymbol{c} - \widehat{\varphi} = \boldsymbol{c} - \boldsymbol{c}_{LL}.$$

**Definition 4.3.8.** The set of *feasible canonical configurations* is defined as

$$\widehat{\mathbb{C}} = \left\{ C \subset \mathbb{Z}^2 \;\middle|\; [1 \leq |C| \leq k] \wedge [\mathcal{F}(C) \neq \emptyset] \wedge \left[ \forall\, (c_x, c_y) \in C,\ (c_y > 0) \vee (c_y = 0 \wedge c_x \geq 0) \right] \right\}.$$

The set of all possible canonical lattice coordinates can be defined as the set of all coordinates from any canonical configuration:

$$S_{\widehat{\mathbb{C}}} = \bigcup_{C \in \widehat{\mathbb{C}}} C. \tag{4.21}$$

Figure 4.6 shows $S_{\widehat{\mathbb{C}}}$ by noting that each canonical configuration has a walker with a leg attached at the origin, and the furthest that another leg can be feasibly attached is distance $d_{\max} = 2\ell$, so

$$S_{\widehat{\mathbb{C}}} = \left\{ (c_x, c_y) \in \mathbb{Z}^2 \;\middle|\; \left[ (c_y > 0) \vee (c_y = 0 \wedge c_x \geq 0) \right] \wedge \left[ \| \mathfrak{L}((c_x, c_y)) - \mathfrak{L}((0,0)) \| \leq 2\ell \right] \right\}. \tag{4.22}$$



www.manaraa.com

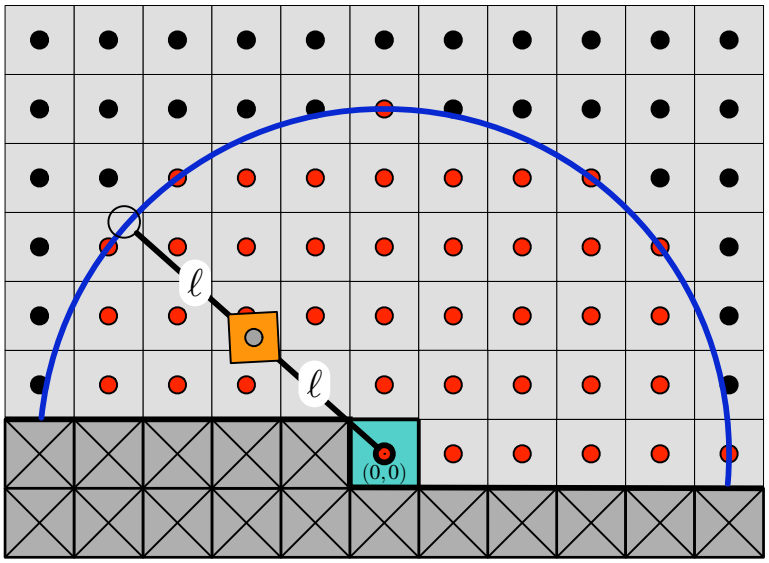


**Figure 4.6:** The set of all possible canonical lattice coordinates is $S_{\widehat{\mathbb{C}}}$, shown in red. Any canonical configuration has one attached foot at the origin and can only reach sites closer than the maximum site distance of $d_{\max} = 2\ell$.

### 4.3.6 Unique canonical configurations

The set $\widehat{\mathbb{C}}$ contains a single representative from each translational equivalence class of attached leg configurations. However, as shown in Figure 4.7, not every attached leg actually adds a new constraint on the body location. If $C \subset C'$ and $\mathcal{F}(C) = \mathcal{F}(C')$, then precomputed values for $\boldsymbol{B}(C)$, $S_{\mathcal{F}(C)}$, and $f_{\boldsymbol{B}(C)}$ can be used for $C'$, with the exception that sites in $C' \setminus C$ are feasible from $C$, but not from $C'$, so that

$$S_{\mathcal{F}(C')} = S_{\mathcal{F}(C)} \setminus C'. \tag{4.23}$$

This allows us to further reduce the set of precomputed configurations, from all of $\widehat{\mathbb{C}}$ to only those canonical configurations that define a unique set of leg constraints, i.e., to canonical configurations $\widehat{C}$ where $\mathcal{F}(\widehat{C}) \neq \mathcal{F}(\widehat{C}')$ for all $\widehat{C}' \subset \widehat{C}$.

**Definition 4.3.9.** The *unique canonical configuration ordering*, $\preceq$ on $\widehat{\mathbb{C}}$, is defined such that $\widehat{C} \preceq \widehat{C}'$ if $\widehat{C} \subseteq \widehat{C}'$, and $\mathcal{F}(\widehat{C}) = \mathcal{F}(\widehat{C}')$.



www.manaraa.com

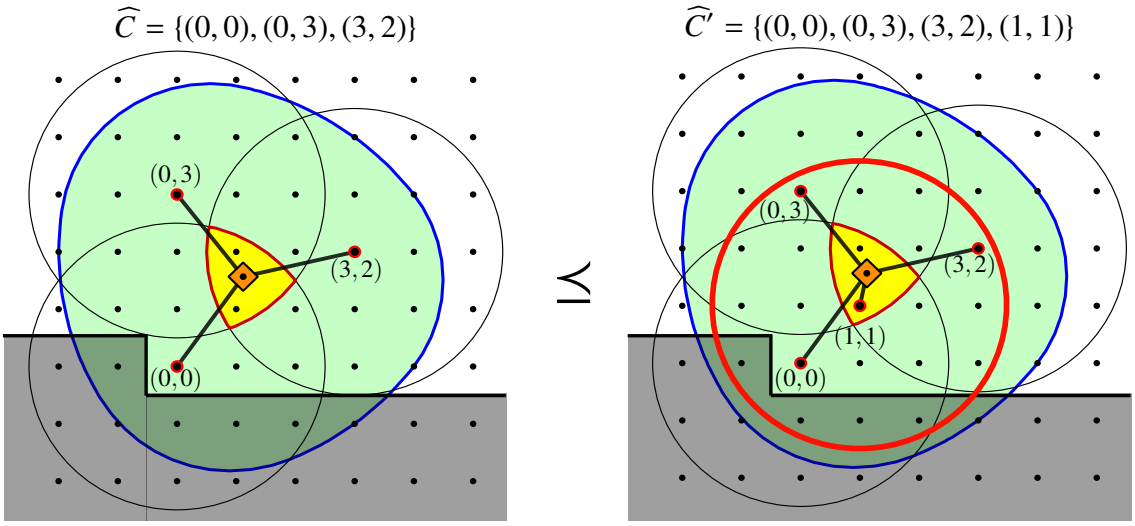


**Figure 4.7:** The canonical configurations $\widehat{C} = \{(0,0),(0,3),(3,2)\}$ and $\widehat{C}' = \{(0,0),(0,3),(3,2),(1,1)\}$ have the property that $\mathcal{F}(C) = \mathcal{F}(C')$ and $C \subset C'$. Thus, the two configurations have the same body position $\boldsymbol{B}$, the same set of feasible sites, and the same feasibility for those sites. We write $\widehat{C} \preceq \widehat{C}'$. In the case of canonical configuration $\widehat{C}$, all of the attached locations impose a constraint on the body, so there is no $\widehat{C}'' \subset \widehat{C}$ with $\mathcal{F}(C'') = \mathcal{F}(C)$. Thus $\widehat{C}$ is a minimal element of the relation $(\widehat{\mathbb{C}}, \preceq)$, and $\mathfrak{U}(\widehat{C}) = \widehat{C}$. We call $\widehat{C} \in \widehat{\mathbb{U}}$ a *unique canonical configuration*.

Using this definition, if $\widehat{C} \preceq \widehat{C}'$, the precomputed body distribution and attachment feasibility probabilities for $\widehat{C}$ can be used for $\widehat{C}'$.

**Definition 4.3.10.** The set of *unique canonical configurations* $\widehat{\mathbb{U}} \subseteq \widehat{\mathbb{C}}$ is the set of minimal elements of the unique canonical configuration ordering $\preceq$ over $\widehat{\mathbb{C}}$.

**Definition 4.3.11.** The *unique canonical mapping* $\mathfrak{U} : \widehat{\mathbb{C}} \to \widehat{\mathbb{U}}$ takes a canonical configuration $\widehat{C}$ to its equivalent unique canonical configuration $\widehat{U} = \mathfrak{U}(\widehat{C})$, which is the minimal element of the chain of $\widehat{C}$ in the canonical equivalent mapping $\preceq$.

## 4.4 KMC simulation of point-bodied spiders

For a KMC simulation in state $\omega = (P, A)$, a single KMC step involves determining the next state $\omega' = (P', A')$ and time increment $\tau$. Our goal in this section is to show how this



www.manaraa.com

| Part | Parameter | Description | Units |
|---|---|---|---|
| **Walker** | $k$ | The number of legs ($\geq 2$) | |
| | $\ell$ | The leg length | nm |
| **Kinetics** | $k_{\mathrm{S}}^{+}$ | The attachment pre-rate for substrates | $\mathrm{s}^{-1}$ |
| | $k_{\mathrm{P}}^{+}$ | The attachment pre-rate for products | $\mathrm{s}^{-1}$ |
| | $k_{\mathrm{cat}}$ | The catalysis rate | $\mathrm{s}^{-1}$ |
| | $k_{\mathrm{S}}^{-}$ | The off rate for substrates | $\mathrm{s}^{-1}$ |
| | $k_{\mathrm{P}}^{-}$ | The off rate for products | $\mathrm{s}^{-1}$ |
| | $T$ | The system temperature | K |
| | $\boldsymbol{f}$ | The force exerted on walkers | pN |
| | $\Delta U_0(\boldsymbol{p})$ | The walker body's internal energy | pN nm |
| **Surface** | $\vec{\delta}$ | The lattice spacing for surface sites | nm |
| | $L$ | The set of valid lattice points | |

**Table 4.1:** The relevant parameters to define the KMC simulation of a point-bodied MVRW moving over a lattice surface under the DNA substrate/product kinetics.

step is computed for the case of point-bodied spiders moving over lattices as described in Sections 4.3 and 4.2.

### 4.4.1 The canonical mapping determines the attachment rates

The MVRW kinetic Monte Carlo simulation must compute transition rates for any walker with attached legs $A$ corresponding to configuration $C = \mathfrak{L}^{-1}(A)$. This configuration has a canonical mapping $\widehat{\Phi}$ and canonical translation $\widehat{\varphi}$, so that we have canonical configuration $\widehat{C} = \widehat{\Phi}(C) \in \widehat{\mathbb{C}}$. Based on Definitions 4.3.10 and 4.3.11, this canonical configuration corresponds to a *unique canonical configuration* $\widehat{U} = \mathfrak{U}(\widehat{C}) \in \widehat{\mathbb{U}}$. Thus, to make the computation of transition rates efficient, the MVRW kinetic Monte Carlo simulation uses the following precomputed values for each $\widehat{U} \in \widehat{\mathbb{U}}$:

- $S_{\mathcal{F}(\widehat{U})}$ — The set of feasible sites from configuration $\widehat{U}$.
- $\left\langle \boldsymbol{B}(\widehat{U}) \right\rangle$ — The mean body location estimated as $\left\langle \boldsymbol{B}(\widehat{U}) \right\rangle = \frac{1}{N} \sum_{i=1}^{N} \boldsymbol{b}_i$ with samples $\left\{ \boldsymbol{b}_i \sim \boldsymbol{B}(\widehat{U}) \right\}_{i=1}^{N}$.



www.manaraa.com

- $f_{\boldsymbol{B}(\widehat{U})}(\boldsymbol{s})$ — The feasibility probability for each site in $S_{\mathcal{F}(\widehat{U})}$.

### 4.4.2 Possible transitions

From $\omega = (P, A)$ all transitions correspond to the five reactions of Equation 3.6, and include the unimolecular dissociation and cleavage reactions of the attached legs, and the bimolecular association reactions of the unattached legs. The rates for these reactions are summarized in Table 4.2. The overall rate of all possible transitions is then

$$R = R_{\mathrm{P}}^{-} + R_{\mathrm{S}}^{-} + R_{\mathrm{cat}} + R_{\mathrm{P}}^{+} + R_{\mathrm{S}}^{+}. \tag{4.24}$$

| Chemical pathway | Candidate sites | Overall rate | Next state ($\omega'$) |
|---|---|---|---|
| Product dissociation | $A_{\mathrm{P}} = \{A_{\mathrm{P}}^i\}_{i=1}^{n}$ | $R_{\mathrm{P}}^{-} = nk_{\mathrm{P}}^{-}$ | $(A \setminus \{A_{\mathrm{P}}^i\}, P)$ |
| Substrate dissociation | $A_{\mathrm{S}} = \{A_{\mathrm{S}}^i\}_{i=1}^{m}$ | $R_{\mathrm{S}}^{-} = mk_{\mathrm{S}}^{-}$ | $(A \setminus \{A_{\mathrm{S}}^i\}, P)$ |
| Substrate cleavage | $A_{\mathrm{S}} = \{A_{\mathrm{S}}^i\}_{i=1}^{m}$ | $R_{\mathrm{cat}} = mk_{\mathrm{cat}}$ | $(A \setminus \{A_{\mathrm{S}}^i\}, P \cup \{A_{\mathrm{S}}^i\})$ |
| Product association | $F_{\mathrm{P}} = \{F_{\mathrm{P}}^i\}_{i=1}^{u}$ | $R_{\mathrm{P}}^{+} = (d)k_{\mathrm{P}}^{+} \sum_{i=1}^{u} f(F_{\mathrm{P}}^i)$ | $(A \cup \{F_{\mathrm{P}}^i\}, P)$ |
| Substrate association | $F_{\mathrm{S}} = \{F_{\mathrm{S}}^i\}_{i=1}^{v}$ | $R_{\mathrm{S}}^{+} = (d)k_{\mathrm{S}}^{+} \sum_{i=1}^{v} f(F_{\mathrm{S}}^i)$ | $(A \cup \{F_{\mathrm{S}}^i\}, P)$ |

**Table 4.2:** The rates of all possible chemical transitions determine the probability of the next transition and the overall rate $R$. The association reactions are proportional to $d = k - |A|$, the number of detached legs. The next state $\omega'$ that the system moves to when attached site $i$ is chosen for dissociation or cleavage, or when feasible site $i$ is chosen for attachment is shown for each of the pathways.

**Unimolecular rates.** The set of attached sites $A$ is non-empty and can be divided into the attached products $A_{\mathrm{P}} = P \cap A$, and attached substrates $A_{\mathrm{S}} = A \setminus P$. Assume $|A_{\mathrm{P}}| = n$ and $|A_{\mathrm{S}}| = m$. Then the overall rates for product dissociation, substrate dissociation, and substrate cleavage are $R_{\mathrm{P}}^{-}$, $R_{\mathrm{S}}^{-}$, and $R_{\mathrm{cat}}$, as given in Table 4.2.



www.manaraa.com

**Attachment rates.** Using the definitions of the canonical mapping from Section 4.4.1, let the current configuration be $C$, with unique canonical configuration $\widehat{U}$. If $|C| < k$, then there are $d = k - |C|$ detached legs, and so we must determine the rate of each possible association reaction for these legs. The set of feasible sites from configuration $C$ is

$$S_{\mathcal{F}(C)} = \left(\widehat{\Phi}^{-1}\left(S_{\mathcal{F}(\widehat{U})}\right) \cap L\right) \setminus C. \tag{4.25}$$

In Equation 4.25, the intersection with $L$ ensures that the feasible sites are *valid* lattice coordinates, and, as in Equation 4.23, the removal of points in $C$ ensures that those sites corresponding to sites in $\widehat{C} \setminus \widehat{U}$ are not feasible since they are already attached. Thus, the effect of Equation 4.25 is to shift the feasible sites for the unique canonical configuration $\widehat{U}$ back to the locality of the current configuration $C$, and to eliminate those sites which are outside the lattice or already part of $C$. The result is the feasible sites $S_{\mathcal{F}(C)}$, which we divide into $F_{\mathrm{P}} = S_{\mathcal{F}(C)} \cap P$ and $F_{\mathrm{S}} = S_{\mathcal{F}(C)} \setminus P$. Let $|F_{\mathrm{P}}| = u$ and $|F_{\mathrm{S}}| = v$. Then the rates for these attachment reactions are given by Definition 3.6.3, and are summarized in Table 4.2

### 4.4.3 KMC step

With all transitions and rates enumerated in Table 4.2, we can now employ the KMC algorithm to select the next walker action and elapsed time according to the methodology of Section 4.1.1. We use random number $\alpha \sim \mathrm{Uniform}(0, R)$ to select the next category of chemical reaction and the specific site and leg involved, as depicted in Figure 3.6. Then we use $\tau \sim \mathrm{Exp}(R)$ as the time increment.

**Dissociation.** The only potential problem with this scheme is that dissociation of all legs will lead to dissociation of the entire walker from the surface, which leads to an undefined walker state. From the standpoint of measuring diffusive properties of walker motion, dissociation is problematic (Section 6.5). However, as long as the attachment rates are much faster than detachment rates, the probability of simultaneous detachment of all legs



www.manaraa.com

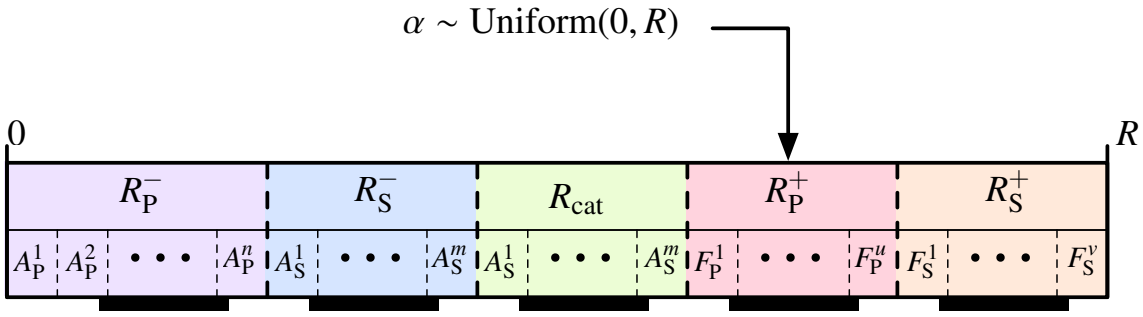


**Figure 4.8:** From a state with more than one leg attached, there are 5 categories of transitions that can occur, corresponding to each of the chemical reactions of Equation 3.6. The KMC algorithm uses a single random number uniform over $[0, R)$ to select the next reaction type and the corresponding site so that the overall probability of selecting a transition with rate $r$ is $r/R$.

becomes exponentially small in terms of the number of legs $k$. In the unlikely, but still possible, event that $|A| = 1$ and the next action chosen is dissociation or cleavage, we define a hopping rule, whereby the walker maintains its previous body distribution $\boldsymbol{B}(A)$, and associated attachment rates, while it waits for one of the $k$ detached legs to attach to a feasible site $\boldsymbol{s} \in S_{\mathcal{F}(A)}$. Thus, the next state will be $A = \{\boldsymbol{s}\}$, and the next time will be incremented by the $\tau$ time increment computed for dissociation, plus the time increment $\sigma \sim \text{Exp}(R')$, where $R'$ is the total attachment rate for all $k$ legs to all feasible sites $F_\text{S} = \{F_\text{S}^i\}_{i=1}^v$, and $F_\text{P} = \{F_\text{P}^i\}_{i=1}^u$, as well as the attachment rate for the now unoccupied site $\boldsymbol{a}_1 \in A$,

$$R' = R_\text{S}'^{+} + R_\text{P}'^{+} = (k)k_\text{S}^+ \sum_{i=1}^{v} f(F_\text{S}^i) + (k)k_\text{P}^+ \sum_{i=1}^{u} f(F_\text{P}^i) + (k)k_{\pi(\boldsymbol{a}_1)}^+ f(\boldsymbol{a}_1). \tag{4.26}$$

### 4.4.4 Precomputation of configurations and transition rates

The sets $\widehat{\mathbb{U}}$, $\widehat{\mathbb{C}}$, and $S_{\widehat{\mathbb{C}}}$ and the mapping $\mathfrak{U}$ depend only on the simulation variables $(k, \ell, \boldsymbol{\delta})$ from Table 4.2. This means that these values need to be precomputed only for each unique set of parameters $(k, \ell, \boldsymbol{\delta})$, which we call a *pattern* of the walker gaits on the lattice. The



www.manaraa.com

pattern $(k, \ell, \boldsymbol{\delta})$ uniquely defines the variables

$$\begin{aligned} \widehat{\mathbb{U}}(k, \ell, \boldsymbol{\delta}) &= \widehat{\mathbb{U}}, \\ \widehat{\mathbb{C}}(k, \ell, \boldsymbol{\delta}) &= \widehat{\mathbb{C}}, \\ S_{\widehat{\mathbb{C}}}(k, \ell, \boldsymbol{\delta}) &= S_{\widehat{\mathbb{C}}}, \text{ and} \\ \mathfrak{U}(k, \ell, \boldsymbol{\delta}; \widehat{C}) &= \mathfrak{U}(\widehat{C}). \end{aligned} \tag{4.27}$$

The simulation framework makes use of our natural-key based uniqueness constraint management (Chapter 9) to ensure that there is at most one *pattern cache* for each unique value of the tuple $\mathfrak{n}_{\text{pat}} = (k, \ell, \boldsymbol{\delta})$.

The body distribution $\boldsymbol{B}$ and site feasibility probabilities $f_{\boldsymbol{B}}$ for all configurations in $\widehat{\mathbb{U}}$ are also cached by the simulation framework. These depend on the pattern parameters $\mathfrak{n}_{\text{pat}} = (k, \ell, \boldsymbol{\delta})$, as well as the parameters $\Delta U_0(\boldsymbol{p})$, $T$, and $\boldsymbol{f}$ from Table 4.2. Thus, the parameters $(\mathfrak{n}_{\text{pat}}, \Delta U_0, T, \boldsymbol{f})$ define the attachment transition rates, and so define

$$\begin{aligned} \boldsymbol{B}(\mathfrak{n}_{\text{pat}}, \Delta U_0, T, \boldsymbol{f}; C) &= \boldsymbol{B}(C), \\ \mathcal{F}_{\boldsymbol{B}(C)}(\mathfrak{n}_{\text{pat}}, \Delta U_0, T, \boldsymbol{f}) &= \mathcal{F}_{\boldsymbol{B}(C)}, \text{ and} \\ f_{\boldsymbol{B}(C)}(\mathfrak{n}_{\text{pat}}, \Delta U_0, T, \boldsymbol{f}; \boldsymbol{s}) &= f_{\boldsymbol{B}(C)}(\boldsymbol{s}). \end{aligned} \tag{4.28}$$

Similar to the pattern cache, the simulation framework represents at most one *transition cache* for each unique value of the tuple $\mathfrak{n}_{\text{trans}} = (\mathfrak{n}_{\text{pat}}, \Delta U_0, T, \boldsymbol{f})$.



www.manaraa.com

# Chapter 5

# Metropolis Sampling and the Equilibrium Body Position

The MVRW model assumes that the body and unattached legs come to mechanical equilibrium in between the discrete chemical state transitions. In Section 3.6 we explain how the attachment rate $r^i_{\boldsymbol{B}}(\boldsymbol{s})$ of a leg to a feasible site $\boldsymbol{s}$ depends on the equilibrium distribution, via the attachment feasibility probability $f^i_{\boldsymbol{B}}(\boldsymbol{s})$ (Definition 3.6.2). Calculation of $f^i_{\boldsymbol{B}}(\boldsymbol{s})$ requires integration over the probability distribution $p_{\boldsymbol{B}}$,

$$f^i_{\boldsymbol{B}}(\boldsymbol{s}) = \int_{\mathcal{F}} p_{\boldsymbol{B}}(\boldsymbol{v}, \theta) I^i_{(\boldsymbol{v},\theta)}(\boldsymbol{s})\ d\boldsymbol{v}\, d\theta.$$

With a sample from $\boldsymbol{B}$, we can approximate the value of this integral and obtain the needed feasibility probabilities $f^i_{\boldsymbol{B}}(\boldsymbol{s})$ for each $\boldsymbol{s} \in S_{\mathcal{F}}$. Samples $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n \sim \boldsymbol{B}$ can be used as an unbiased estimator for a function $f$ of the random variable $\boldsymbol{B}$ giving the body's equilibrium position [68],

$$\langle f(\boldsymbol{B}) \rangle = \int_{\mathcal{F}} f(\boldsymbol{p}) p_{\boldsymbol{B}}(\boldsymbol{p}) d\boldsymbol{p} = \left\langle \frac{1}{n} \sum_{i=1}^{n} f(\boldsymbol{b}_i) \right\rangle. \tag{5.1}$$

Distribution $p_{\boldsymbol{B}}$ is defined in Equation 3.8 as the Boltzmann distribution over $\mathcal{F}$ under energy function $\Delta U_f(\boldsymbol{b})$ from Equation 3.18. The energy at each position is simple to compute; however, normalizing the probabilities of the Boltzmann distribution in Equation 3.8 requires computation of the partition function (Equation 3.9),

$$Z = \int_{\mathcal{F}} e^{-\beta E(\boldsymbol{p})} d\boldsymbol{p}. \tag{5.2}$$

Other than in the simplest cases, direct computation of $Z$ is difficult. The Metropolis-Hastings (MH) sampling algorithm (Section 5.1) allows $\boldsymbol{B}$ to be sampled using only ratios of the point probabilities $p_{\boldsymbol{B}}(\boldsymbol{b}')/p_{\boldsymbol{B}}(\boldsymbol{b})$ and thus the partition function $Z$ cancels, eliminating the need to compute it.

In the general MVRW model as outlined in Chapter 3 we potentially need to sample from a different equilibrium distribution $\boldsymbol{B}$ on each KMC step. However, as described in



www.manaraa.com

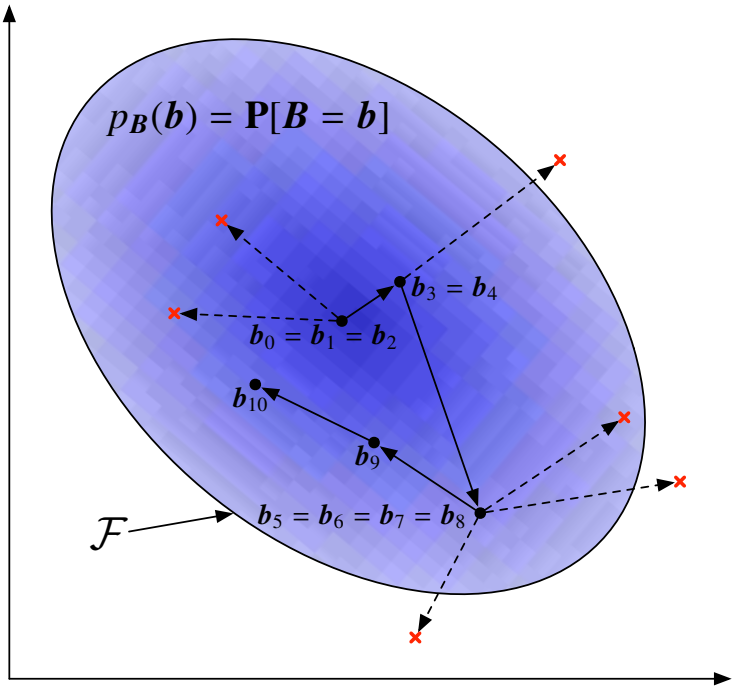


**Figure 5.1:** The Metropolis-Hastings algorithm samples from probability distribution $p_{\boldsymbol{B}}$, by simulating a Markov chain with an equilibrium distribution equal to $p_{\boldsymbol{B}}$. The algorithm iteratively generates a sequence of points $\{\boldsymbol{b}_i\}_{i=0}^{N}$. From position $\boldsymbol{b}_i$ it chooses candidate point $\boldsymbol{b}^\star$, and decides with probability $\alpha = \min\{1, p_{\boldsymbol{B}}(\boldsymbol{b}^\star)/p_{\boldsymbol{B}}(\boldsymbol{b}_i)\}$ to accept the candidate point and set $\boldsymbol{b}_{i+1} := \boldsymbol{b}^\star$, or reject the candidate point and set $\boldsymbol{b}_{i+1} := \boldsymbol{b}_i$. In this figure a red cross represents a rejected point, and a labeled black point represents an accepted point.

Section 4.4, the KMC simulation of point-bodied walkers over a regular lattice can take advantage of translational invariance to precompute transition rates for all possible leg configurations. Hence, Metropolis-Hastings is run as a preliminary step for each unique canonical configuration for a particular set of walker parameters, and then all simulations of that walker can rely on the single master set of transition probabilities (Section 4.4.4).



www.manaraa.com

## 5.1 The Metropolis-Hastings algorithm

The Metropolis-Hastings (MH) algorithm [54, 87, 107] samples from $p_{\boldsymbol{B}}$ by transforming any *candidate distribution* $Q(\boldsymbol{b} \to \boldsymbol{b}')$ over the distribution domain into an ergodic Markov process that has $p_{\boldsymbol{B}}$ as an equilibrium distribution. The Metropolis Markov chain is defined by transition probabilities $\widetilde{Q}$, where

$$\widetilde{Q}(\boldsymbol{b} \to \boldsymbol{b}') = Q(\boldsymbol{b} \to \boldsymbol{b}')\alpha, \tag{5.3}$$

and

$$\alpha = \min\left\{1, \frac{p_{\boldsymbol{B}}(\boldsymbol{b}')Q(\boldsymbol{b}' \to \boldsymbol{b})}{p_{\boldsymbol{B}}(\boldsymbol{b})Q(\boldsymbol{b} \to \boldsymbol{b}')}\right\}. \tag{5.4}$$

The result of the MH algorithm is a sequence of values $\{\boldsymbol{b}_i\}_{i=0}^{N}$. At step $i$, the simulation has value $\boldsymbol{b}_i$ and it uses this to draw a candidate value $\boldsymbol{b}^\star \sim q(\boldsymbol{b}) = Q(\boldsymbol{b} \to \boldsymbol{b}^\star)$. If Equation 3.8 is written as $p_{\boldsymbol{B}}(\boldsymbol{b}) = f(\boldsymbol{b})/Z$, and we use a symmetric candidate distribution, $Q(\boldsymbol{b} \to \boldsymbol{b}') = Q(\boldsymbol{b}' \to \boldsymbol{b})$, then from Equation 5.4, we calculate

$$\alpha = \min\left\{1, \frac{(f(\boldsymbol{b}^\star)/Z)Q(\boldsymbol{b}^\star \to \boldsymbol{b}_i)}{(f(\boldsymbol{b}_i)/Z)Q(\boldsymbol{b}_i \to \boldsymbol{b}^\star)}\right\} = \min\left\{1, \frac{f(\boldsymbol{b}^\star)Q(\boldsymbol{b}^\star \to \boldsymbol{b}_i)}{f(\boldsymbol{b}_i)Q(\boldsymbol{b}_i \to \boldsymbol{b}^\star)}\right\} = \min\left\{1, \frac{f(\boldsymbol{b}^\star)}{f(\boldsymbol{b}_i)}\right\}. \tag{5.5}$$

With probability $\alpha$ we choose to accept the point and we set $\boldsymbol{b}_{i+1} \coloneqq \boldsymbol{b}^\star$, otherwise we reject this candidate value and set $\boldsymbol{b}_{i+1} \coloneqq \boldsymbol{b}_i$. We repeat this until we have generated $N$ values. This procedure is illustrated in Figure 5.1. Importantly, using Equation 5.5 we never have to compute the partition function $Z$ because it cancels in the ratio of the probabilities. This makes the MH algorithm an efficient and effective means of sampling from $p_{\boldsymbol{B}}$.

## 5.2 Metropolis-Hastings implementation

Implementation of MH involves the setting of many algorithm parameters, including the candidate distribution, which can drastically affect the convergence rate and the rejection rate of the sampling procedure. The computational time and the resulting quality of the



www.manaraa.com

MH samples depend on these parameters. When the MH samples are being used for a particular statistical task, such as estimating tail probabilities of the equilibrium distribution, there are some analytical methods for selecting optimal parameter values [103]. However, for most applications, including the MVRW model, the choice of parameters must be done by trial and error [46], where one attempts to reduce the rejection rate, while simultaneously increasing the mixing rate and reducing the autocorrelation of the sequence.

### 5.2.1 Candidate distribution

The candidate distribution $Q(\boldsymbol{b} \to \boldsymbol{b}')$ can be chosen almost arbitrarily, but the choice of distribution affects the rejection rate and therefore the mixing time of the chain. Two common approaches to the selection of candidate distribution are the *random walk Metropolis chain* and the *independent sample Metropolis chain* [25]. A random walk Metropolis chain has

$$Q(\boldsymbol{b} \to \boldsymbol{b}') = q(\boldsymbol{b}' - \boldsymbol{b}), \tag{5.6}$$

where $q(\boldsymbol{x})$ is a symmetric multivariate density. The candidate site is $\boldsymbol{b}^\star = \boldsymbol{b}_i + \boldsymbol{x}$, where $\boldsymbol{x} \sim q(\boldsymbol{x})$. Thus the candidate site is chosen by taking a random step $\boldsymbol{x}$ from the current site.

An independent sampling chain uses

$$Q(\boldsymbol{b} \to \boldsymbol{b}') = q(\boldsymbol{b}'). \tag{5.7}$$

Thus, the choice of candidate site is independent of the current site, and $q(\boldsymbol{b}')$ should be chosen to be as close to $p_{\boldsymbol{B}}(\boldsymbol{b}')$ as possible. If the chosen candidate distribution is not symmetric, then the assumptions of Equation 5.5, do not hold, and the ratio $q(\boldsymbol{b}_i)/q(\boldsymbol{b}^\star)$ must also be computed to get $\alpha$.

Either of these methods can work but the choice of the candidate distribution, as well as the distribution parameters, affects the convergence rate and rejection rate of the Markov



www.manaraa.com

process. Typically the candidate distribution is thought of as a *tuning parameter* [125], and the choice is dictated by trading off large variance (which leads to fast mixing but high rejection rates) with small variance (which has slow mixing but low rejection rates).

For the MVRW application, we use the random walker Metropolis chain and choose the variance as a function of the size of the feasible region $\mathcal{F}$. We can easily compute the maximum horizontal or vertical dimension of the bounding box around $\mathcal{F}$, as

$$d_{\max} = \max\left[\left\{|b_x - b'_x| \;\middle|\; (b_x, b_y) \in \mathcal{F}\right\} \cup \left\{|b_y - b'_y| \;\middle|\; (b_x, b_y) \in \mathcal{F}\right\}\right]. \tag{5.8}$$

Then we let our candidate distribution be

$$q(\boldsymbol{b}) = \text{Uniform}(-\delta, \delta)\,, \tag{5.9}$$

where $\delta$ is defined by free parameter $\rho$,

$$\delta = d_{\max}/\rho. \tag{5.10}$$

As the parameter $\rho$ is increased, the steps in our random walk Metropolis chain become smaller. When $\rho \leq 1$, the steps are large enough to cover all of $\mathcal{F}$ from any current position $\boldsymbol{b}_i$, so the method becomes essentially equivalent to independent chain Metropolis. We found that values of $\rho \in [3, 10]$ give acceptable mixing times and low rejection rates. We use $\rho = 5$ for the results of Chapter 6.

### 5.2.2 Burn-in

The sequence of MH samples $\{\boldsymbol{b}_i\}_{i=0}^{N}$ always starts at a provided initial point $\boldsymbol{b}_0$, which is not necessarily a sample from equilibrium distribution $\boldsymbol{B}$. Thus the first few values of the chain will depend on the initial point. A typical implementation strategy drops a fixed number of initial points $a_{\text{MH}}$, assuming that thereafter the chain has reached equilibrium [46]. Depending on the nature of the distribution and the application, a threshold can be set so that subsequent points are independent of the starting value with high probability [45].



www.manaraa.com

| Symbol | Description | Standard Setting |
|---|---|---|
| | Random number generator | LCG64 |
| $N$ | number of samples | 100000 |
| $a_{\text{MH}}$ | Initial samples to skip | 500 |
| $b_{\text{MH}}$ | Thinning | 10 |
| $\rho$ | Step size ratio | 5 |
| $q(\boldsymbol{b})$ | Random Walk Metropolis candidate distribution | Uniform$(-\delta, \delta)$ |

**Table 5.1:** Metropolis-Hastings parameters

### 5.2.3 Thinning

While the sequence $\{\boldsymbol{b}_i\}_{i=0}^{N}$ will eventually be samples from the equilibrium distribution, these samples are correlated, especially when the candidate distribution has small variance. While correlated samples can be used as an unbiased estimator for $\boldsymbol{B}$, the number of samples needed increases with the autocorrelation of the series [45]. To reduce correlation, and thus the number of stored samples needed, MH implementations often use a process of *thinning* or *sub-sampling* of the sequence of points returned. The algorithm keeps only every $b_{\text{MH}}$-th sample after burn-in is complete.

### 5.2.4 MH parameters

We summarize the parameters used for the MVRW simulations in Table 5.1. In Section 8.2 we discuss parallel random number generation using the leapfrogging method for linear congruential pseudo-random number generators [11]. The overall number of MH samples needed when dropping $a_{\text{MH}}$ samples for burn-in and thinning by $b_{\text{MH}}$, given we desire $N$ total samples returned, is

$$N_{\text{total}} = a_{\text{MH}} + N(b_{\text{MH}}). \tag{5.11}$$



www.manaraa.com

# Chapter 6

# Results: Multivalent Random Walkers Move Superdiffusively Along Tracks

By itself a multivalent random walker is just a rather unsophisticated multivalent enzyme, but when paired with an appropriately designed nanoscale track of substrates it becomes a molecular transport device, able to move superdiffusively even under the influence of an external load force. Using the KMC simulations for point-bodied walkers on regular lattices (Section 4.4), we studied the motion of MVRWs moving over a semi-infinite track of substrates 3-wide (Figure 6.1). The relevant simulation parameters are summarized in Table 6.1. As shown in Figure 6.1, the walker starts with a single leg attached to the middle leftmost site. The remaining legs quickly attach, and the walker begins to move over the surface. From this initial position, the lack of substrates to the left means the walker can only move in the $+\hat{\mathbf{x}}$ direction, so we apply a force in the $-\hat{\mathbf{x}}$ direction to oppose the walker's motion. If the force applied to the walker is $\boldsymbol{f} = (f_x, f_y)$, we let $f_y = 0$, and write $f = -f_x$ as a scalar for the magnitude of the force in the $-\hat{\mathbf{x}}$ direction. We limit $f \leq 4.0\,\mathrm{pN}$ because larger forces result in insignificant motion under the parameters of Table 6.1. The upper bound of $f = 4.0\,\mathrm{pN}$ is near the maximum force a DNA-based realization of a MVRW could *a priori* be expected to move against, as the stall force for kinesin is approximately $5 - 8\,\mathrm{pN}$ [123], and the dissociation force for double-stranded DNA is $< 12\,\mathrm{pN}$ [35].

Under the kinetics from Table 6.1, $k_{\mathrm{cat}}$ serves a special role as it represents the sole kinetic difference between substrates and products. We have fixed $k_{\mathrm{S}}^{+} = k_{\mathrm{P}}^{+}$ so that there is no attachment bias between substrates and products. An unattached leg will just as rapidly bind to a feasible substrate as to a feasible product. Once bound, a leg–product complex unbinds at rate $k_{\mathrm{P}}^{-} = 1$, and a leg–substrate complex unbinds at rate $k_{\mathrm{S}}^{-} + k_{\mathrm{cat}}$. We assume that substrate unbinding is much less probable than substrate catalysis so we let $k_{\mathrm{S}}^{-} = 0$. Then if $k_{\mathrm{cat}} = 1$, there is no *residence time bias* between substrates and



www.manaraa.com

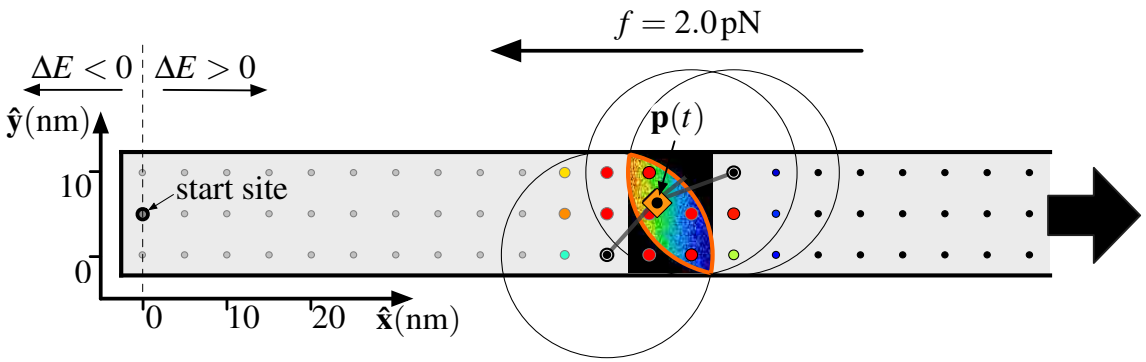


**Figure 6.1:** A snapshot several hundred steps into a MVRW simulation. The surface track for walkers in this set of simulations consists of a semi-infinite strip of substrate sites 3-wide. Shown are the circular constraints imposed by the attached legs, and the probability density $p_{\boldsymbol{B}}$ for the body's equilibrium position (as a heat map). The walker has one unattached leg, and the feasibility probability ($f(\boldsymbol{s})$) with which it would attach to site $\boldsymbol{s}$ is shown by the size and color of the site.

products—the expected duration of a leg–product binding is the same as that for a leg–substrate binding. While substrates are still converted into products, the kinetics of the walker attachment and detachment are identical for both species. Hence, a walker with $k_{\mathrm{cat}} = 1$ is equivalent to a walker moving over an all-product surface. But an all-product

**Table 6.1:** Model parameters used for simulations.

| Parameter Description | Symbol | Value |
|---|---|---|
| Number of legs | $k$ | 4 |
| Leg length | $\ell$ | 12.5 nm |
| Substrate spacing | – | 5.0 nm × 5.0 nm |
| Track width | – | 3 sites |
| Track length | – | semi-infinite |
| Initial set of product sites | $P$ | $\emptyset$ |
| Effective substrate binding rate | $k_{\mathrm{S}}^{+}$ | $1.0 \times 10^{3}\ \mathrm{s}^{-1}$ |
| Effective product binding rate | $k_{\mathrm{P}}^{+}$ | $1.0 \times 10^{3}\ \mathrm{s}^{-1}$ |
| Substrate dissociation rate | $k_{\mathrm{S}}^{-}$ | $0.0\ \mathrm{s}^{-1}$ |
| Product dissociation rate | $k_{\mathrm{P}}^{-}$ | $1.0\ \mathrm{s}^{-1}$ |
| Catalysis rate | $k_{\mathrm{cat}}$ | $\leq 1.0\ \mathrm{s}^{-1}$ |
| Temperature | $T$ | 300 K |
| Force in $-\hat{\mathbf{x}}$ direction | $f$ | $\leq 4.0$ pN |
| Largest simulated time | $t_{\mathrm{max}}$ | $1.0 \times 10^{6} - 1.0 \times 10^{7}$ s |



www.manaraa.com

surface provides no chemical free energy, and so an all-product walker system must move diffusively. In reality, the walkers with $k_{\text{cat}} = 1$ still release chemical energy when they catalyze a substrate, but the kinetics prevent them from utilizing that energy, and so they represent the no-energy baseline motion of walkers. In contrast, when $k_{\text{cat}} < 1$ there is a residence time bias, where leg–substrate bindings are longer in duration than leg–product bindings, as the leg must wait until the slow catalysis step completes before it can unbind. The only part of the walker kinetics which takes into account the chemical free energy released in substrate catalysis is the assumption of irreversibility in the enzymatic conversion from substrate to product. As described in Section 2.1, in enzyme kinetics there is some non-zero rate for the reverse of the catalytic process. However, if the Gibbs free energy ($\Delta G$) drop from substrate to product is large enough, the reverse rate is so small that it is for all practical purposes zero, and is omitted from the walker kinetics in our model. Thus, we vary the $k_{\text{cat}}$ parameter to control the residence time bias between visited and unvisited sites, and at $k_{\text{cat}} = 1$ the motion of the walker is *equivalent* to the no-free-energy case, but we do not directly incorporate $\Delta G$ into the model, as any free energy change large enough to make the substrate modification irreversible is sufficient to maintain the invariant that all unattached substrate sites are unvisited and therefore maintains the residence time bias between unvisited substrate sites and visited product sites.

## 6.1 Measuring the motion of multivalent random walkers

Unlike natural molecular motors, the motion of MVRWs is not ergodic, in the sense that the motion of the walkers depends on the state of the surface sites, and as time increases the limited local supply of substrates is depleted. The irreversibility of substrate cleavage implies that the Markov process is not recurrent, and proceeds through a series of transient states (Section 3.8). This implies that many statistical measures reported when analyzing the motion of natural molecular motors and other single particle transport systems, such as velocity [14, 69] and temporal MSD [74], are not valid in the case of the MVRW model,



www.manaraa.com

and we require more general methods of analyzing the motion of walkers applicable to non-ergodic systems.

### 6.1.1 Mean squared displacement

In single-particle tracking, the stochastic motion of individual molecules is frequently analyzed in terms of the mean squared displacement (MSD) [129]. The MSD is the variance in the displacement, $\mathbf{Var}\left(\|\boldsymbol{p}(t)\|\right) = \left\langle \|\boldsymbol{p}(t)\|^2 \right\rangle$. For any diffusive process (i.e., an unbiased random walk) the MSD will scale linearly with time. Anomalous diffusion [40, 55] is characterized by the MSD scaling as some non-linear power $0 \leq \alpha \leq 2$,

$$\left\langle \|\boldsymbol{p}(t)\|^2 \right\rangle = (2dD)t^{\alpha}, \begin{cases} \alpha = 0 & \text{stationary} \\ 0 < \alpha < 1 & \text{subdiffusive} \\ \alpha = 1 & \text{diffusive} \\ 1 < \alpha < 2 & \text{superdiffusive} \\ \alpha = 2 & \text{ballistic or linear} \end{cases}. \tag{6.1}$$

In Equation 6.1, $D$ is the diffusion constant, and $d$ is the dimension of the space the walkers move in, which for fixed-width tracks (Figure 6.1) is effectively $d = 1$. MSD can either be computed as a temporal average (over different $\delta t$ values for a single walker trajectory) or an ensemble average (over absolute $t$ for an ensemble of trajectories from identical walker systems). Many biological systems are (or are at least assumed to be) *ergodic* in the sense that the motion of a walker is independent of its absolute position on the track and does not depend on its previous motion over a region of that track [102]. Under the assumption of ergodicity the temporal and ensemble MSD are equivalent (assuming sufficient measurement resolution), but when a non-ergodic system is analyzed, only the ensemble average is meaningful for use in characterizing anomalous diffusion [67, 80]. MVRWs are a non-ergodic system because they irreversibly modify the surface as they move over it. Thus, the motion of the walker depends on its absolute position on the track



www.manaraa.com

and specifically on whether the local sites are products or substrates. Hence, only the ensemble MSD can be used to study MVRWs.

### 6.1.2 Number of sites cleaved

The number of sites a walker cleaves as a function of time, $N(t)$, is a random variable that provides several pieces of useful information, especially for walkers moving on 1D tracks. Under the kinetics of Table 6.1, a leg that binds a substrate always cleaves the site to leave a product because rate $k_S^- = 0$. This implies that $N(t)$ is equivalent to the number of sites visited, which for unbiased random walkers in 1D is

$$N(t) \propto t^{\gamma} \begin{cases} \gamma = 0 & \text{stationary} \\ 0 < \gamma < 1/2 & \text{subdiffusive} \\ \gamma = 1/2 & \text{diffusive} \\ 1/2 < \gamma < 1 & \text{superdiffusive} \\ \gamma = 1 & \text{ballistic} \end{cases} . \tag{6.2}$$

The only source of energy in the system is manifested in the irreversibility of substrate cleavage ($\Delta G < 0$), thus a walker needs to constantly visit and cleave new substrate sites to maintain a constant supply of energy. Unless $\gamma = 1$,

$$\frac{dN(t)}{dt} \propto t^{\gamma-1} \xrightarrow[t\to\infty]{} 0. \tag{6.3}$$

Thus, in 1D, a MVRW can maintain a constant supply of energy only while it is moving ballistically.

### 6.1.3 First passage time

Another useful measure of random motion is the *first passage time*, Fpt($d$), which for each distance $d > 0$ is a random variable giving the time for a walker to move at least distance



www.manaraa.com

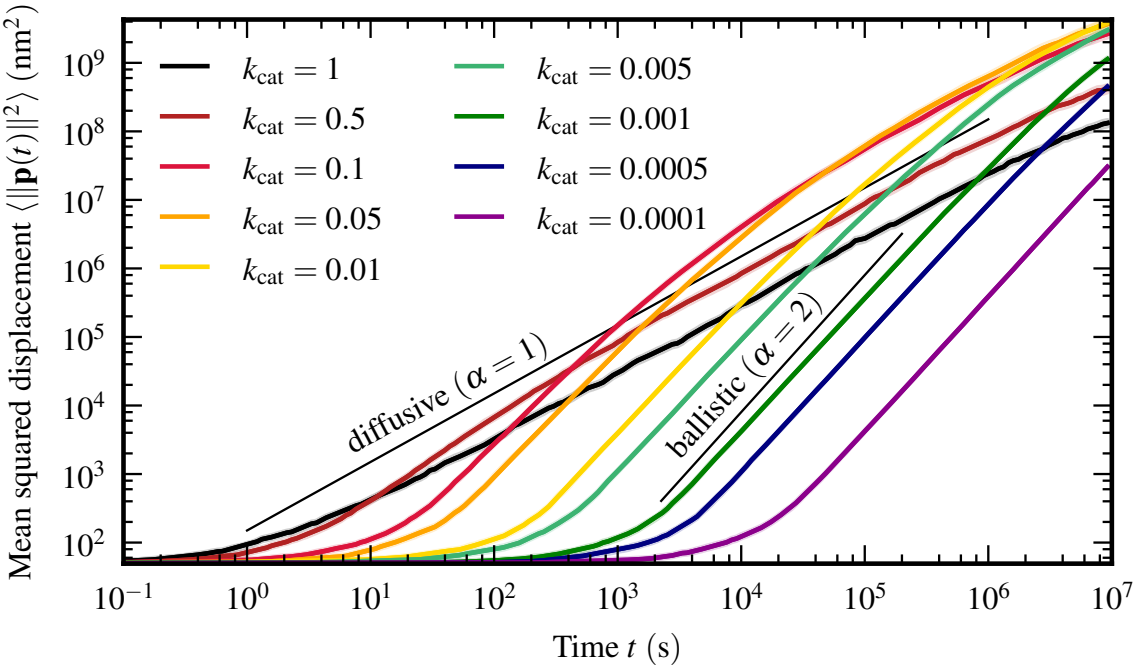


**Figure 6.2:** Simulation estimate of $\left\langle \|\boldsymbol{p}(t)\|^2 \right\rangle$ when $f = 0$. Walkers with $k_{\text{cat}} = 1$ move diffusively, those with $k_{\text{cat}} < 1$ move superdiffusively, but eventually exhaust their local supply of substrates and become ordinary diffusive. True transitions to diffusion will occur above simulated time $t_{\text{max}} = 10^7$.

$d$ from the origin [105]. The mean first passage time, $\langle \text{Fpt}(d) \rangle$, gives the average transport time for the walker to move distance $d$. First passage time can also be used to characterize diffusive motion where $\langle \text{Fpt}(d) \rangle \propto t^2$ for diffusive motion and $\langle \text{Fpt}(d) \rangle \propto t$ for ballistic motion in 1D.

## 6.2 Walkers move superdiffusively

Figure 6.2 shows the ensemble estimates ($N = 1000$) for MVRWs moving in the absence of a load force. Initially (below the characteristic timescale of $1/k_{\text{cat}}$) the walkers move subdiffusively. As expected the $k_{\text{cat}} = 1$ walkers never move faster than diffusion. However, as $k_{\text{cat}}$ is decreased, walkers initially move more slowly due to the the slower catalysis kinetics, but once sufficient time has passed, they move superdiffusively with $\alpha > 1$. The



www.manaraa.com

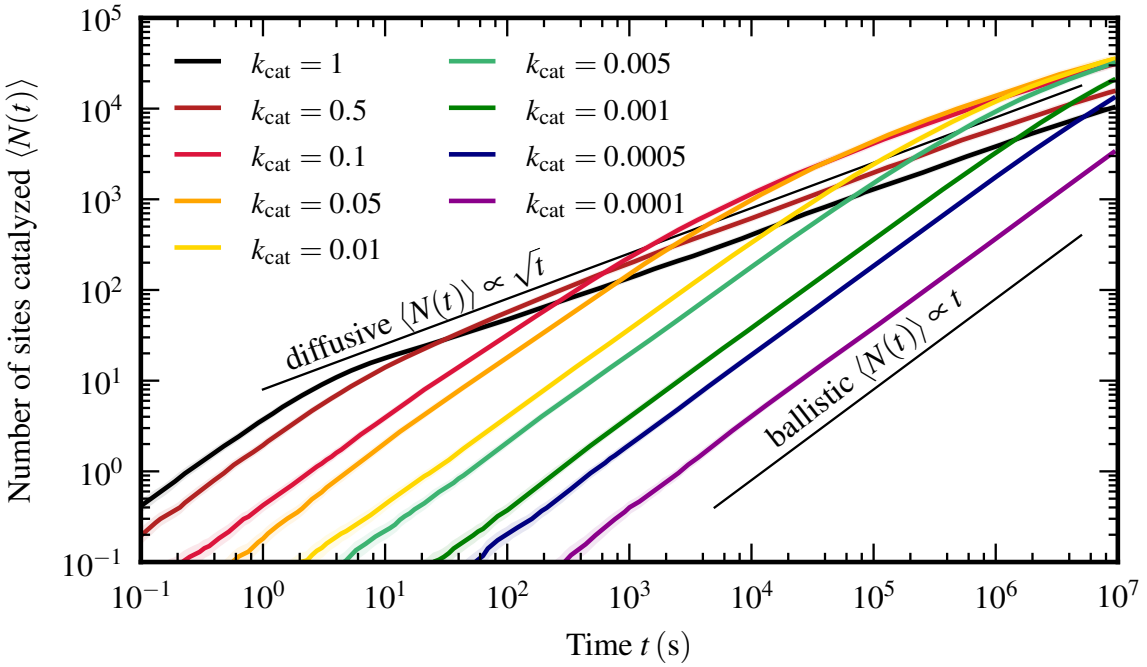


**Figure 6.3:** Simulation estimate of $\langle N(t)\rangle$, the number of substrates catalyzed to products when $f = 0$. Since $k_S^- = 0$, this is equivalent to the number of distinct sites visited at time $t$. Walkers with $k_{cat} < 1$ catalyze substrates at a nearly linear rate over many decades in time. This is necessary to maintain a constant supply of chemical energy to sustain superdiffusive motion.

smaller the value of $k_{cat}$, the more superdiffusively the walkers move, with $\alpha$ approaching 2 for the smallest $k_{cat}$ values. This superdiffusive behavior persists over several decades in time, during which the walkers are moving with a bias away from the origin and in the direction of unvisited sites. Because of this outward-directed bias, the walkers with $k_{cat} < 1$ eventually overtake (in MSD) the $k_{cat} = 1$ walkers given sufficient time. However, the ability to move superdiffusively depends on the local availability of the immobile substrate fuel, which is consumed as the walker moves over the track. Hence if a walker moves back over previously visited sites, it becomes starved for fuel. In these energy-devoid regions the walker can only move diffusively like the $k_{cat} = 1$ walkers, and so superdiffusion must eventually give way to regular diffusion, even for the smallest values of $k_{cat}$.

Figure 6.3 shows the number of sites catalyzed over time, and its rate of change repre-



www.manaraa.com

sents the average availability of substrate fuel. As long as the number of sites cleaved is linear with time, the walkers are receiving fuel at a constant rate and their motion is biased superdiffusively in the direction of new sites, which allows their constant fuel supply to be maintained. When $N(t)$ becomes sub-linear the walkers begin their transition in MSD from superdiffusion back to ordinary diffusion.

The mean first passage time, $\langle \mathrm{Fpt}(d) \rangle$, shown in Figure 6.4, gives yet another way to analyze the motion of these same walker configurations. Especially for the application of cargo transport, the first passage time is a practical measure of the utility of the MVRW system, showing that the walkers with $k_{\mathrm{cat}} < 1$ will eventually make it to a goal a given distance $d$ from the origin faster than the no-energy $k_{\mathrm{cat}} = 1.0$ walkers, as long as the distance $d$ is large enough.



www.manaraa.com

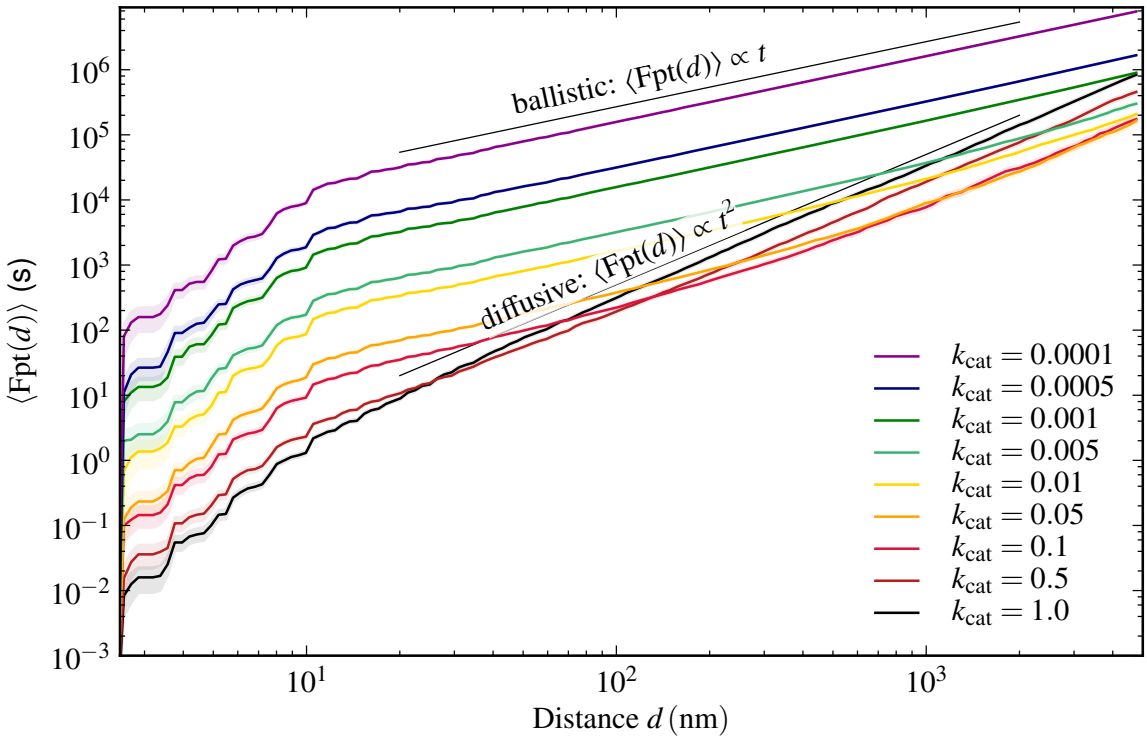


**Figure 6.4:** Simulation estimate of $\langle \mathrm{Fpt}(d) \rangle$, the mean first passage time to reach distance $d$ from the origin when $f = 0$. Lower times indicate faster mean time to travel distance $d$. We find that for short distances, the faster kinetics of the diffusive $k_{cat} = 1.0$ walkers is superior, but the superdiffusive motion of the walkers with $k_{cat} < 1$ leads to faster mean transit times for longer distances.

## 6.3 Walkers do work against a load

In order to experimentally quantify the walking velocity and other motor characteristics, natural motors have been studied using experimental techniques that allow the application of precise sub-piconewton forces on walkers while simultaneously measuring position to nanometer accuracy [30, 91, 119]. The force $\boldsymbol{f}$ applied to walkers in these experimental setups is constant, but can be applied in any direction. This allows the mechanics of the walking mechanism to be probed in controlled ways not directly possible for walkers moving actual nanoscale cargo loads. The natural biological function of a molecular motor does not necessarily involve motion in direct opposition to a constant force solely for the



www.manaraa.com

purpose of generating mechanical work. Natural molecular motors have more practical transportation tasks that are not as narrowly defined. However, observing the behavior of walkers under these artificial, controlled conditions leads to a microscopic understanding of how energy is used to bias the walking mechanism and allows probing of the sequence of internal states the walker molecule moves through in a single step [29].

### 6.3.1 Experimental setup

As a parallel to the experimental measurements of the effect of load force used for measuring kinesin, we study the motion of a point-bodied MVRW as it moves under the effect of a constant conservative load force $f$ in the $-\hat{\mathbf{x}}$ direction (Figure 6.1). Figure 6.5 shows ensemble ($N = 4000$) estimates of $\left\langle \|\boldsymbol{p}(t)\|^2 \right\rangle$ under a range of forces for $k_{\text{cat}} = 1$ and $k_{\text{cat}} = 0.01$. Again, $k_{\text{cat}} = 1$ (dashed lines) illustrates the no-energy case and, as shown previously (Figure 6.2), these $k_{\text{cat}} = 1$ walkers move diffusively without the influence of force.

When $f > 0$, the random walk over products is biased in the $-\hat{\mathbf{x}}$ direction. The constraints imposed by the surface prevent the walker from moving to the left of the origin, so the biased random walk will eventually reach an equilibrium position, after which the motion is stationary ($\alpha = 0$). Indeed, this is seen for the $k_{\text{cat}} = 1$ walkers, which never move faster than diffusion; their MSD increases monotonically to the equilibrium value exactly as if they were undergoing constrained diffusion in a box [106]. In contrast, when $k_{\text{cat}} < 1$ we again see nearly ballistic motion for all walkers except those under the highest load forces $f > 2.0\,\text{pN}$. Thus, even though the load force attempts to pull the walker body away from the substrate fuel, the long residence time for leg-substrate binding allows a few substrate-bound legs to resist the force and keep the walker in proximity to the substrate sites. Eventually, as in the $f = 0$ case, all walkers, regardless of $k_{\text{cat}}$, will exhaust their local supply of substrates and will find themselves moving over energy-devoid product sites, which ultimately brings them to the same equilibrium position as the $k_{\text{cat}} = 1$ walkers (for



www.manaraa.com

a given $f$).

The change in potential energy of the walkers as they move in opposition to the load force can be quantified by evaluating the ensemble estimate of the mean position of the walker body, $\langle \boldsymbol{p}(t) \rangle$. We choose to set $\Delta E = 0$ when $p_x = 0$, and then $\Delta E = f p_x > 0$ for walkers to the right of the origin (Figure 6.1). Figure 6.6 shows the ensemble estimate of $\langle \Delta E(t) \rangle$. As the load force is increased above 0, the walkers attain progressively higher potential energies, and their peak energies come earlier, as they need to move less distance to do the same amount of work. However, as the forces are increased beyond $f = 2\,\mathrm{pN}$, the walkers are not able to move very far without being pulled backwards away from their substrate fuel, and they achieve only modest values of $\Delta E$.



www.manaraa.com

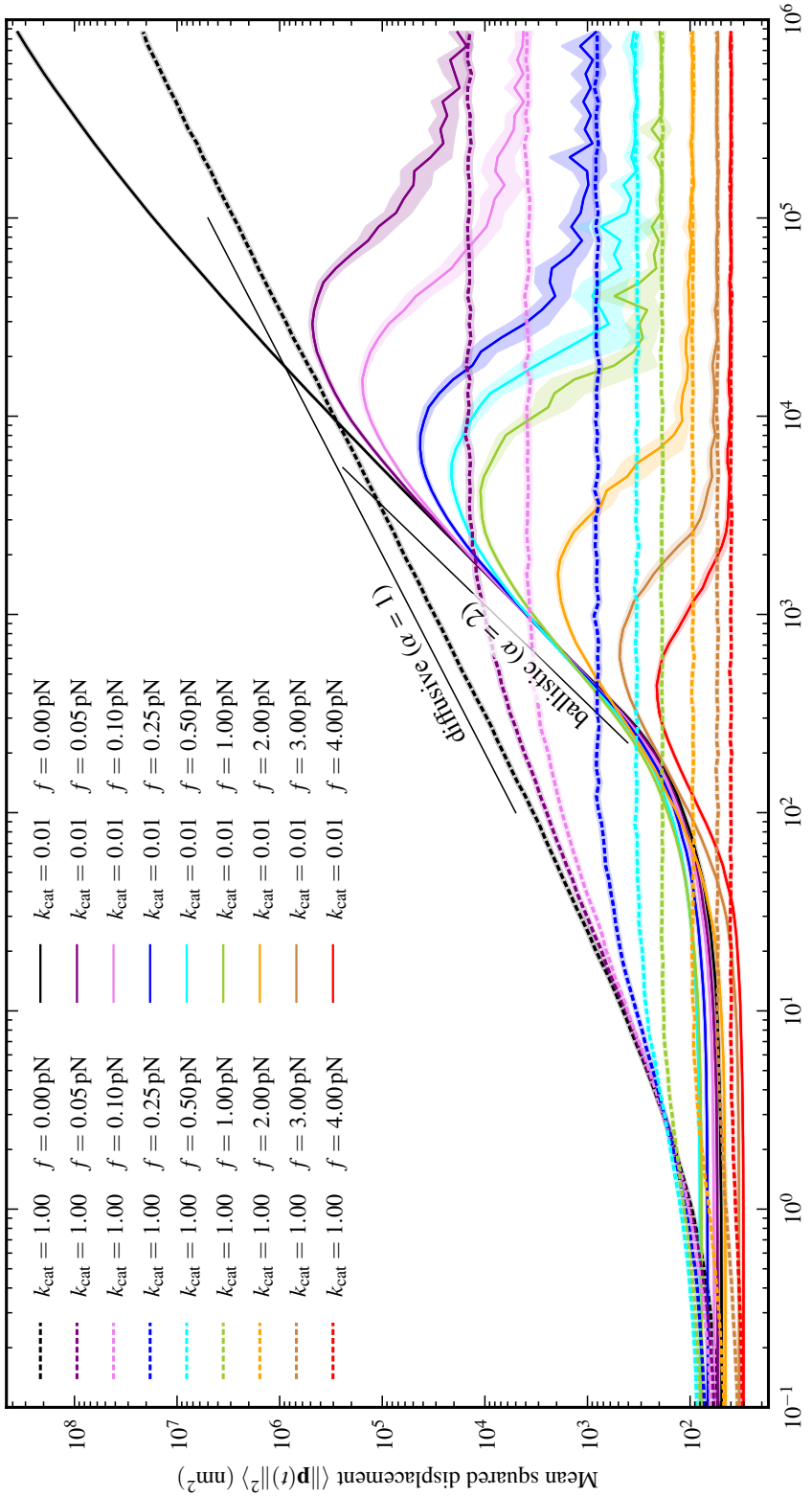


**Figure 6.5:** Simulation estimate of $\left\langle \|\boldsymbol{p}(t)\|^2 \right\rangle$ and 95% confidence bounds (shading) on a log-log scale. Reference lines are shown for ordinary diffusion ($\alpha = 1$) and ballistic motion ($\alpha = 2$). Walkers with $k_{cat} < 1$ move superdiffusively, but when $f > 0$, they eventually slow down and return to the same equilibrium position as the $k_{cat} = 1$ walkers.



www.manaraa.com

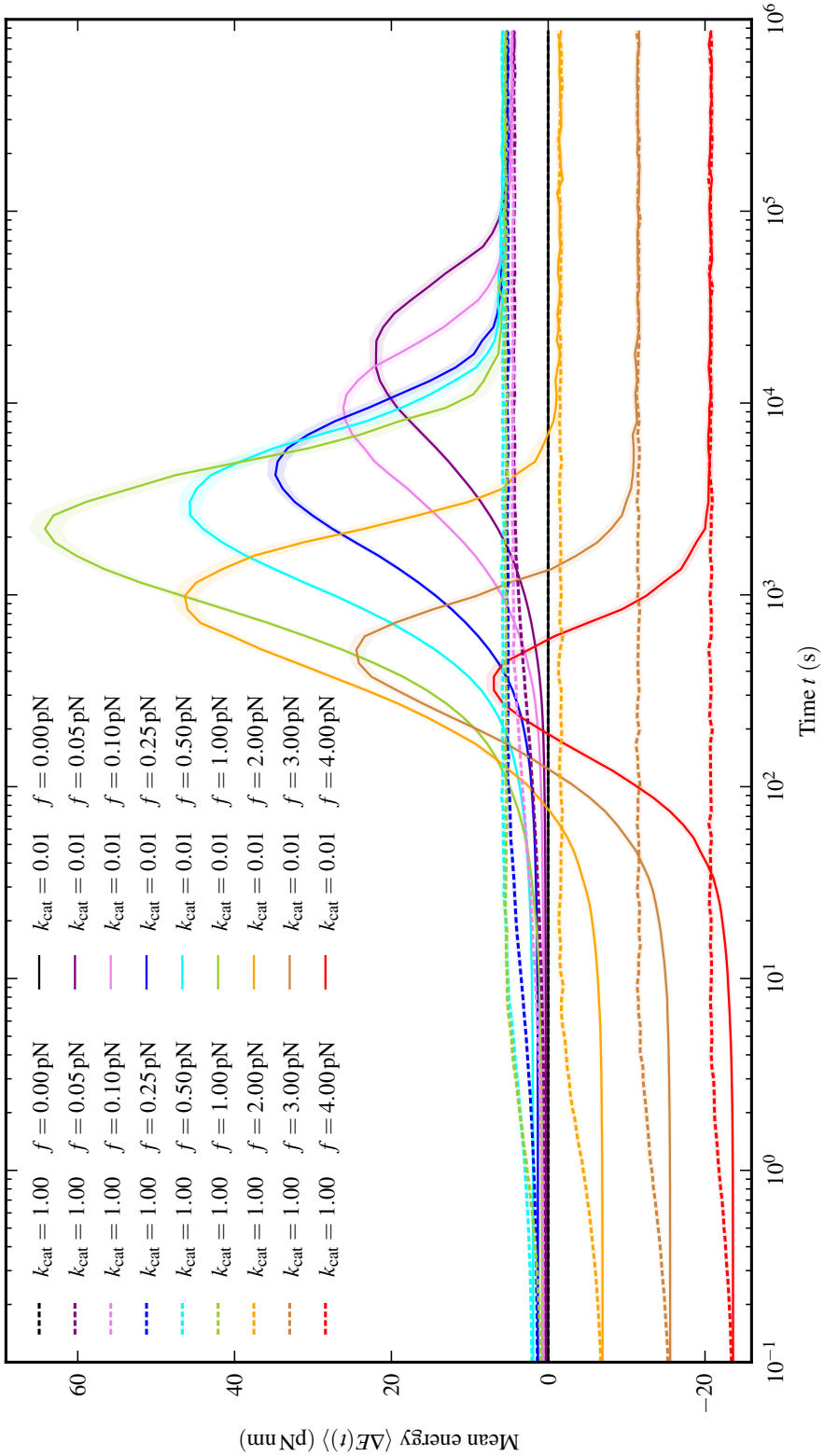


**Figure 6.6:** Simulation estimate of $\langle \Delta E(t) \rangle$ and 95% confidence bounds (shading) on a log-linear scale. Walkers with $f = 0$ always have $\Delta E = 0$. Those with $f > 0$ and $k_{cat} < 1$ do significant amounts of work, reaching a peak energy before eventually coming to an equilibrium with the $k_{cat} = 1$ walkers. This equilibrium value depends on the force, but for fixed $f$, walkers under any $k_{cat}$ will eventually reach the same equilibrium value.



www.manaraa.com

## 6.4 Peak work

When $f > 0$ all walkers eventually move to an equilibrium position with energy $\Delta E_\infty(f)$. This value is greater than the initial energy, because the walkers begin out of equilibrium with only a single leg attached (Figure 6.1). The initial energy of the walker $\Delta E_0(f) < 0$ because we measure $\boldsymbol{p}$ as the body's equilibrium position $\langle \boldsymbol{B} \rangle$, which under any non-zero force will have $p_x < 0$. However, the kinetics of $k_{\mathrm{P}}^+ \gg k_{\mathrm{P}}^-$ lead to an equilibrium where legs are almost always attached to a site, and because all sites are to the right of the origin, the equilibrium position $\Delta E_\infty(f)$ will also necessarily be greater than $\Delta E_0(f)$. Thus, to characterize the amount of useful work that a walker can do we take into account the equilibrium energy specific to each force.

**Definition 6.4.1.** The expected peak work for a walker moving under force $f$ is

$$w^\star(f) = \max_{t \in [0, t_{\max}]} \langle \Delta E(t; f) \rangle - \Delta E_\infty(f).$$

We estimate $\Delta E_\infty(f)$ as $\langle \Delta E(t_{\max}; f) \rangle$ for the $k_{\mathrm{cat}} = 1$ walker. Figure 6.7a shows $w^\star$ as force and $k_{\mathrm{cat}}$ are varied. The $k_{\mathrm{cat}} = 1$ walkers never have $w^\star > 0$, but the walkers with $k_{\mathrm{cat}} < 1$ can do significant work under moderate forces. We also show the values for $p_x^\star(f) = \max_{t \in [0, t_{\max}]} \langle p_x(t; f) \rangle - p_x^\infty(f)$ in Figure 6.7b. Note that the walkers move significantly farther under small loads, although they do nearly the same work.

## 6.5 Dissociation

There is a non-zero probability for a walker to detach from the track if $k - 1$ legs are simultaneously in the detached state, and the next action chosen is for the remaining leg to detach. A walker with $k$ detached legs is free to diffuse in solution, and cannot be ascribed a well-defined position with a discrete state Markov process. Hence, dissociation poses mathematical difficulties for analyzing a non-ergodic motive process and comparing



www.manaraa.com

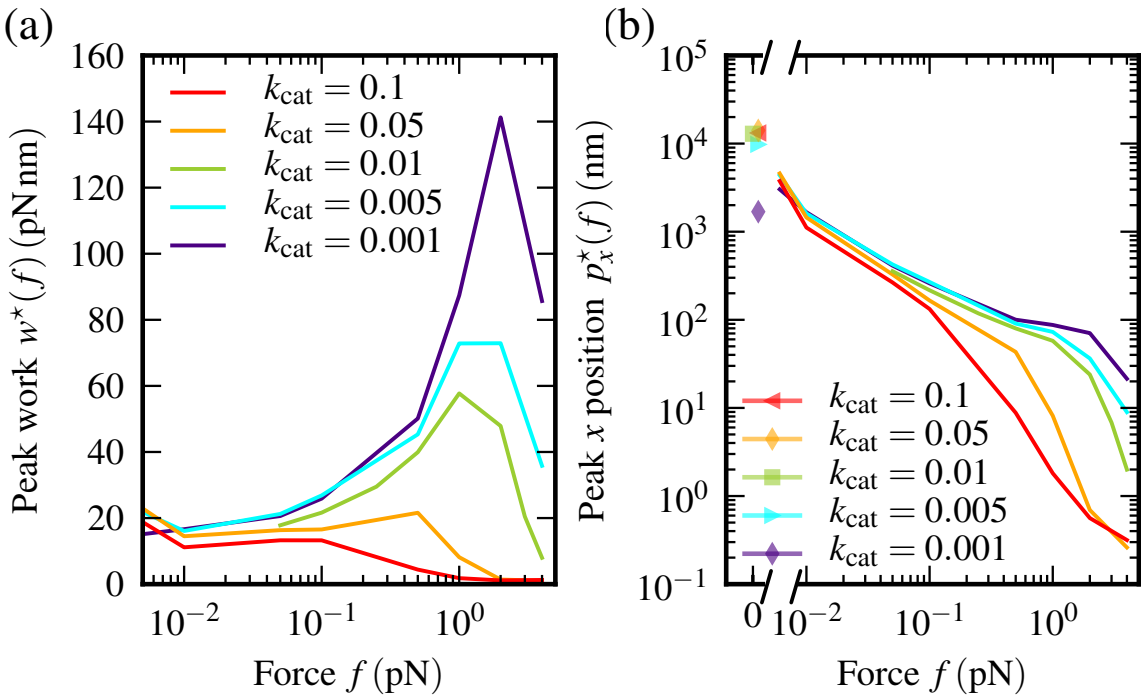


**Figure 6.7:** Simulation estimate of (a) peak work $w^\star(f)$ and (b) peak $x$ position $p_x^\star(f)$. Estimates for walkers with $k_{cat} \in \{1, 0.01\}$ use $N = 4000$ samples; estimates for walkers with other $k_{cat}$ values use $N = 250$ samples. The peak position for $f = 0$ is shown as well, which is limited by the simulated time $t_{max} = 1.0 \times 10^6$. In particular when $f = 0$, the $k_{cat} = 0.001$ walkers are still moving superdiffusively at $t = 1.0 \times 10^6$, but are limited by their slower stepping kinetics. At longer times the $k_{cat} = 0.001$ walkers will achieve a peak position greater than those achieved by the larger $k_{cat}$ walkers.

it with other mathematical models of anomalous diffusion. Ergodic models of kinesin can simultaneously analyze motion and dissociation because the transport characteristics and dissociation probabilities can be understood independently by studying a single motor cycle [8, 79]. MVRWs, being non-ergodic, have transport and dissociation probabilities that depend on the current state of the local chemical sites, and cannot be analyzed with similar techniques.

One approach to dealing with dissociation in non-ergodic walker models is to have a single absorbing dissociated state to which all walkers will eventually go and never return. This state is then the single equilibrium state of the system, and analysis is done



www.manaraa.com

on the remaining walkers. However, analyzing MSD becomes challenging because at any $t > 0$ there is necessarily some non-zero proportion of walkers in the dissociated state. Ensemble MSD is no longer well-defined, as we cannot ascribe a position to dissociated walkers. Instead of this approach, we implement a *hopping rule*, whereby a walker with $k-1$ legs whose next KMC chosen transition is to detach its one remaining leg is prevented from diffusing away from its dissociation location, and is held in place while one of its $k$ legs attaches to a local feasible site. The detachment and subsequent attachment are implemented as one KMC step, and both detachment and attachment times are added as the total hopping time. Section 4.4.3 gives a concrete mathematical description of the hopping rule in the context of the KMC simulation algorithm.

For any finite $k_S^+$ and $k_P^+$ rates, it is possible for walkers to temporarily dissociate. However, in practice when the walker has many legs, the on-rates are sufficiently fast, the legs are long, and the substrates are densely spaced, the probability of dissociation is low. Over the course of the simulations shown in Figs. 6.5 and 6.6, only 4/56000 walkers with $f < 3.0\,\text{pN}$, and 100/16000 walkers with $f \geq 3.0\,\text{pN}$ experienced any hopping event.

## 6.6 Effect of variation of number of legs and leg length

As summarized in Table 6.1 our results focus on 4-legged walkers with leg length $\ell = 12.5\,\text{nm}$, which is 2.5 times the 5.0 nm substrate spacing distance. Both leg length and number of legs can be freely varied. However, there are sensible ranges for these parameters, outside of which the motion of the walkers is not as processive, or is exceedingly slow. To be efficient molecular transport devices, walkers need to simultaneously avoid dissociation, resist the effect of forces, and remain attached to substrates near the boundary.

First, consider the number of legs, which is varied in the range $2 \leq k \leq 5$ in Figure 6.8. For the residence time bias to lead to a directional bias, we require $k \geq 2$ [112]. With few



www.manaraa.com

legs ($k = 2$), walkers are more likely to have all of their legs detached simultaneously and undergo a hopping step (Figure 6.8b). As the number of legs is increased this probability drops exponentially, as each leg's probability of detachment is approximately independent. Walkers with more legs also tend to move more superdiffusively and processively (Figure 6.8a), as they have a higher probability that at least one leg remains attached to a substrate at the boundary between visited and unvisited sites. However, walkers with many legs have a significantly smaller diffusion constant. Hence, $k = 4$ was chosen as a reasonable compromise value that prevents dissociation, maintains a strong tendency to remain on the boundary, and moves appreciably fast.

The leg length $\ell$ must be considered in relation to the substrate spacing as together these parameters determine the number of feasible sites an unattached leg can potentially attach to. The substrate spacing is constrained by limits on the sizes of molecules and how closely substrates can be arrayed on a surface. We chose 5.0 nm as a reasonable lower limit on this spacing, as it approximates the density of DNA substrates arrayed on a DNA origami [109] surface, such as the substrate density used in molecular spider experiments [82].

Figure 6.9 shows the effect of varying the leg length for 4-legged walkers while keeping the substrate spacing constant. We find that if legs are too short ($\ell \leq 5.0$ nm), the number of feasible sites is too small to maintain a superdiffusive effect. For leg lengths $\ell \geq 7.5$ nm, which is 1.5 times the substrate spacing of 5.0 nm, there is little qualitative difference in the walker motion, although longer legs do lead to a faster diffusion constant in the absence of force. Under load, however, leg length and substrate spacing should both be minimized to maximize the peak work and displacement of walkers. Longer legs allow a larger feasible region $\mathcal{F}$, leading to a larger bias in $\boldsymbol{B}$ under any non-zero load. This in turn makes it more likely for long-legged walkers to move backwards. We found that a leg length of approximately 2.5 times the substrate spacing provides a good balance between dissociation and processivity, although a full analysis of this relationship is reserved for



www.manaraa.com

future study.



www.manaraa.com

**Figure 6.8:** Simulation estimates ($N = 400$) showing the effect of the number of walker legs ($k$) on walker motion when $f = 0$. (a) The MSD is shown with 95% confidence intervals in shading. Walkers with more legs move with smaller diffusion constant when $k_{cat} = 1.0 = k_P^-$ and there is no residence time bias. However, when $k_{cat} = 0.01$, the walkers with more legs experience a stronger directional bias towards the local substrate concentration gradient and hence move superdiffusively over longer times and distances. Of the configurations studied, walkers with $k = 5$ legs and $k_{cat} = 0.01$ eventually achieve the greatest mean squared displacement. The black lines show the case where $k = 4$ which corresponds to walkers in Figure 6.2. (b) The proportion of walkers that never experience a dissociation (hopping event) depends exponentially on the number of legs. Walkers with $k = 2$ tend to dissociate over the time scales simulated ($t_{max} = 3.0 \times 10^5$). Dissociation is rare for walkers with $k > 2$, and can largely be ignored for these walkers with the kinetic parameters summarized in Table 6.1.



www.manaraa.com

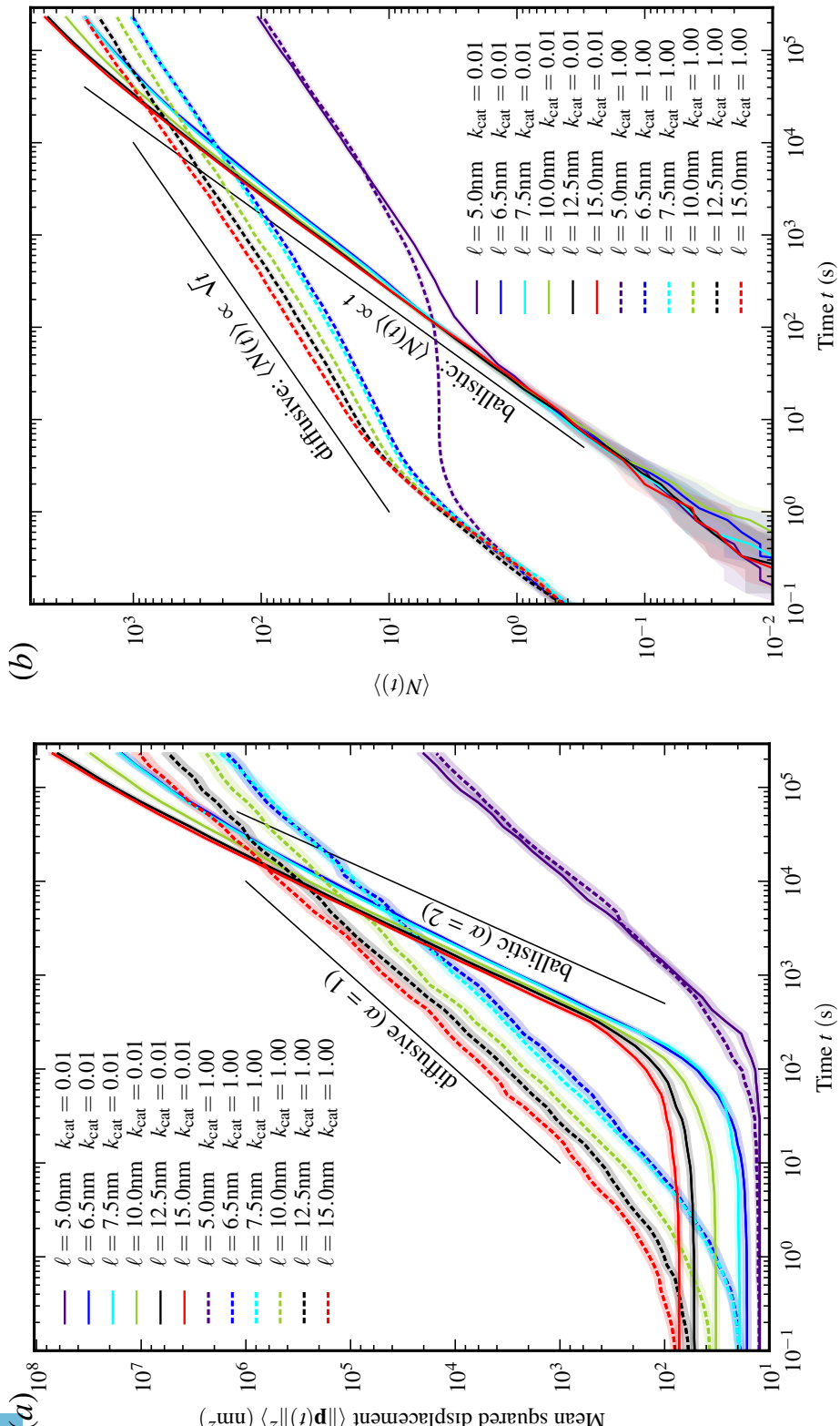


**Figure 6.9:** Simulation results ($N = 400$) showing the effect (at $f = 0$) of varying the leg length 5.0 nm $\leq \ell \leq$ 15.0 nm, while the number of legs is fixed at $k = 4$ and the substrate spacing is fixed at 5.0 nm × 5.0 nm. (a) The effect of $\ell$ on MSD is shown with shading indicating the 95% confidence interval for the mean. The effect of changing the leg length is essentially manifested as a change in the diffusion constant, but not in the qualitative characteristics of the superdiffusive motion for the $k_{cat} = 0.01$ walkers. The exception is for the very short leg length $\ell = 5.0$ nm, where the average number of feasible sites becomes so small that walkers lose their superdiffusive transport behavior (b) Similarly, the value of $\langle N(t) \rangle$ also shows a distinct difference between the behavior of the very short $\ell = 5.0$ nm legs and leg lengths that are sufficiently long to have many feasible sites available.



www.manaraa.com

## 6.7 Sensitivity to kinetic parameters

The MVRW model has too many independently variable kinetic parameters to simultaneously examine the effect of each of them on walker motion characteristics. We have chosen representative kinetics summarized in Table 6.1 to act as a reference point. Clearly, any chemical realization of the multivalent random walker model (e.g., molecular spiders) will have potentially very different (and likely much faster) rates than those we have chosen. However, it is not our purpose to model a specific chemical implementation. Instead, we show that the qualitative characteristics of superdiffusive walker motion persist over a wide range of kinetic values, as long as the residence time bias between visited and unvisited sites leads to an effective bias in the direction of the local substrate concentration gradient.

In Figure 6.10 we show that the superdiffusive behavior as quantified by MSD is persistent over an order of magnitude in variation of the $k_S^+$ rate. Indeed, even if the walker is biased 10:1 in attachment preference to products over substrates, the residence time bias of 100:1 of substrate to product binding duration is still sufficient to achieve a superdiffusive scaling of MSD over several decades in time. This robustness even to large changes in attachment rates allows us to be confident that superdiffusive behavior is a pervasive feature of multivalent random walker systems and is not critically dependent on our particular choice of attachment rates.

We show the results of varying $k_S^-$ in Figure 6.11. In other results we have assumed that $k_S^- = 0$, which is reasonable as this rate is likely to be much slower than $k_P^-$ or $k_{cat}$ for any practical enzymatic implementation of a multivalent random walker. Figure 6.11 shows that indeed the superdiffusive behavior is robust to changes in $k_S^-$, as long as it remains significantly slower than $k_P^-$ and $k_{cat}$. However, setting $k_S^- = k_P^-$ does eliminate any superdiffusive effect, as there is no longer a residence time bias between substrates and products, and the motion of the walker near the boundary is no longer biased in the direction of the local substrate concentration gradient.



www.manaraa.com

Finally, In Figure 6.12 we examine the sensitivity of the MSD to an increase in $k_{\mathrm{P}}^{-}$, and find that the superdiffusive effect is also robust to modest increases in product dissociation.

## 6.8 Effect of forces on dissociation reactions

In describing our model we show how forces affect the bimolecular association rates in Section 3.7. However, applying a load force to walkers should also affect the kinetics of the unimolecular dissociation events. From the high-level viewpoint of chemical kinetics given in Section 2.1, a unimolecular reaction depends on a molecule having enough internal energy to surmount some reaction energy barrier $U_0$, so the rate laws follow the Arrhenius formula, $k(T) \propto \exp\left(-U_0/k_B T\right)$. The effect of the force $f$ applied to the molecule is a mean change in energy of $\Delta U_f$, and the rate is modified to

$$k(T) = \nu \exp\Big((\Delta U_f - U_0)/k_B T\Big). \tag{6.4}$$

The value of the constant $\nu$ and the relationship of $\Delta U_f$ with force $f$ depend on the specific internal chemistry of the leg tethers, enzymes, and substrates [37, 76, 127], the details of which are beyond the scope of our coarse-grained walker model. We surmise that the effect of small forces is a slight increase in $k_{\mathrm{S}}^{-}$ and $k_{\mathrm{P}}^{-}$, although this change would not be uniform over all legs, as those attached to sites further in the $+\hat{\mathbf{x}}$ direction will oppose more of the load force on average than other sites. Based on Figs. 6.11 and 6.12, small increases in $k_{\mathrm{S}}^{-}$ and $k_{\mathrm{P}}^{-}$ do not qualitatively change the motive properties of the walker with regard to MSD, except when the forces are large enough so that $k_{\mathrm{S}}^{-} + k_{\mathrm{cat}} \geq k_{\mathrm{P}}^{-}$, which eliminates the residence time bias and all superdiffusive motion. Overall, these results show that even though the present formulation of the MVRW model does not describe the effect of force on dissociation rates, we expect an extension of the model including these rates to predict similar superdiffusive behaviors, as long as the forces and corresponding rate changes are small.



www.manaraa.com

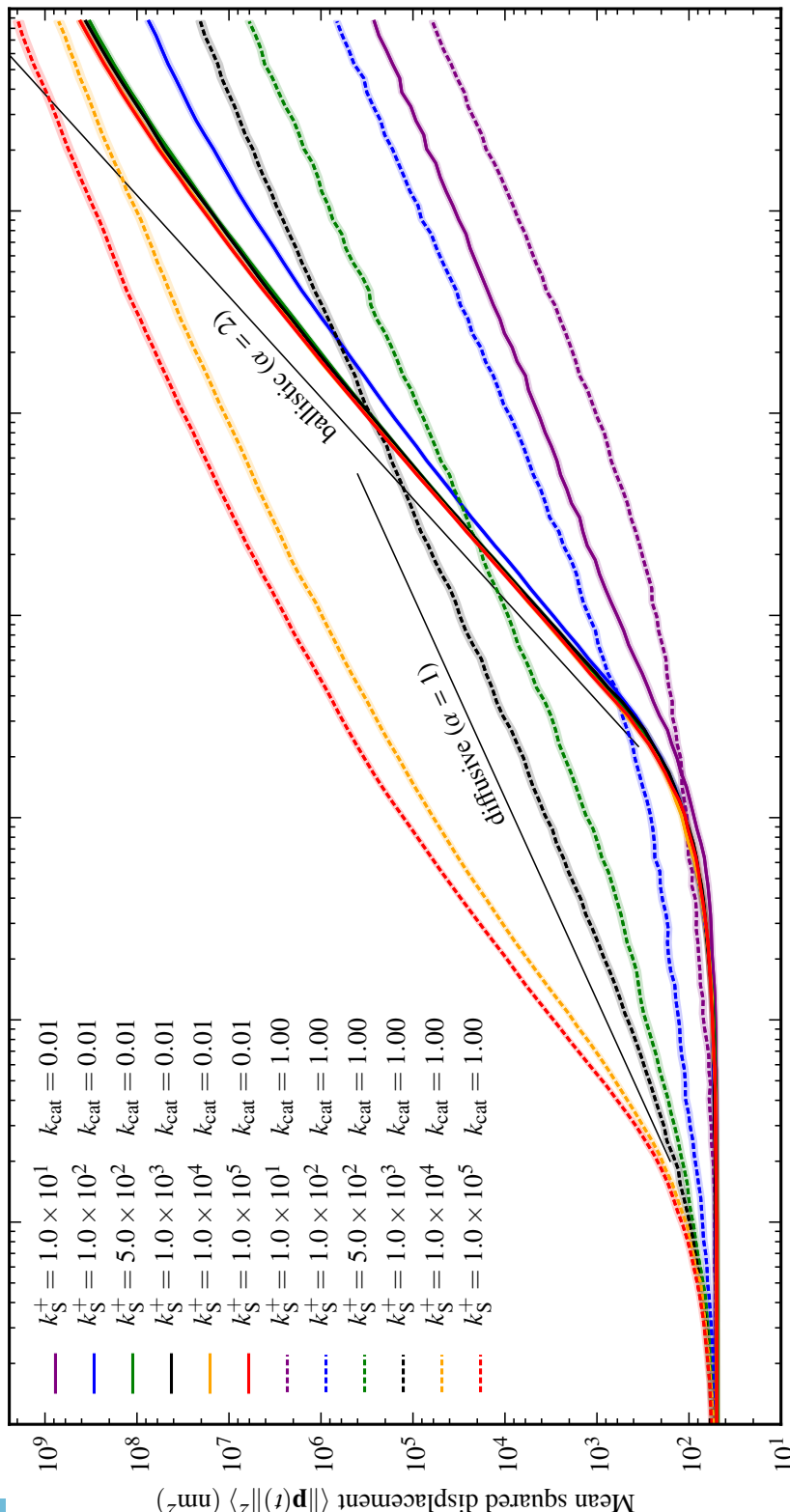


**Figure 6.10:** Simulation results ($N = 1000$) showing the effect (at $f = 0$) of varying $k_S^+$ on the MSD of walkers while $k_P^+$ is fixed at $1.0 \times 10^3$ s$^{-1}$. The black lines represent the case where $k_P^+ = k_S^+ = 1.0 \times 10^3$, which is used in all other simulations results. When $k_P^+ = k_S^+$, there is no attachment preference for a substrate over a product. An unattached leg will just as rapidly bind to a feasible substrate as to a feasible product. However, as we take $k_S^+ < k_P^+$, the walkers have an attachment bias to products, which should be expected to lead to less pronounced superdiffusive behavior. These results show that the MSD is robust to moderate changes in the on-rates. Even for $k_S^+ = k_P^+/10$, there is an appreciable superdiffusive effect when $k_{cat} = 0.01$. However taking $k_S^+ = k_P^+/100$ overwhelms the residence time bias of the walkers in the $B$ state and prevents any superdiffusive motion.



www.manaraa.com

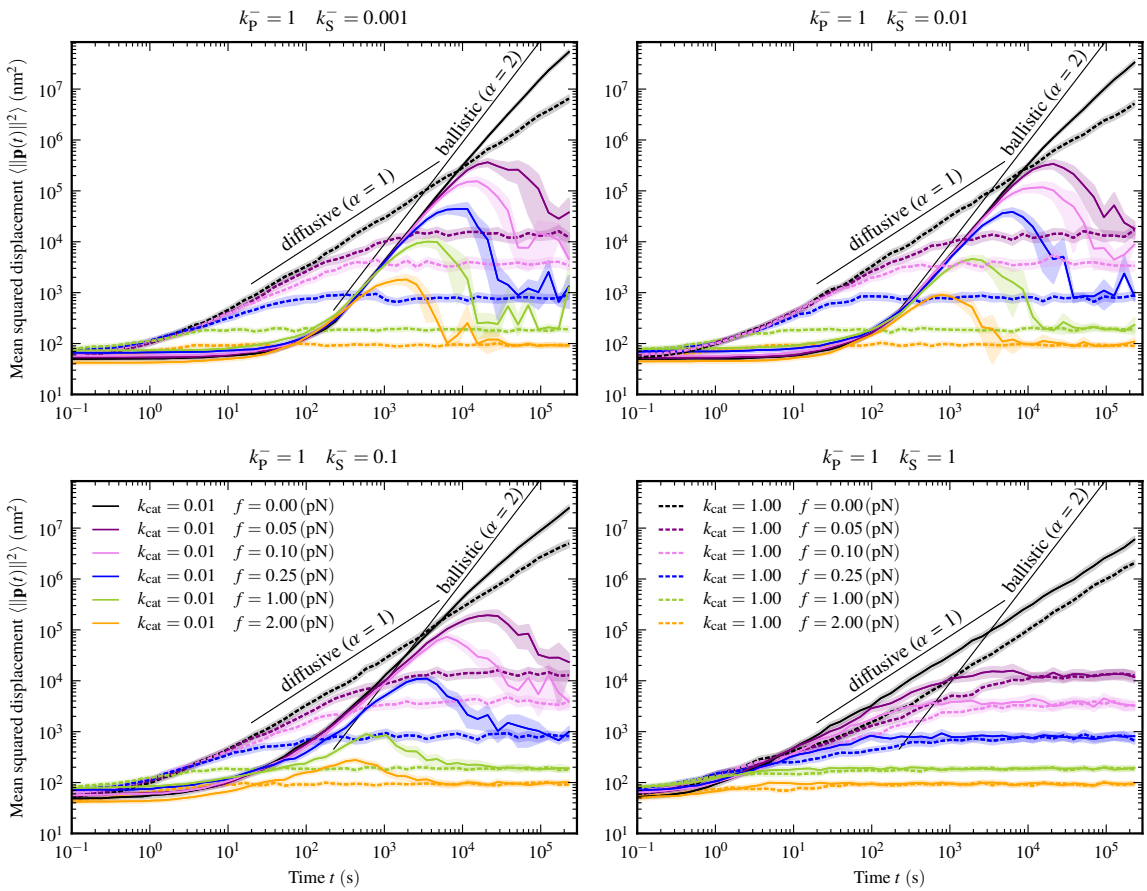


**Figure 6.11:** Simulation estimates ($N = 400$) of the mean squared displacement of the walkers as $k_S^-$ is varied. Shading shows 95% confidence intervals for the mean. Each subplot shows the same 12 walker configurations, varying only the values of $k_S^-$. Fiducial lines for diffusion and ballistic motion are shown in the same position on each subplot for reference. These data can be compared with Figure 6.5 which shows the case $k_S^- = 0$. The value of $k_S^-$ determines the rate of detachment without enzymatic conversion of the site to a product. As long as $k_S^- + k_{cat} < k_P^-$, there remains a residence time bias, and our results show that walkers with $k_{cat} = 0.01$ move superdiffusively for $k_S^- < k_P^- = 1$. This shows that the qualitative behavior of the walkers is unchanged for small variations in $k_S^-$, and the choice of $k_S^- = 0$ in the model is appropriate as small values of $k_S^-$ do not significantly affect the walker motion. However, when $k_P^- = k_S^- = 1$ the superdiffusive motion is eliminated, as there is no longer an effective residence time bias between visited and unvisited sites.



www.manaraa.com

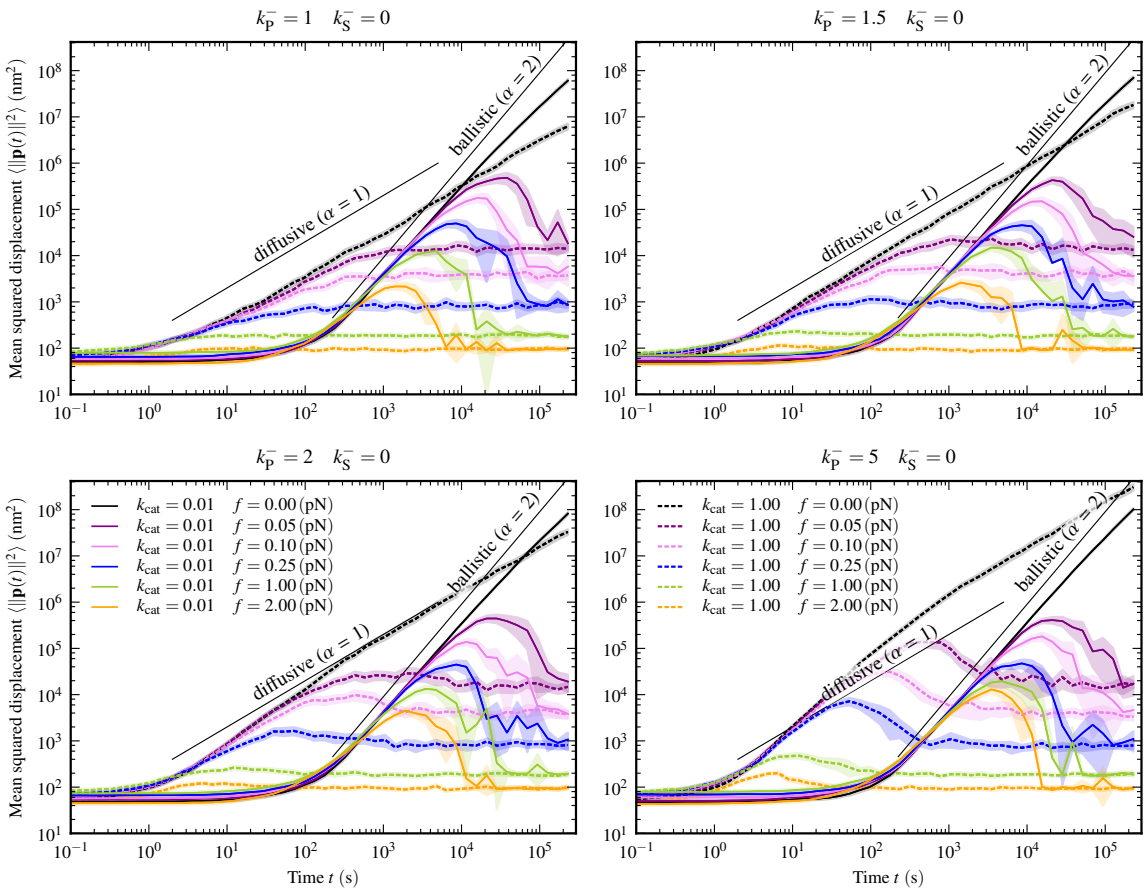


**Figure 6.12:** Simulation estimates ($N = 400$) of the mean squared displacement of the walkers as $k_P^-$ is varied. Shading shows 95% confidence intervals for the mean. Each subplot shows the same 12 walker configurations, varying only the values of $k_P^-$. Fiducial lines for diffusion and ballistic motion are shown in the same position on each subplot for reference. The subplot with $k_P^- = 1$ corresponds to the same parameters used in Figure 6.5. Increasing $k_P^-$ corresponds to faster product dissociation, and leads to a larger MSD. In these plots we maintain the values for $k_{cat} \in \{0.01, 1.00\}$. Thus, as $k_P^-$ is increased, the $k_{cat} = 1.0$ walkers are no longer the "zero energy" case, and even the $k_{cat} = 1.0$ walkers also begin to move superdiffusively for significant distances when $k_P^- = 5$. We chose the default value of $k_P^- = 1$ to serve as a reference point to compare with the other rates. In practical implementations of a multivalent-walker system, such as the molecular spiders, the value of $k_P^-$ is likely to be much faster than $1(s^{-1})$, which will accordingly lead to faster walker motion. However, these results show that the qualitative characteristics of walker motion depend on the *relative* values of the kinetic rates, and not on their absolute values.



www.manaraa.com

# Chapter 7

# Multivalent Random Walkers are Molecular Motors

A *molecular motor* is a nanoscale device capable of transforming chemical free energy into useful work and directed motion. Molecular motors can function as cargo transport devices, enabling superdiffusive transport of materials and information in synthetic nanoscale systems. The results described in Chapter 6 show that multivalent random walkers are molecular motors. A walker functions as a superdiffusive transport system that transforms the chemical free energy available in surface-bound substrates into directed motion along prescriptive tracks and this motion persists even when opposed by a load force, allowing a walker to transduce chemical energy into mechanical work. Hence, a MVRW is able to extract order out of the otherwise disordered random collisions and molecular vibrations that drive all molecular motion at the nanoscale, and it can use this energy to direct molecular motion in ways not possible at equilibrium.

In Section 7.1, we show that the superdiffusive behavior of walkers with $k_{\text{cat}} < 1$ can be understood by observing that the substrate molecules are the sole source of Gibbs free energy available to the walker. The ability of the walker to move in a biased, directional way depends on its maintaining a steady supply of local substrate molecules. The substrate fuel is, however, a locally limited, immobile resource, and once a region of the walker surface has been depleted of substrates, the walker loses any ability to move directionally or superdiffusively in that locality. Thus, walker motion will be qualitatively different in regions that have already been visited and are devoid of substrate fuel. From this perspective, we describe a MVRW system as moving between two metastates that are distinguished by the presence or absence of a local substrate concentration gradient. This concentration gradient results from an emergent asymmetry in substrate concentration at the boundary between visited product sites and unvisited substrate sites. In Section 7.1.3 we show that as long as the walker stays close to this boundary it moves ballistically away



www.manaraa.com

from the origin as it consumes the substrate fuel. However, if the walker moves away from the boundary and over previously visited products it cannot help but to move diffusively as it lacks any source of energy. By viewing the MVRW system as alternating between ballistic and diffusive modes of operation, we can explain both the transient superdiffusive motion of walkers and the asymptotic decay to ordinary diffusive motion.

Given the many potential nanoscale applications for molecular spiders, it is interesting to see that the MVRW model predicts that walkers move superdiffusively over significant times and distances, even in the presence of a force. This motion is not a product of differing $k^+$ rates, but is rather of a more subtle nature, emerging from the interaction of a residence time bias, local substrate anisotropy, and constraints imposed by multiple legs attached to a single body. In other words, multivalent random walkers are able to function as molecular motors because they act as Brownian ratchets (Section 7.2). The physical motion of the walker is solely the result of random, thermally-driven molecular motions, but the asymmetrical chemical kinetics allow this motion to be rectified, resulting in a net bias away from previously visited sites.

We can better understand the significance of the force-generating ability of multivalent random walkers by comparing them to natural cellular molecular motors that are essential to the complex molecular tasks necessary to sustain life (Section 7.3). These natural motors, such as kinesin, dynein, and myosin [122] walk along *oriented* tracks, consuming chemical energy in the form of ATP and converting it to mechanical energy that can be used to do work against external load forces [29, 119]. Natural molecular motors rely on complicated non-local conformational changes to couple the binding of fuel with the kinetics of track binding [120], and use this conformationally-mediated chemomechanical coupling to coordinate their processive hand-over-hand walking gait [121]. These mechanisms make natural motors efficient but also make them hard to mimic in synthetic systems.

In contrast, the MVRW model shows that mechanisms for designing molecular motors



www.manaraa.com

exist without the need for chemomechanical coupling, conformational coordination, rigid walking gaits, or inherent orientation of walker or track. Multivalent random walkers, like natural molecular motors, are Brownian ratchets [44, 99] that rectify random molecular motion into ordered work and directional transport. Both MVRWs and natural motors achieve this rectification by utilizing the chemical free energy of a substrate fuel. However, the mechanisms by which MVRWs do this are significantly different from natural motors. Unlike kinesin and other natural motors, MVRWs move over arbitrarily arranged 2D tracks, and are able to do so without inherent orientation or structural asymmetry. The gaits of a molecular walker are uncoordinated and acyclic, yet the irreversible modification of surface sites causes an emergent asymmetry in local substrate concentrations that is able to bias the motion of walkers, allowing them to move directionally along prescriptive landscapes. The structural and chemical simplicity of MVRWs is one of their most important properties as it means that the conceptual functionality of a molecular spider is independent of the specific enzyme/substrate system used in their implementation. Hence, multivalent random walkers provide a different perspective for better understanding what structures, properties, and mechanisms are *minimally necessary* to turn a molecular walker into a molecular motor.

## 7.1 Mechanism of superdiffusive motion

The results of Chapter 6 show that MVRWs can move superdiffusively in the direction of new sites even in opposition to a force. Over significant spans of time, the walkers will have effectively done work against the force as their motion is biased by the chemical energy in the sites they cleave. MVRWs operate by modifying a substrate site, leaving behind a lower energy product—an irreversible reaction. A walker starting on a substrate-covered surface is a system far from equilibrium, and consequently has the potential to do useful work as it relaxes towards equilibrium.



www.manaraa.com

### 7.1.1 The residence time bias

Under the kinetic parameters investigated (Table 6.1), there is a *residence time bias* between leg–substrate and leg–product bindings for the walkers with $k_{\text{cat}} < 1.0\,\text{s}^{-1}$. In other words, leg–substrate bindings are much longer-lived than the leg–product bindings, and legs attached to substrates constrain the motion of walkers, keeping the walker body and other legs from moving too far away from an attached substrate, until the slow catalysis kinetics finally allows the substrate-attached leg to modify the site and detach. According to Equation 3.6, the rate of detachment for a leg–substrate complex is $k_{\text{cat}} + k_{\text{S}}^{-}$, versus $k_{\text{P}}^{-}$ for a leg–product complex. For simplicity of presentation we can focus on the ratio of these relative kinetic rates, $r = (k_{\text{cat}} + k_{\text{S}}^{-})/k_{\text{P}}^{-}$.

If $r = 1$, there is effectively no difference between substrate and product; although the substrate sites are transformed to products via an energetically downhill and irreversible reaction, the distinction between substrate and product cannot affect the walker, as the kinetics of attachment and detachment are identical for the two species. Thus, a walker system with $r = 1$ is equivalent to a walker moving over an all-product surface. But an all-product surface is a system with no chemical free energy, and so a walker in such an environment can only move by ordinary unbiased diffusion. Thus, we expect a walker to move diffusively when $r = 1$. Indeed, in the results of Chapter 6, we consistently see the walkers with $k_{\text{cat}} = 1.0\,\text{s}^{-1}$ move diffusively with $\left\langle \|\boldsymbol{p}(t)\|^2 \right\rangle \propto t$ when $f = 0$.

However, when $0 < r < 1$, a leg–substrate bond lasts longer than a leg–product bond, and substrates effectively act like anchors. A leg attached to a substrate restricts the movement of the walker body and other legs until the substrate is cleaved, and the other legs are constrained to attach to feasible sites close to the substrate-attached leg. During this time, if another walker leg attaches to a product, it will quickly detach at the relatively fast rate of $k_{\text{P}}^{-}$, and it will soon be free again to attach to another feasible site in the walker's local environment. If there are any other substrates in the local environment, one of the other legs will eventually find and attach to one. Thus, the legs are in some sense attracted



www.manaraa.com

to substrates, but not because they specifically seek out the substrates or prefer them to products. Instead, the bias is more subtle, caused by a combination of the residence time bias manifested in the kinetics and the collective constraints on the legs imposed by the connection to a common body. The legs eventually find the substrates simply because if they attach to a product, they will quickly end up detaching and randomly choosing a new attachment site again and again until they find a substrate. Note that this effect is only present when the walker has more than one leg and has $r < 1$, so both of these properties are critical for spiders to move superdiffusively.

This local attachment bias, of course, also depends on the local availability of substrates. Once a leg attaches to a substrate, the site will eventually be irreversibly transformed into a product. Thus, while the legs (passively) seek out the substrates, they eventually will deplete the local substrate supply. For a small environment with a limited number of sites, substrates will all quickly be turned into products, at which point the system will be at equilibrium and the walker will move diffusively. However, with larger environments this march towards equilibrium takes a significant amount of time, and during this non-equilibrium period there is potential for superdiffusive motion and for doing physical work against a force.

### 7.1.2 Directional bias at the boundary

Now, consider what happens when the local environment has a non-uniform distribution of substrates. Suppose, as in Figure 7.1, the walker has a single leg attached to a substrate at site $s$ with location $x = 0$. The local environment of feasible sites will then consist of all sites within two leg lengths ($2\ell$) from $x = 0$. Suppose that all sites with position $x \geq 0$ are substrates and all sites with position $x < 0$ are products. Now consider what happens when the process is started. The initially attached leg will likely remain attached to the substrate for some time if $r < 1$. During this time the other $k - 1$ legs will be restricted to the feasible sites. Short-lived product attachments mean that legs will end



www.manaraa.com

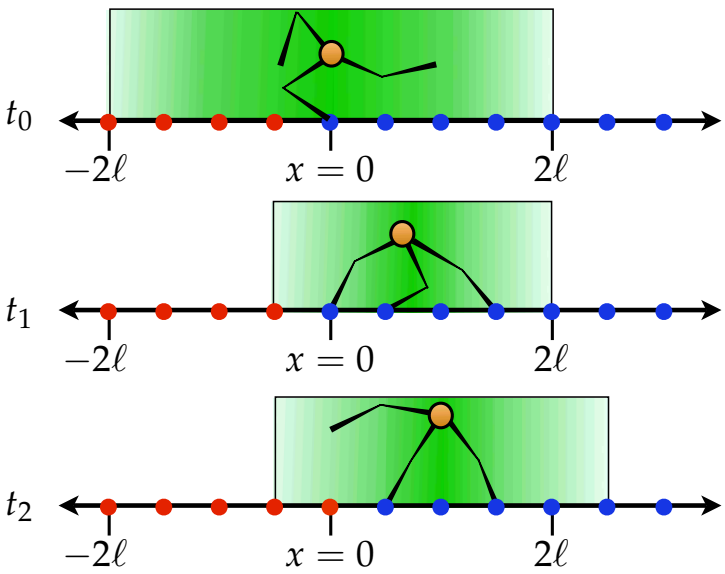


**Figure 7.1:** A residence time bias combined with a non-uniform local distribution of substrates can lead to a directional bias. There is a boundary at $x = 0$ between substrates (blue) and products (red). At time $t_0$ a single leg is attached to a substrate, and the other legs can attach to any feasible sites (shaded area). Because the leg-product pairs are short-lived, the legs are more likely to end up attached to substrates at time $t_1$. When the first leg detaches at time $t_2$, the equilibrium position and substrate boundary will move right.

up preferentially attached to substrates by the time the first leg cleaves and detaches. At this point if most of the legs are on substrates, and all of the substrates are to the right, the spider's equilibrium body position will move right. At the same time, because the site at $x = 0$ is now a product, the boundary between the substrates and products also moves right. Thus, the walker is biased towards moving right, and simultaneously shifts the biasing-inducing substrate/product boundary rightward as well. As long as the walker stays attached to substrates by the boundary, it will tend to move along with the boundary, leading to ballistic motion in the direction of unvisited substrates. However, there is still some probability that the walker detaches from all substrates and moves backwards over previously visited sites. In this case, the walker must move diffusively.

In previous work [112] we also observed significant periods of superdiffusive motion in the simpler one-dimensional molecular spider models of Antal and Krapivsky [6]. For these models, we explained this superdiffusive motion by showing that the Markov process can be viewed as consisting of two metastates (illustrated in Figure 7.2): a boundary ($B$) state where the walker is on the boundary between cleaved and uncleaved sites, and a diffusive ($D$) state where the walker is moving over previously visited sites. The walker



www.manaraa.com

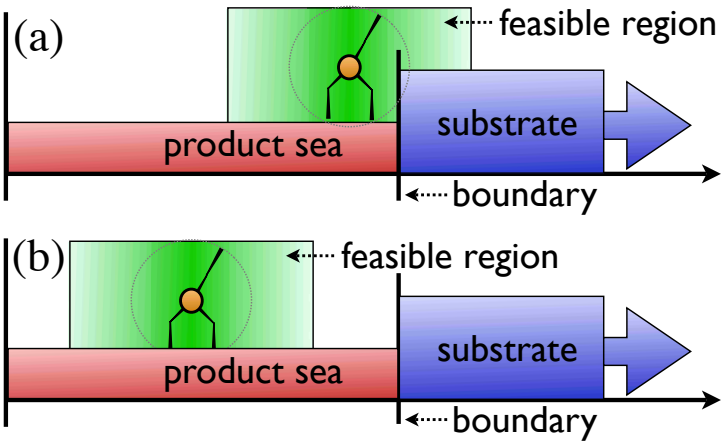


**Figure 7.2:** (a) The walker in a boundary state $B$ where it is attached to substrates on the boundary between visited and unvisited sites. The residence time bias and non-uniform local distribution of substrates gives the spider an outward bias. (b) The walker in the diffusive state $D$ where it moves over previously visited sites.

moves ballistically in the $B$ state and diffusively in the $D$ state, and the overall motion depends on how much time the walker spends in each of the metastates. Similarly, the initial superdiffusive motion in the MVRW model for $r < 1$ can be understood as the walker moving between a $B$ and a $D$ state as shown in Fig 7.2. The walker initially spends most of its time in the $B$ state, moving ballistically away from the origin in the direction of unvisited sites. However, the walker has a constant probability of falling off the boundary and into the $D$ state where it moves diffusively over previously visited sites. As the size of the region of cleaved products (the *product sea*) grows, the spider takes increasingly long to return to the $B$ state, and eventually becomes on average diffusive in the limit of long times as observed in Figure 6.2.

### 7.1.3 The boundary and diffusive metastates

The superdiffusive motion of walkers and its eventual decay to diffusion ($f = 0$) or stationary equilibrium ($f > 0$) can be understood by noting that the only source of energy available to the walkers is present in the substrate molecules, which are a locally-limited, immobile resource.

After the walker starts moving and catalyzing sites, a contiguous region of product sites called the *product sea* begins to form (Figure 7.3). At the *boundary* between the



www.manaraa.com

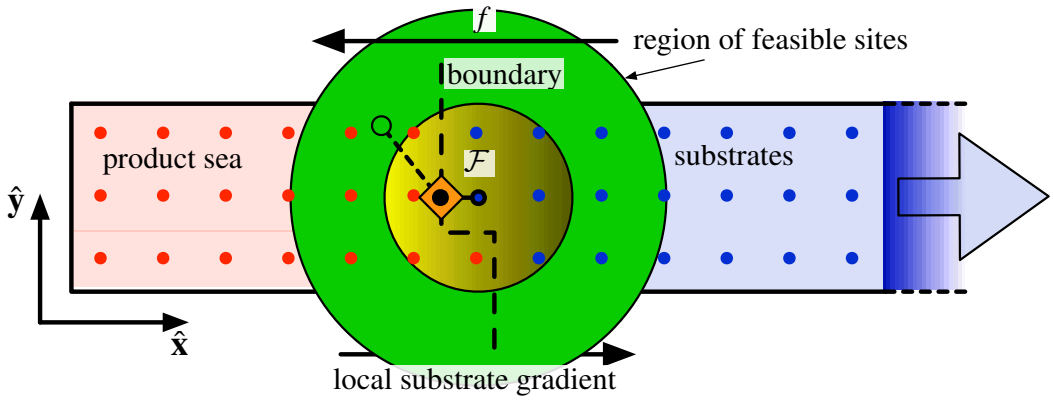


**Figure 7.3:** The irreversible catalysis of substrates to products leads to a spatial asymmetry in substrate concentration at the *boundary* between the contiguous *product sea* and the contiguous region of unvisited substrates. A residence time bias at the boundary causes a walker with $k_{\text{cat}} < 1$ to move ballistically in the direction of local concentration gradient. The boundary moves with the walker in the $+\hat{\mathbf{x}}$ direction because the legs irreversibly catalyze the attached substrates into products.

product sea and unvisited substrates, the local substrate concentration gradient is in the $+\hat{\mathbf{x}}$ direction.[1] The emergence of spatial asymmetry in concentration makes it possible for an unoriented, symmetric walker to develop a directional bias. At the boundary, a MVRW with $k_{\text{cat}} < 1$ is biased in the $+\hat{\mathbf{x}}$ direction not because the legs are more likely to attach to substrates, but because when they do attach to a substrate, they stay bound longer—there is an effective residence time bias.

A walker with $k_{\text{cat}} < 1$ is only directionally biased when near the boundary, in which case its legs irreversibly catalyze attached substrates to products, moving the boundary in the $+\hat{\mathbf{x}}$ direction as well. Thus, as argued in Section 7.1.1, as long as a walker is near the boundary, it and the substrate/product boundary move ballistically outwards, away from the origin.

The emergence of the boundary between the product sea and the unvisited substrates

[1]This is due to the semi-infinite surface configuration. For more general surface shapes and orientations, the boundary will have a different orientation.



www.manaraa.com

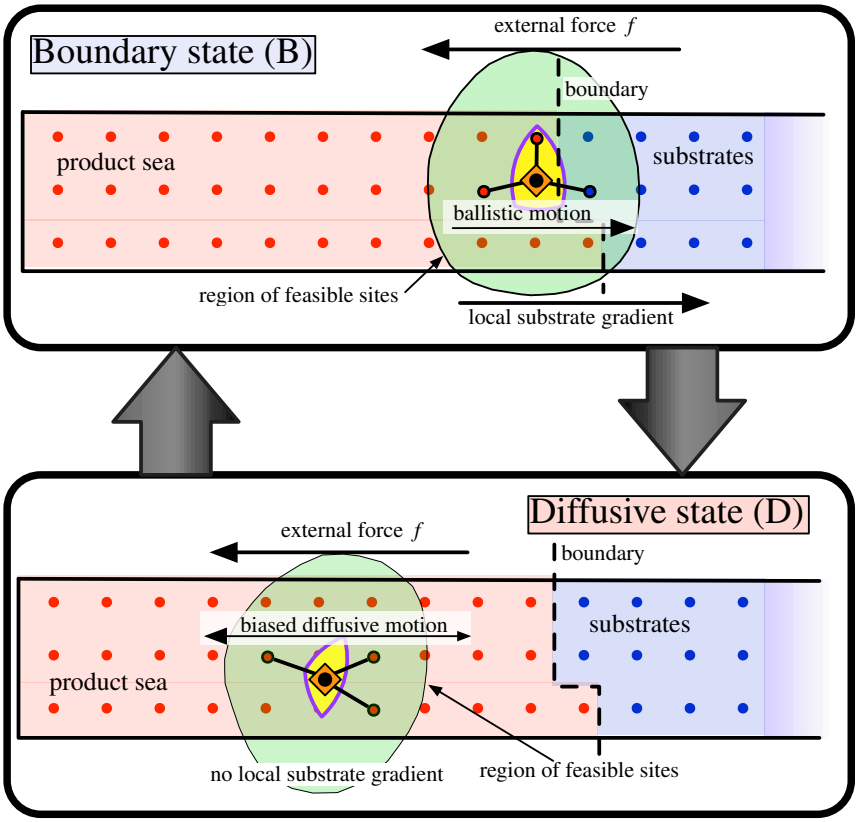


**Figure 7.4:** The walker moves between boundary ($B$) and diffusive ($D$) metastates. The walker moves ballistically in the direction of local substrate gradient when in the $B$ state, but moves diffusively over previously visited sites in the $D$ state. The walker initially spends most of its time in the $B$ state, consuming substrate fuel. However, as the product sea grows the time to exit the $D$ state increases, leading to asymptotically diffusive motion in the absence of force, and equilibrium stationary motion in the presence of force.

causes the walker to move superdiffusively, but eventually all walkers either move diffusively ($f = 0$) or move to a stationary equilibrium distribution ($f > 0$). This can be understood by decomposing the Markov process into two metastates: a boundary state ($B$) wherein the walker is attached to substrates near the boundary of unvisited sites, and a diffusive state ($D$) wherein the walker moves over the energy-devoid product sea (Figure 7.4).

When the walker is in the $B$ state it moves ballistically in the $+\hat{\mathbf{x}}$ direction, but when it is in the $D$ state it has no directional orientation, and it moves by ordinary unbiased



www.manaraa.com

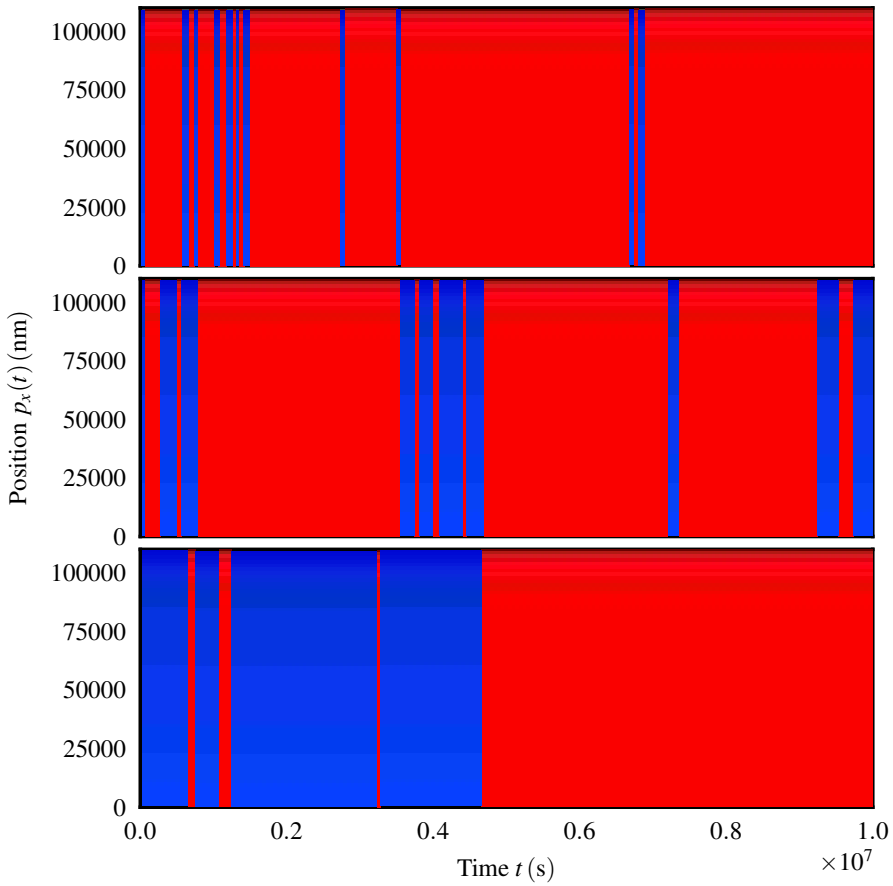


**Figure 7.5:** Typical traces of $p_x$ for a MVRW with $f = 0$ for three $k_{\text{cat}}$ values. The traces are shaded blue when the walker is in the $B$ metastate, and red when it is in the $D$ metastate. Walkers with smaller $k_{\text{cat}}$ have longer $B$ periods, but smaller velocity. The duration of $D$ periods is independent of time and grows with the size of the product sea.

diffusion for $f = 0$, or by $-\hat{\mathbf{x}}$-biased diffusion when $f > 0$. Figure 7.5 shows three typical traces of the position of individual walkers under zero force, where $B$ and $D$ periods have been shaded to show the alternation between states and the distinction between the ballistic and diffusive motion.

The probability of a walker leaving the $B$ state by moving sufficiently far in the $-\hat{\mathbf{x}}$ direction is independent of the absolute position of the boundary. Thus, the $B$ metastate is Markovian since the transition rate to the $D$ metastate is independent of how long the



www.manaraa.com

walker has been moving or the current size of the product sea. The duration of the $B$ state does, however, depend on $k_{\text{cat}}$, with smaller values leading to longer durations of ballistic motion, but at smaller velocities (Figure 7.5).

In contrast, the $D$ metastate is non-Markovian. The duration of a $D$-period depends on the size of the product sea, and hence this duration grows as the walker catalyzes more sites. In the case where $f = 0$, the time is quadratically dependent on the size of the product sea, but when $f > 0$ this dependence becomes exponential, and for sufficient forces and sufficiently large product seas, the probability of returning to the boundary once departed a significant distance becomes effectively 0. Hence, the duration of $B$-periods is constant in time, but the duration of $D$-periods grows. Eventually walkers spend nearly all their time moving over products in the $D$ state, and so approach the same equilibrium distribution as the $k_{\text{cat}} = 1$ walkers, as seen in Figures 6.5 and 6.6.

In preliminary work investigating simple 1D spiders models without force, we have shown analytically that the motion at the boundary is ballistic [112]. From results in Figs. 6.2 and 6.5 the motion in 2D (at the ensemble level) is nearly ballistic even when it opposes small forces, implying that individual walkers near the boundary must also be moving nearly ballistically.

In summary, the ability of MVRWs to move superdiffusively when in the $B$ metastate depends on three fundamental conditions.

1) *Multivalency* — The MVRW must have $k \geq 2$ legs, and these legs must be constrained so that an unattached leg cannot attach too far away from another attached leg.

2) *Residence time bias* — There must be a residence time bias between modified and unmodified sites, such that the leg-substrate binding is longer lasting than the leg-product binding. This happens when $k_{\text{P}}^{-} > k_{\text{cat}} + k_{\text{S}}^{-}$.



www.manaraa.com

3) *Irreversibility* — The irreversibility of substrate catalysis leads to the emergence of the product sea and a substrate concentration gradient at the boundary between visited and unvisited sites.

## 7.2 Brownian motors and biased transport in the MVRW model

A *Brownian motor* or *Brownian ratchet* is a physical or chemical system that rectifies random thermal energy into some form of useful work [99]. Thermodynamically, the function of such a device requires an input of free energy, or must result in a net increase in the entropy of the local environment. A principal property of a Brownian motor is its reliance on random thermal (Brownian) energy either as a means for supplying the net motion of the motor, or as a means for energetically inducing some chemical conformational change. The purpose of the energy input is to bias or *rectify* this Brownian motion in some preferred direction. From a theoretical point of view, Brownian motors are interesting as they are one of the more practical examples of systems that extract order from randomness [13]. One interpretation of the second law of thermodynamics states that the average randomness of a closed system must never decrease with time, and no Brownian motor is known to violate this principle. A Brownian motor either operates in an open environment where free energy is supplied chemically or in the form of a time-varying potential, or it operates in an environment with ample reserves of Gibbs free energy that can be used to power the motor for a sufficient amount of time.

Brownian motors have been described as a system where "mass motion is exclusively powered by thermal fluctuations, i.e., Brownian movement, but under conditions in which specific boundary conditions have been asymmetrically established at the expense of metabolic free energy." [44] In the case of multivalent random walker systems, the boundary between unvisited substrates and visited products is the asymmetry that rec-



www.manaraa.com

tifies the otherwise unbiased walker motion, resulting in biased superdiffusive motion. The metabolic free energy expended is the result of the energetically downhill ($\Delta G < 0$) substrate conversion to product. Without a strongly negative $\Delta G$, products might be transformed back into substrates by the actions of the legs. The irreversibility of substrate conversion to products is critical to the emergence of a continuously expanding product sea and the associated outward moving, bias inducing substrate concentration gradient at the boundary between this product sea and the unvisited substrates. Thus, while $\Delta G$ is not a primary parameter in the MVRW model, it does play a functional role in enforcing directionality in the system. For the results of Chapter 6 to hold, we require only that $\Delta G$ it negative enough that the substrate catalysis remains effectively irreversible.

## 7.3 Natural molecular motors

Cells are increasingly understood as crowded environments where diffusion of larger molecules is slow and constrained by complex internal structures [122]. Diffusion is often a limiting factor in chemical processes needed to maintain the biological functions of the cell. Instead of relying on random, uncontrollable diffusion to move chemicals, eukaryotic cells have developed a taxonomy of molecular motors that transport cargo along 1D polymeric tracks. These translational molecular motors consume chemical free energy and transduce it into mechanical work and directed motion.

One of the most studied natural translational molecular motors is kinesin. Kinesin molecules are incredibly efficient motors, and are critical to many cellular transport processes, including mitosis [38, 128], organelle transport [59], and signaling in neurons [50, 63]. Hence, understanding how models describe their ability to move and do work is an important reference point for developing models for other walkers in general, and multivalent random walkers in particular.



www.manaraa.com

### 7.3.1 Kinesin structure and motion

Kinesin is a two-headed protein molecular walker that moves over an oriented 1D polymeric track, called a microtubule. Each *head* of a kinesin is identical and contains two coupled binding sites. One site binds and catalyzes the breakdown of ATP, and the second site binds in an oriented manner to sites that occur at regular intervals on the oriented microtubule track. Each head is connected by a *neck linker* to a coiled-coil central *neck* which acts as the body of the walker and connects the walking heads to a cargo [120].

Increasingly complicated observational experiments have shown that kinesins move processively along a microtubule using a hand-over-hand type of gait, where catalysis of the energy-carrying ATP molecule into ADP and Pi leads to the rearward head unbinding and attaching towards the + end of the track, resulting in an approximately 8 nm step [119]. In the context of kinesin, directed motion means that a walker moves preferentially in a particular direction (the +-end of the microtubule). Processivity means that the walker makes many steps before dissociation. It has been shown that kinesins can move processively, against a force, up to a so-called stall force of approximately $5 - 8$ pN [123].

### 7.3.2 MVRWs are a fundamentally different kind of motor

We now note the significant differences between how spiders and other multivalent random walkers move, and how kinesin is thought to move.

#### Ergodicity

The most fundamental mathematical difference from a modeling perspective between the two walkers models is that of ergodicity. Kinesins, and other motors that move over periodic, translationally invariant tracks, without modifying the track, and so are described by ergodic Markov processes. They operate in a steady state, where each complete step brings the walker back to the same initial (canonical) conformational state. A step may



www.manaraa.com

change the absolute position of the walker on its track, but translational invariance means that the chemistry at this new site is not changed. In other words the translational invariance of the track means that there is no local difference in the walker's environment or chemical state, and so each step operates under the same chemomechanical conditions and at the same rates. Hence, when the states of the walker are discretized and the stochastic actions of the walker are defined by a Markov process, the process has a regular, periodic structure [8, 77, 79]. Thus, kinesin motion can be fully understood by examining those chemomechanical cycles that start and end at a particular conformational state.

Multivalent random walkers are, however, a non-ergodic system because the walker modifies the tracks over which it moves. The motion of the walker is not translationally invariant as it explicitly depends on the irreversible modification of sites by past actions of the walker. Neither does a MVRW move with the orderly and easy-to-model rigid walking gaits of kinesin; instead, it moves via a multitude of uncoordinated gaits that lead to complex, highly branched state spaces, making static analysis difficult to apply directly. It is for these reasons that kinetic Monte Carlo simulations are the primary means for analyzing MVRW behavior.

Furthermore, the non-cyclical and non-ergodic behavior of MVRWs makes it difficult to compare their motion with kinesin because many standard descriptive statistics for kinesin and other cyclical, ergodic translational processes do not make sense for MVRWs. For example, MVRWs do not have a well defined force-dependent velocity $v(f)$. A kinesin will move with the same experimentally measurable [30] and analytically predictable [9] velocity against force $f$ at any location on its track. However, the motion of a MVRW depends on its location on the track and the location of the boundary between substrates and products, and thus any velocity must be a function of time. Even reporting an ensemble estimate of random variable $v(f;t)$ as a function of $t$ is not very informative, because the ensemble behavior averages together some walkers which are in the $D$ state with some in the $B$ state, which means that $v(f;t)$ tells us little about the typical velocity (i.e., that of an



www.manaraa.com

individual walker). Hence, we employ different measurement statistics such as ensemble MSD (Section 6.1.1) and $\langle \Delta E \rangle$ (Section 6.3), as they are more appropriate for understanding biased motion in a non-ergodic system.

## Structural differences

Multivalent random walkers accomplish processive, superdiffusive transport but they do so through significantly different mechanisms from known natural molecular motors, such as kinesin. Thus, a multivalent random walker presents a different perspective on biased molecular motion and molecular motors, in that many of the chemical and structural features that are essential to kinesin's ability to act as a motor are not present in the multivalent random walker model.

- Kinesins move over regular, oriented polymeric tracks, while MVRWs move over arbitrary, unoriented 2D surfaces of substrate tracks. The MVRW tracks can be heterogeneous or homogeneous, structured or unstructured.

- Kinesin heads are oriented, with separate binding sites for the track and the substrate fuel, whose chemical kinetics are conformationally coupled [91]. The actions of the two kinesin heads are mechanochemically coupled by long-range conformational changes [120], whereby the kinetics of a single head dissociating depends on the chemical conformational state of the other head [30]. In contrast, the legs of a MVRW are unoriented and uncoupled. The conformational state of an individual MVRW leg has no influence on the chemical or mechanical actions of the other legs.

- Kinesin always moves with a rigid hand-over-hand walking gait where the leading head alternates at each step. The chemomechanical coupling ensures that the trailing head understands its orientation with respect to the track and the leading head, making it more likely to unbind and move than the leading head [121]. A MVRW,



www.manaraa.com

however, has no preferred gait, and legs are unaware of their orientation. The leg motion is independent and uncoordinated, restricted only by the finite length of legs, and their connection to a common body.

In each of these areas of difference between kinesin and MVRWs, the mechanisms employed by kinesin are more complicated to engineer from chemical building blocks, but lead to more efficient and processive motion. However, based on the simulation results presented in Section 6.3, we have shown that none of these properties are actually necessary for a molecular walker to act as a translational molecular motor, as MVRWs can still transform chemical energy into mechanical work without any of these complex structural and chemical mechanisms.



www.manaraa.com

# Part II

# The Multivalent Random Walker Model Simulation Architecture



www.manaraa.com

# Chapter 8
# Simulation Architecture

A large part of the complexity of numerical simulations is involved in storing, organizing, retrieving, and analyzing data gathered from simulations. Data often need to be accessed concurrently from multiple simulation and analysis processes, and this leads to fundamental issues in data consistency and availability.

This chapter explores the software engineering and data management issues involved in simulation of multivalent random walker models. Our MVRW simulation framework is a set of tools and libraries for distributed concurrent simulation and analysis of the MVRW model. The object-oriented framework is written in Python and built arround a core set of classes that define persistent objects allowing the manipulation of database tuples as in-memory objects. In designing our simulation framework we set three basic requirements:

(1) Simulation data should be stored in a central relational database. Issues of data consistency and correctness in the context of distributed, concurrent simulation and analysis processes should be handled using the built-in transactional mechanisms provided by the relational database.

(2) The simulation environment should use an object-oriented strategy so that we can take advantage of inheritance to express the relationships between various simulation objects and provide for code reuse.

(3) The simulation architecture must provide for management of large numerical datasets, providing fast access, but also maintaining data consistency under concurrent access patterns.

Requirements (1) and (2) lead us to using object-relational mapping (ORM) techniques to allow the mapping of class hierarchies to sets of relations in a relational database. ORM



www.manaraa.com

allows objects to be made persistent and available to concurrent distributed access by storing object state as tuples in the relational database. There are, however, fundamental issues when using relational databases to store objects as tuples, as the semantics of the relational model differs significantly from the semantics of object-oriented languages. To address these issues, we have developed an ORM system called the *natural entity framework*, which we describe in full detail in Chapter 9. Specifically, we use the SQLAlchemy ORM software for Python, and our natural entity ORM framework builds on top of the access layer provided by SQLAlchemy.

In the remainder of this chapter we address the issues of the MVRW simulation architecture that are orthogonal to the ORM topics covered in Chapter 9. In Section 8.1 we discuss large data storage in the context of the goals of requirement (3), allowing us to provide secure, fast, and highly structured storage for large numerical simulation datasets.

Finally, we present an overview of random number generation in the MVRW simulation framework in Section 8.2. Random number generation is an issue that must be dealt with carefully in any Monte Carlo simulation, but becomes more complex in distributed, parallel simulations such as those we employ for the MVRW KMC simulation. We designed the MVRW simulations to use parallel random number generation strategies that allow a single master stream of random numbers from a single initial seed to be used to generate an arbitrary number of independent random number streams, which can each be used to generate a separate parallel KMC simulation trace.

## 8.1 Large numerical data storage

The MVRW simulations produce large amounts of measurement data that need to be organized and recorded. The application structure of the MVRW simulation relies on object-relational mapping (ORM) (Chapter 9) to store persistent data as objects in a relational database. Relational databases provide the transactional isolation and consistency guaran-



www.manaraa.com

tees that make the distributed simulation architecture possible. They also provide sophisticated indexing and querying techniques that make retrieving objects easy and fast. However, to achieve these goals the relational database model has a very rigid data model. As explained in detail in Section 9.2.1 a relation or table in a relational database is a collection of tuples, and each attribute or column of a table corresponds to a atomic (unstructured) data type. Specifically, this implies that a tuple cannot contain attributes of array type, as this violates what is known as the 0-th normal form in relational models [26, 33]. In the case of the MVRW simulations, some generated data take the form of a large numerical array, which must be correctly and consistently associated with the database tuple describing the simulation object.

### 8.1.1 Storage options for numerical arrays

A strict adherence to relational design would dictate that each array-like attribute is translated to a relation with a foreign key constraint that references back to the associated relation's primary key. This array representation is not practical or efficient for large numerical datasets where fast sequential access to array elements is needed. Understanding this limitation, modern databases, such as PostgreSQL, provide a more efficient array storage mechanism by extending the relational model (and violating normal form) to allow attributes of array type. This array data representation has the advantage that the semantics of the array type attributes are recorded in the database directly, so the database can provide extra functionality for storing, querying, and modifying data within the array.

Alternatively, PostgreSQL provides types for large binary data, allowing arbitrary data to be stored directly in tuple values. Python provides a `pickle` module (and the faster `cPickle` version) that can serialize nearly any Python object into a binary representation which can then be stored directly in a binary-typed relational attribute. The SQLAlchemy ORM framework provides column types for managing the storage of object attributes of arbitrary Python type as binary attributes in PostgreSQL via the `cPickle` Python module.



www.manaraa.com

This method also has the advantage that the database holds all simulation data directly and can manage concurrent access in a transactionally secure way. However, unlike an array-typed relational attribute, PostgreSQL has no semantic understanding of the structure of the data stored in a binary type attribute, and cannot provide any useful functionality to manipulate the data within the SQL language.

Finally, we can bypass the database and store the array data in external files. In this strategy the database tuple contains a file name or some other unambiguous external resource key that can be used by the simulation software to retrieve the external array data. The array data on the file system are no longer protected from incompatible concurrent access patterns by the database, so extra care must be taken in writing applications. However, avoiding the database overhead can often be worth the extra data management complexity. These external data files could contain serialized Python objects via the `cPickle` module, or they could be structured in a more organized format, such as HDF5.

## HDF5 data files

Hierarchical data format 5 (HDF5) [42] is a data file format developed with scientific and numerical datasets in mind [28, 52, 108]. Like XML, HDF5 provides a structured format rather than a flat file, but the HDF5 data layout and library access is optimized for multidimensional numerical arrays. Each HDF5 file can also store relation-like tables, key/value attribute pairs, and arbitrary binary data, all of which are arranged in an internal hierarchical file-system-like structure. The PyTables module provides fast Python access to HDF5 files, allowing their array data to be read from and written to `numpy` multidimensional array objects in Python [3].

The primary drawback of the HDF5 data format is that it lacks any concurrency control. This allows the HDF5 libraries to be small and fast, but can lead to catastrophic data loss as there are no safety controls for files that are accessed concurrently while they are being written to.



www.manaraa.com

### 8.1.2 Access speeds for large data sets

In order to select the best data representation for the array-like data stored in the MVRW simulation objects, we devised a simple test case. Using the SQLAlchemy ORM we create a class of persistent objects with a `numpy` array-type attribute, and store an array of N 64-bit floating point values. We measure the time to create a new object and save the array, as well as the time to retrieve, read and sum all of the data in the array. Array elements are read in after being invalidated, so that the ORM software is forced to reload the data from the database, or from disk. However, we do not control for caching of data in memory by the database or by the Linux file cache.

As shown in Figure 8.1, for each of the following representation methods we measure and report the mean execution time for 50 trials of reading and writing array data of size $1 \leq N \leq 10^6$:

1. Using SQLAlchemy and PostgreSQL
    a. As a PostgreSQL column with Array(double) type
    b. As a PostgreSQL LargeBinary column storing the pickled array using the binary `cPickle` protocol number 2, and the SQLAlchemy `PickleType`
    c. As a PostgreSQL Text column storing the array using the text-format `cPickle` protocol 0, and an SQLAlchemy extension of a `TextPickleType`
2. Using the cPickle module directly to read/write array data to a file
    a. Using a local file
    b. Using a remote file shared over NFS
3. Using PyTables to access the data in an external HDF5 file, where data is stored in a PyTables `EArray` type
    a. Using a local HDF5 file



www.manaraa.com

b. Using a remote HDF5 file shared over NFS

Our results show that HDF5 files are by far the fastest representation method for arrays larger than $10^4$ elements, which is the majority of data stored in the MVRW simulations. Hence, we made the decision to store all non-atomic data in an external HDF5 file associated with each individual persistent object. The ability to structure the HDF5 file internally as a file system makes it easy to store several array-valued attributes of an object in a single HDF5 file. The inability to allow concurrent access to the file, however, precludes the possibility of using one file per persistent class, rather than the one file per object strategy we employ.

The relative overhead of accessing files remotely over NFS is only a factor of 2-4 for HDF5 files, with the penalty decreasing for larger files. This is acceptable for our MVRW application as the convenience of uniform file availability over network-attached storage eliminates many implementation complications.

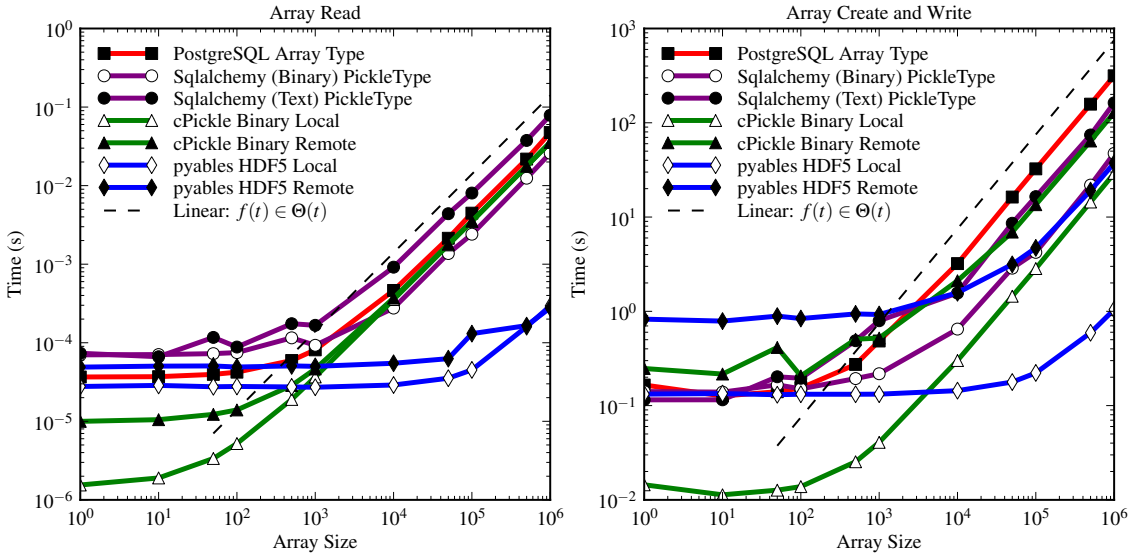


**Figure 8.1:** The read/write speeds for accessing a single large floating point array stored as a property of a persistent class using various data representation methods.



www.manaraa.com

Finally, we note that for applications with even larger data set sizes than are necessary for MVRW simulations, HDF5 files will provide further advantages. Array sizes much beyond $10^8$ become impractical for in-memory storage. At this size, the HDF5 library's B-tree index of blocks allows PyTables to quickly index into large arrays stored on disk, eliminating the need to store an entire array in memory at once as any pickling-based strategies must.

## 8.2 Random number generation

Random numbers are a fundamental resource for all Monte Carlo algorithms. While almost all proofs of correctness and complexity for Monte Carlo algorithms assume that numbers can be drawn uniformly at random over some interval, such a resource of truly random numbers is normally not available on most computers. Fortunately, there are many efficient pseudorandom number generators available that have all the distribution and correlation properties necessary for Monte Carlo techniques.

A pseudorandom number generator (PRNG) is a deterministic algorithm that produces a sequence of seemingly random numbers. A PRNG maintains a fixed-size internal state $s$ which is used to generate the random numbers. At step $i$, the PRNG generates the random number $x_i = f(s_i)$ and new state $s_{i+1} = g(s_i)$. Because the functions $f$ and $g$ are deterministic, whenever $s_i = s_j$ all subsequent states will also be equal. One of the most useful features of pseudo-random number generators is the deterministic nature of their output. Determinism makes the processes of check-pointing, verifying, and debugging Monte Carlo code much easier.

Because of determinism, a PRNG must have the state initialized to $s_0 = h(\theta)$, where $\theta$ is called a seed and is typically an integer or array of integers, and $h$ is some method that ensures that all possible seeds lead to well chosen starting values. The proper choice of initialization function $h$ is essential for Monte Carlo experiments where many independent



www.manaraa.com

runs of the same code will be run with different starting seeds. The sequences generated with different seeds need to be uncorrelated, which means all starting states must be different.

### 8.2.1 Leapfrogging for parallel random number generation

A consequence of the finite size of the states $s_i$ is that any PRNG must eventually revisit some previous state. In other words, there is a finite period $p > 0$ such that $s_i = s_{i+kp}$ and hence $x_i = x_{i+kp}$ for all $k \geq 0$. This must be taken into account for long-running simulations.

A Monte Carlo simulation that will be run $m$ times, using up to $n$ random numbers on each run should have the property that all $mn$ states of any run at any time will be different, and the sequences of random numbers generated should be independently and identically distributed and uncorrelated both within and between sequences. For most Monte Carlo simulations, including those used in the MVRW simulation, only the first few decimal places of the random floating point numbers are important, so distribution properties of the small-order bits are largely irrelevant.

In the MVRW simulation, random numbers are used in two algorithms: (1) the kinetic Monte Carlo simulation of the MVRW Markov process, and (2) The Metropolis-Hastings sampling of the body's equilibrium position. In order to prevent the possibility of overlap between the random number sequence between two different random number sequences chosen using arbitrary seeds, we use the leapfrogging strategy of parallel random number generation as implemented in the *Tina random number generator (TRNG)* library [10].

The leapfrogging method allows a single seed value $\theta$ to be used to simultaneously initialize an arbitrary number, $m$, of parallel random number streams of indefinite length [11]. Leapfrogging works by modifying the functions $f(\cdot)$ and $g(\cdot)$, so that they *leap* ahead by $m$ iterations of the PRNG sequence, defining $\widehat{f}(s_i) = f^m(s_i)$ and $\widehat{g}(s_i) = g^m(s_i)$. Then from



www.manaraa.com

the single random seed $\theta$, which gives initial state $h(\theta) = s_0$, we simultaneously initialize $m$ random number sub-streams, each starting at $s_i$ for $0 \leq i < m$, and each advanced using the leapfrogged functions $\widehat{f}$ and $\widehat{g}$. For arbitrary PRNGs, it may be expensive to compute $\widehat{f}$ and $\widehat{g}$, but for the restricted class of linear congruential generators [75], this can be done quickly as a precomputation step, allowing leapfrogging to be nearly as fast as the single-stream version of the PRNG [11, 86]. The use of leapfrogging guarantees that no two PRNG streams used in the simulation overlap at any point. Then, the same mathematical guarantees of PRNG quality for single-threaded applications [71] also apply to the distributed, parallel simulations, preventing the types of correlation problems that have been shown to lead to inaccuracies in other KMC simulations [39].

Our MVRW simulation framework provides a Python module with an interface into the TRNG library, allowing access to the same PRNG leapfrogged streams in both C and Python. Thus, for the MVRW simulations we use a single seed for all the KMC simulations that are used for investigating a single set of model parameters, and a single seed for all the MH simulations used to precompute transition rates for a particular surface and spider configuration.



www.manaraa.com

# Chapter 9

# Object Relational Mapping and The Natural Entity Framework

In order to address the fundamentally important issue of object identity and uniqueness in object relational mapping we devised a new ORM strategy we call the natural entity framework. We use the data uniqueness and consistency guarantees provided by the natural entity framework to allow the built-in uniqueness constraints provided by relational databases to be enforced within the OO program runtime environment. This allows us to prevent erroneous duplication or loss of data due to violation of value-based uniqueness constraints on the persistent objects that represent simulation constructs and store simulation data. This material is based on joint work with David Mohr and Darko Stefanovic [92].

## 9.1 Introduction

In an object-oriented (OO) language, data are represented as objects, but objects are transient—they have no persistence outside a particular process or between subsequent executions of a program. To make the data persistent and accessible for concurrent processes in a structured form, an *object-relational mapping (ORM)* can be used to store objects as tuples in a relational database.[1] An ORM is a method for translating between a data model expressed as a class hierarchy and a data model expressed as a relational schema. ORM software packages allow a program to create, read, update, delete, and query objects stored persistently in a relational database using object and class methods of an OO programming language.

[1]There are other possibilities such as using a persistent object store and a programming language that supports persistence natively. Without going into the merits of different approaches, we concentrate on ORM because of its widespread use.



www.manaraa.com

Designing an ORM presents many challenges because the object data model and the relational data model differ profoundly in how they represent, store, and access data. We focus in this work on just one facet of the mapping between the models: the concept of *identity and uniqueness*. Both data models are used to abstractly represent sets of physical or conceptual entities. An entity has multiple properties; the values of these properties may affect entity identity and entity uniqueness. However, the concepts of identity and uniqueness have different semantics in the object model and in the relational model [66].

In relational models uniqueness is a value-based notion defined by relational keys. A *key* is a minimal set of attributes (columns) of a relation that uniquely identifies a particular tuple (row). It can be a *surrogate key*, an artificial value introduced solely to distinguish tuples; or it can be a *natural key*, consisting of attributes that correspond to meaningful, real-world, properties of the entities. The attributes in a natural key represent those properties of an entity that define its identity and uniqueness in the context of the application and are well-known to the users of the entity. A natural key is a concise description that can be used to query for the existence of a specific individual entity. Every relation must specify a *primary key*, which is used as the default identifier for a tuple. For practical reasons this is often a surrogate key. However, when a natural key exists, it often makes sense to declare its existence as well by enforcing a uniqueness constraint on the natural key attributes. This prevents the database from maintaining two copies of data that represent the same entity. Additionally, declaring a natural key results in the database maintaining an index on the natural key attributes, which allows queries involving the natural key to be optimized [56].

In contrast, in object models value and identity are independent. While an OO execution environment enforces the uniqueness of object identities, this imposes no constraints on the values of objects. Hence, when real-world entities are represented by objects, there can be many distinct objects having the same values for a set of natural attributes and thus representing the same entity. There are no mechanisms to prevent this error-prone



www.manaraa.com

duplication of entity representations, and typically no universal mechanism to query for the existence of an object based on its value.

This fundamental difference in how uniqueness and identity are defined in relational databases and in OO programming languages leads to problems when data representing real-world entities are made persistent with a relational database, but are operated on as in-memory objects. If there are multiple in-memory objects all denoting the same entity, which object represents the true current state of that entity, and which one corresponds to the database's current state, i.e., the tuple representing the entity? This question becomes even more confusing when there are multiple execution contexts operating on entities concurrently.

To properly model the concept of entity uniqueness and identity at both the object and the relational level, we propose a new framework of constraints and semantics for object construction and interactions that can be enforced in modern ORM systems and strongly object-oriented languages. Our *natural entity* framework provides a base class `NaturalEntity` with the functionality described in the remainder of this chapter. Natural entities are persistent objects in an OO execution environment that directly enforce value-based uniqueness constraints on natural attribute values. Other ORMs allow natural keys and uniqueness constraints to be declared on the relational model, but they do not enforce these constraints on the object model, or in the inheritance hierarchy. Making these constraints explicit allows persistent objects to more directly represent the semantics of relational tuples used to store their state. This simplifies the programmer's conceptual model and reduces potential problems with concurrency, entity identity, and uniqueness.

In contrast to creating regular objects, there is overhead when checking for value-based uniqueness, but this overhead is not higher than manual enforcement of uniqueness. It should be stressed that the proposed natural entities are otherwise normal objects that exist alongside, and interact with, other objects, and that they can be queried and used polymorphically. Hence, the natural entity framework does not reduce the expressiveness



www.manaraa.com

of the OO language, and a programmer is free to represent entities using persistent objects that do not enforce uniqueness constraints, or using regular non-persistent objects. However, only through the use of the natural entity framework can the programmer maintain the value-based uniqueness constraints for in-memory objects.

The primary contribution of the natural entity framework is that it allows the ORM to manage and enforce value-based object identity and uniqueness on in-memory objects. These value-based constraints match the constraints imposed by natural keys on the relations that store the persistent state of the natural entities. Thus the object model for natural entities is modified to more closely match that of the relational model.

This framework provides several advantages: (1) natural entities have a strong concept of value-based identity and uniqueness, accessible through object attributes and methods that prevent multiple in-memory objects from representing the same conceptual entity (Section 9.3); (2) the ORM can use an identity map to provide fast value-based queries for in memory objects and a uniqueness constraint to provide fast queries for archived objects (Section 9.4); (3) natural entities have constructor methods that automatically manage the uniqueness constraints for in-memory objects and disambiguate object construction from object retrieval (Section 9.5); and, (4) natural entities inheritance hierarchies can be mapped automatically to a relational schema that uses the appropriate constraints and relations necessary for maintaining natural key uniqueness constraints and for allowing polymorphic queries (Section 9.6).

Given these features, the natural entity framework provides functionality that is lacking in modern ORM systems and presents an often applicable abstraction that is easy to understand and implement, allowing the programmer to spend more time on solving the actual problems at hand.



www.manaraa.com

## 9.2 Background

To be specific about how the concept of uniqueness constraints is implemented, here we summarize the terminology used for relational models and OO programming languages.

### 9.2.1 Relational model

A *relation* is a tuple of attributes denoted $R = R(A_1, \ldots, A_n)$. The attributes come from some domain $\mathbf{A}$, and each attribute $A_i$ has a type $\tau_i$, (written $A_i : \tau_i$), where $\tau_i \in \mathbf{T}$ for some set $\mathbf{T}$ of basic types. For brevity we omit type signatures where they are not essential to the discussion. A *relation instance* is a set of tuples from the domain $(A_1 \times \ldots \times A_n)$ that represents the current factual state of the relation. When it is not otherwise confusing, the term *relation* is used to describe both the relation's schema (attributes, types, and constraints) and its time-varying instances (the tuples and their values). In the concrete context of a relational database, a relation specifies the names and types of the columns of a table, and an instance specifies a set of table rows and their values.

A non-empty set $k \subset \{A_1, \ldots A_n\}$ is a key of relation $R(A_1, \ldots A_n)$ if for any instance of the relation, the value of the attributes in $k$ uniquely determines a tuple and no proper subset of $k$ is also a key. Thus, a key is a minimal set of attributes that can be used to define the identity of a tuple. A relation may have many keys. A key is *simple* if it consists of a single attribute, otherwise it is *compound*. Each table must have a *primary key*, which is used as the canonical set of attributes for identifying a row for the purpose of database operations and references between tuples of relations. Primary key attributes are underlined in the notation for a relation to highlight their role (e.g., $R(\underline{A_1}, \underline{A_2}, A_3)$ has a primary key $\{A_1, A_2\}$.) Associations between relations are expressed with a *foreign key constraint* that restricts a set of attributes to values that come from the relational instance state of a separate set of attributes that form a key [26].

A *relational schema* is a set $\mathbb{R} = \{R_1, \ldots, R_m\}$ of relations along with constraints. A



www.manaraa.com

relational database provides a set of types and mechanisms to define relational schemas over those types. It maintains instances for each relation that obey all the restrictions and allows queries to create, read, update, and delete tuples.

### 9.2.2 Object model

An *object* lives in memory and has identity, type, state, and behavior. An object's state is given by the values of a collection of named attributes that come from a set of types $\mathbf{T}'$[2]. In strongly object-oriented languages, objects have a concept of identity independent of their attribute values or addressability [70]. This allows references to objects to be tested if they refer to the same object, and hence forms a definition for object uniqueness.

An object's type is some class $C$. A class creates objects: it defines names and types for each attribute, and the set of *methods* that operate on the state of an object. These methods define the behavior of the object. An object that belongs to a class is said to be an *instance* of that class.

**Inheritance.**A set of classes $\mathbb{C} = \{C_1, \ldots, C_k\}$ is called a *class schema*. Classes have a concept of inheritance. If $C_i$ inherits from $C_j$, we write $C_i \prec C_j$, and the class $C_i$ inherits all of the attributes and methods of $C_j$. The inheritance relation is reflexive, transitive, and antisymmetric, and so defines a partial ordering on the class schema, called the *inheritance hierarchy*. This relation represents specialization as objects of class $C_i$ now can represent all the state and behavior of $C_j$, but can also add or modify attributes and methods. Thus, if $C_i \prec C_j$ and $o$ is an instance of $C_i$, then $o$ is also an instance of $C_j$. This property is called *polymorphism* and allows objects to act as an instance of any class more general than their own.

The maximal elements in the hierarchy are called the *base classes*. In many languages

[2]The set of OO types $\mathbf{T}'$ may, but does not necessarily, intersect with the set of types $\mathbf{T}$ used in the relational schema. They will almost certainly not be identical.



www.manaraa.com

multiple inheritance is possible, so a class can inherit directly from more than one class. For the purposes of ORM specifically, and OO languages in general, multiple inheritance introduces additional complexity that is best avoided, so we focus on single inheritance. In a single inheritance class schema, the inheritance hierarchy is not a general lattice, but a forest of *inheritance trees*, each rooted at a single base class. For single inheritance hierarchies we can uniquely define the super relation $\text{Super}(C_i) = C_j$ if $C_i \prec C_j$ and $C_i \prec C_k \prec C_j$ implies $C_k = C_i$ or $C_k = C_j$. In other words, the super relation determines the smallest class larger than a given class, called the immediate superclass. Conversely, $C_i$ is said to be a subclass of $C_j$.

A class can be *abstract* or *concrete*. There cannot be objects belonging to an abstract class, only to concrete classes. Abstract classes are only used to be inherited from by other classes.

### 9.2.3 Object-relational mapping

The object and relational models are general enough to apply to most modern OO languages and relational databases, hence they form a good basis for describing how objects can be mapped to relations. An ORM is a mapping from a class schema $\mathbb{C}$ to a relational schema $\mathbb{R}$ that provides a correspondence between objects in $\mathbb{C}$ and tuples (or sets of tuples) from relations in $\mathbb{R}$.

In this mapping attributes of an object with type $t_1 \in \mathbf{T}'$ are mapped to one or more tuple element with type(s) $\tau_i \in \mathbf{T}$. Since the types available in a programming language (subtly) differ from those available in databases, this mapping of types is a necessity, and may not be 1-to-1. However, for most uses the type differences have no practical effect, and we leave exploring the implications for value-based identity as future work.



www.manaraa.com

## 9.3 Object identity and uniqueness

The central issue addressed by the natural entity framework is consistently representing real-world entities that possess a concept of uniqueness described succinctly by the values of one or more well known (natural) attributes, i.e., a natural key.

### Identity in OO languages.

Like objects in the natural world, objects in a programming language have concepts of identity and uniqueness. Many OO programming languages (Python, Smalltalk, Java, Ruby, etc.) have a strong concept of object uniqueness in that each object has an associated immutable internal id(entifier), distinct from the references used to access it [70]. Such an id is called a *surrogate object id* since it has no relation to the value or meaning of the object. It merely serves to define the identity of the object and allows comparing the identity to those of other objects, as there is a bijection from object ids to objects [126].

### Identity in relational databases.

Identity in relational databases is a value-based property determined by a designated primary key. The primary keys should be unique, immutable, and non-null. The database maintains a uniqueness constraint on the primary key, preventing duplicate tuples, and uses an index to quickly select tuples by their primary key or detect violations of the uniqueness constraint. The primary key is also used to define foreign key relationships.

Because of all these important requirements placed on the primary key, it often makes sense to use a surrogate key as the primary key, even when there is a well-known natural key. There are many good reasons to prefer surrogate keys as primary keys, most of which arise from the fact that using surrogate keys allows the relational schema to decouple identity and value [27]. This allows more flexibility when the relational model needs to be updated or refactored [4]. Other benefits arise due to the fact that surrogate keys are



www.manaraa.com

simple (consist of a singleton attribute) and are typically small integral types. Natural keys in contrast are often compound and may include strings and other types that require more space as foreign keys. Since the primary key is always used to represent entity relationships through foreign key constraints, having a small, simple primary key reduces space usage and simplifies join operations. Simple integral keys are also often faster for use in selects against the primary key. For these reasons, ORMs often use surrogate primary keys by default [43].

However, natural keys are still useful and have some desirable characteristics. Declaring a natural key communicates to the database that the relational model has a logical uniqueness constraint on the natural key attributes and prevents a single conceptual entity from being represented by more than one tuple. Additionally, the database can then maintain a uniqueness constraint and index on the natural key. The presence of an index allows clients to quickly retrieve objects by their natural key-values, or determine that no such object exists. This can lead to distinct performance advantages for natural keys in some situations [78].

### 9.3.1 Identity in the natural entity framework

The natural entity framework, like other ORM tools, must reconcile the semantics of object identity in OO languages and tuple identity in relational databases. Our goal is to enforce the uniqueness of entity representation across both data models as determined by natural key attributes, but we simultaneously want to support polymorphic queries, efficient entity relationships, and flexibility for refactoring databases.

To achieve these objectives, the natural entity framework enforces the simultaneous use of surrogate primary keys and auxiliary natural keys. This dual-key representation achieves advantages of both surrogate and natural keys. In particular, our surrogate keys are unique within each inheritance hierarchy rooted at the `NaturalEntity` class. This uniformity of primary keys allows us to use a single top level relation to define a primary



www.manaraa.com

key for every object belonging to the class hierarchy. This makes polymorphic queries and associations much more efficient and uniform than they could be with natural keys. Indeed, without a uniform key for the entire inheritance tree, representing polymorphic associations would become problematic as there would be no single foreign key constraint that could be used to represent an association. Hence, surrogate primary keys are necessary for polymorphism and flexibility, but they do not fulfill the need for maintaining value-based uniqueness. This is achieved by the auxiliary natural keys. To maintain these auxiliary keys, the database must maintain a separate index, which takes up time and space; however, this index is exactly what ensures the logical value-based uniqueness of natural entities, and it is heavily used by constructors (Section 9.5) and other common queries against the natural key.

## 9.4 Management of persistent states and concurrency

Building on the concepts of object and relational identity, an ORM must have a way to track and manage the identity of in-memory objects. Unlike transient objects, which have a limited scope and lifetime, persistent objects must maintain their identity permanently and consistently across concurrent processes. To simplify the tracking of persistent objects and their modifications, modern ORM packages provide the concept of a *session manager*. The natural entity framework relies on a session manager to manage the persistent state of in-memory persistent objects and enforce the uniqueness constraints for natural entities.

Our principal contribution is to provide additional constructor methods which make explicit the assumptions about the state of a persistent object when it is created and prevent the user from violating the value-based uniqueness constraints.



www.manaraa.com

### 9.4.1 Transactions

The session manager has transactional semantics and manages a set of persistent objects by implementing the *unit of work* concept [43]. It tracks object creation, modification, and deletion. The session manager delegates large parts of this work to the database by using transactions. This ensures a consistent database state, even when objects are modified concurrently by other processes. It follows that the concurrency guarantees are largely provided by the transaction. The session manager supplies methods to control the global transactional state for an execution context. The `begin()` method starts a transaction and is implicitly called as needed if no transaction is currently in progress. The `flush()` method sends pending modifications to the database, but does not end the transaction. The `commit()` method commits a transaction, and this implies a flush operation if there are still pending changes. Finally, the `rollback()` method undoes all database changes made during the transaction.

### 9.4.2 Object states

From the perspective of an OO execution environment, reasoning about persistent objects is much more complicated than standard transient objects because the data representing the object can be stored in memory, in a relation(s) in the RDBMs, and/or in the memory of other concurrent processes. The session manager acts as the single point of persistence management for an OO execution environment. It determines how a persistent object relates to its external relational state in the database. Any object of a class that derives from a persistent base class, such as `NaturalEntity`, will be understood by the session manager to be in one of the following six states:

- *Transient* – The object is not managed as persistent by the session, while a corresponding tuple with the same natural key in the database may or may not exist; there is no operational connection with any persistent object.



www.manaraa.com

- *Pending* – The object does not yet have a permanent record but has been successfully added to the session and will be added to the database when the session state is flushed to the database. Until the object is successfully flushed it has not yet been assigned a primary key.

- *Persistent Clean* – The object has a primary key and a corresponding representation in the database. No persistently managed attributes have changed values, so no updates need to be sent to the database.

- *Persistent Dirty* – The same as a persistent clean object, except the value of one or more of the persistently maintained attributes has been changed, so that an SQL update operation is needed to save the state of the object. Copies of this object in other sessions do not know about the changes and may have made conflicting changes of their own.

- *Expired* – The object's state is no longer valid because it was created in a session that has been committed or rolled back, so its state needs to be reloaded from the database. This reloading is done transparently by the session manager when necessary.

- *Archived* – The object is not part of the store but is persistently stored in the database. Strictly speaking, this is not a state of an object, since no corresponding object exists in the session, but conceptually the tuple in database represents an object that is not currently loaded.

It is important to remember that the identity of a persistent object is provided by the natural key, and maintained through transactions and the constraint imposed by the database key. In case of conflicting concurrent transactions, e.g., simultaneous inserts or deletes, one of the concurrent processes will be prevented from committing its changes by an exception. In Figure 9.1 we show the effect of various operations on the persistent state of an object, but omit the expired state and other effects that occur at transaction boundaries. The



www.manaraa.com

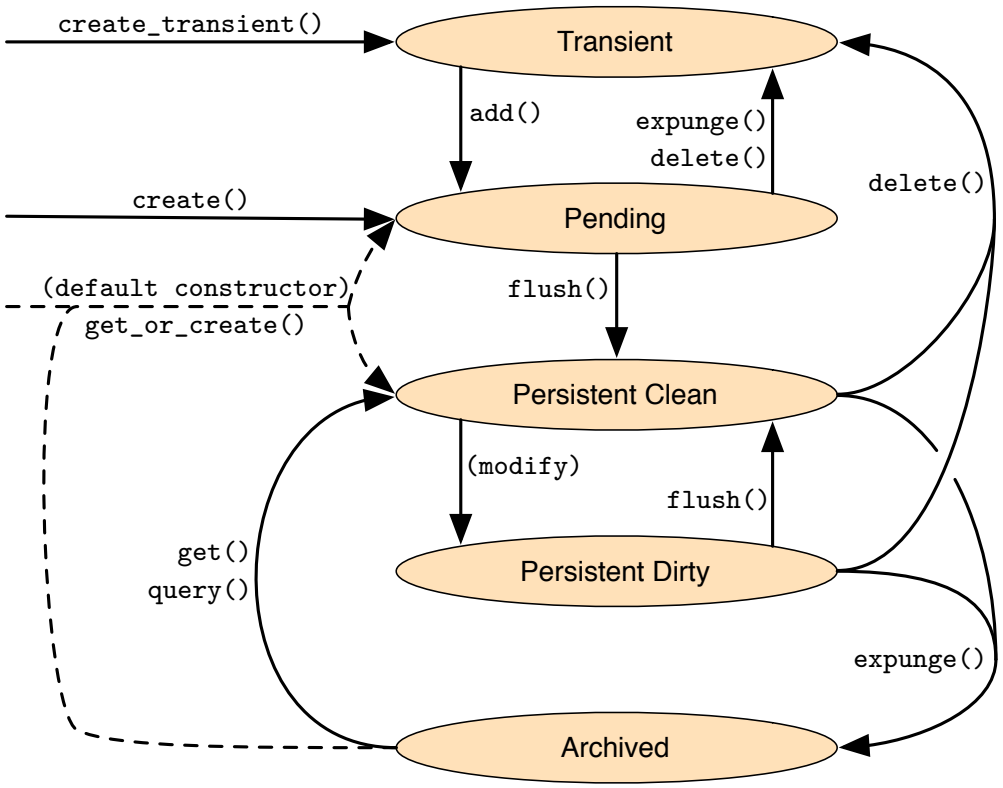


**Figure 9.1:** Persistent object states and effect of constructors and session commands within a single transaction context. The effects of transaction boundaries and the expired state are omitted for clarity.

effect of commits is to expire all pending and persistent objects and the session manager updates any identity maps of persistent objects accordingly (Section 9.5.1).

## 9.5 Object creation

Maintaining a value-based uniqueness constraint for persistent objects causes difficulties with object creation. Normally, the programming environment's concept of object identity is all that determines object uniqueness. When an object constructor is called, a new object with a unique object id is always created, and an initializer method is called. However,



www.manaraa.com

natural entity classes with value-based uniqueness constraints necessitate different semantics. First, the constructor must be given the values for each of the natural key attributes since they must not be null. Given the natural key value, the constructor is presented with several possibilities: (1) an object with those values already exists in memory so we are not allowed to create a new object with a new object id and the same natural key values; (2) an object with those values exists in an archived state, so it must be loaded from the database; or, (3) there is no persistent or in-memory object with the given natural key, so a new object should be created and added to the database.

Such a constructor requires a natural-keyed dictionary of in-memory persistent objects, i.e., an identity map (Section 9.5.1), and a mechanism to query for the existence of archived objects. Both of these can be provided efficiently by the session manger, but they nevertheless impose a significant cost, especially when the round trip time for remote database queries is involved. Unfortunately, such queries are necessary if we wish to maintain the consistency constraints; allowing the constructor to make new objects without regard to the natural key values would result in duplicate objects in memory. Furthermore, note that the cost of frequent queries can be reduced by allowing the caching of natural keys or prefetching of objects (particularly when the database transaction isolation prevents non-repeatable reads). When queries are necessary they can be handled efficiently because of the unique index maintained on the natural key attributes.

Together all of these considerations impose a significant change to the semantics of object creation, and can lead to conceptual problems for programmers. The natural entity framework addresses this conceptual ambiguity by providing additional constructor methods with different semantics. These constructors allow programmers to explicitly state their intentions or assumptions when creating an object.

- `get()` - A constructor that takes the natural key and returns the object uniquely identified by that key, either by returning a reference to an in-memory object representing that entity, or by loading an archived object from the database and returning



www.manaraa.com

it in the persistent clean state. If no such object exists, an exception is raised.

- `create()` - A constructor that takes the natural key and returns a newly created object in the pending state, but only if no persistent object with the same natural key exists in memory or in the archived state. An exception is raised if the object already exists.

- `get_or_create()` - A constructor with the combined semantics of the `get()` and `create()`. It takes the natural key and either returns an existing persistent object, or returns a newly created object in the pending state. This is the default constructor.

- `create_transient()` - A constructor with normal transient object semantics that always returns a new object in the transient state. It can take arbitrary arguments and ignores the uniqueness constraints.

The `get_or_create()` constructor does whatever it takes to get a reference to the unique object that has the provided natural key. It will find that object if it is in memory and return a reference, or it will look in the database for an archived version and return it, and if no such persistent object exists, it will construct a new object and make it persistent by moving it to the pending state. In practice we found that the `get_or_create()` gives the expected semantics in the vast majority of situations, and is thus the default constructor, leading to particularly succinct code (e.g., in Python `var=ClassName(...)`).

The `create()` and `get()` constructors are used in cases where the existence or non-existence of a particular `NaturalEntity`object represent a logical error, and the programmer would like an exception to be raised so that the errors are not silently ignored.

Finally the `create_transient()` constructor has several uses when the normal semantics of the natural entity construction are too rigid. Unlike the other constructors, `create_transient()` does not need to be given the natural key, and does not use any database connections or in-memory identity maps. This is useful for testing object behav-



www.manaraa.com

ior without using a database. Transient objects are also useful when the user does not wish to immediately pay the cost of the database query to check for archived objects. Furthermore, they support situations where not all of the natural key attributes are immediately available, but it makes sense to partially construct a `NaturalEntity` object, and then finish filling in the natural key attributes later. This is often the case in GUI or web-based applications where objects are built up sequentially by user actions. A transient object can be made persistent by using the `add()` method, which will check that all natural key attributes are specified and will raise an exception if the object already exists.

### 9.5.1 Identity map

When the (non-transient) constructors are called, they are provided with the complete natural key for the desired object. If an object with that natural key already exists in memory in the pending, expired, persistent clean, or persistent dirty states, it would be incorrect to construct and return a new object. Instead we must return a reference to the in-memory object. The ORM's session manager is able to track the persistent state of objects, but it also needs a way to look up objects by their natural key. This is a common requirement for ORMs, which Fowler calls the *identity map* pattern [43]. The purpose of an identity map is simply to map database keys to in-memory objects. When working with persistent objects, sometimes different parts of the code need access to the same data object without understanding whether that object is already in memory. The solution is to keep a global registry (or identity map) of in-memory objects keyed by their primary key. Normally, this identity map is stored in the session manager object, and it is used for internal ORM lookups of foreign key mappings. However, when primary keys are surrogates, it is awkward for a user to make use of this identity map, because the surrogates by definition are meaningless and often obscured from the user. It is much more common for a user to query using natural key attributes, and the constructors must be able to do this efficiently for in-memory objects. Hence, the natural entity system implements an



www.manaraa.com

auxiliary identity map, keyed on the natural key attributes. The identity map only stores in-memory persistent objects, i.e., transient objects are excluded. If an object is removed from the persistent store with the `delete()` method, it becomes transient. Thus, a constructor will not return a reference to a deleted object, even if that object is still in memory.

### 9.5.2 Initialization

Since the `NaturalEntity` constructors have multiple possible mechanisms for retrieving or creating objects, the concept of initialization also needs to be refined. For natural entities there are three distinct ways a new in-memory object could be created and require initialization: (1) it could be created as a transient object; (2) it could be retrieved from an archived state in the database; or, (3) it could be created as a new persistent object in the pending state. (In the case where the constructor already found the object in-memory through the identity map, no initialization is needed.) The `NaturalEntity` class provides three different initializers that will be called by the constructor in each of the three cases.

- `initialize()` – This method is called when a new persistent object is created. The object will be in the pending state and the object's (immutable) natural key attributes will have been set to the values provided to the constructor.

- `reinitialize()` – This method is called when an archived object is brought into memory by a constructor. The object will be in the persistent clean state and all persisted attributes (including the natural key attributes) will have been set by the ORM system.

- `initialize_transient()` – This method is called if and only if the object is constructed with the `create_transient()` method. The object will be in the transient state, and any supplied natural key attributes will have been set, but those omitted by the user (which is permitted for transient objects) will have no default value.



www.manaraa.com

### 9.5.3 Comparison with other ORMs

The multiple constructors of the natural entity framework represent a departure from the normal mechanism of persistent object creation presented by modern ORMs. In many modern ORM systems, all objects are initially created as transients, and only after a call to an `add()` method are they moved to a pending (or equivalent) state [73,96]. The difficulty with this mechanism is that it does not allow the ORM to directly manage value-based object uniqueness. In fact, the only way that a user will know if the in-memory objects conflict with persistent archived objects is to issue a database flush. When concurrent transactions attempt to make potentially conflicting changes, calls to `flush()` can hang indefinitely until other transactions have commit or rollback. For maximal concurrency it is best to flush infrequently or to also issue a commit (which cannot hang but may raise an exception). Commits, however are expensive as they require the ORM to expire the state of all in-memory objects, which must be subsequently reloaded from the database. Furthermore, if a persistent process avoids the expenses of flushes and commits, but does not guarantee consistency of object uniqueness, there is a potential for the process to do significant amounts of work (perhaps large computational simulations) only to find out when it finally issues a flush or commit that the constraints have been violated and the entire computation must be scrapped. Thus, while forcing the ORM to manage value-base object identity using natural keys imposes costs initially, particularly on object creation, these costs are often amortized by the need for less frequent flushes and commits and the reduced chances of database consistency constraint violations.

## 9.6 Mapping natural entity inheritance hierarchies

All natural entity classes must inherit from the `NaturalEntity` class, thus we must map all the classes in each inheritance subtree rooted at `NaturalEntity` into a relational schema. The natural entity system supports flexible mapping of hierarchies to relations,



www.manaraa.com

that allows for polymorphic queries and associations, as well as allowing different natural keys for separate subtrees of the inheritance hierarchy. The user only needs to supply minimal information about the desired inheritance mapping strategy and the ORM can automatically construct the appropriate tables and constraints. As an example we consider a distributed computer simulation system, with two inheritance hierarchies: an abstract `Experiment` class with two concrete subclasses; and an abstract `Measurement` class also with two concrete classes (Figure 9.2). An `Experiment` has a one-to-many relationship with measurements, so that each `Measurement` has a foreign key to the `Experiment` hierarchies primary key–a polymorphic association. We examine natural keys in the relation further in Section 9.6.2.

### 9.6.1 Inheritance mapping strategies

The relational data model has no built-in concept of inheritance, but support for inheritance and polymorphism can be enforced by appropriately structuring the relational schema and queries. There are three standard methods for mapping inheritance hierarchies to a relational schema [43]: (1) the *single table* strategy maps all classes in an inheritance hierarchy to a single table; (2) the *class table* strategy maps each class to its own table; and (3) the *concrete table* strategy maps only concrete classes to tables.

The single and class table strategies are particularly useful for polymorphic queries and associations as for every class in the hierarchy they store the class name (i.e., the type) and a surrogate object id in a single top level table. Concrete table inheritance lacks these properties and is not considered further.

Single and class table strategies are distinguished by the technique they use to represent the differing attributes for classes in the hierarchy. Single table inheritance has a single relation which includes all attributes of all classes in the hierarchy. It allows polymorphism by permitting attributes to be null for objects that do not include them. In contrast, class table inheritance only includes non-inherited attributes in each class table. It permits poly-



www.manaraa.com

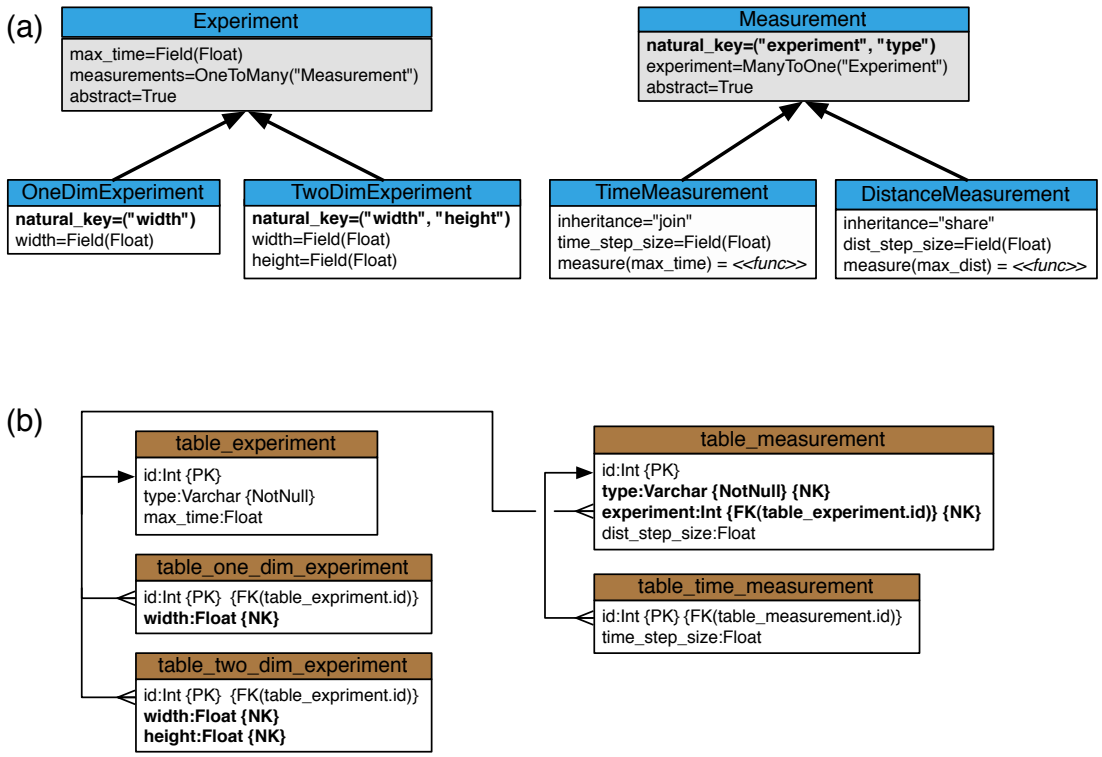


**Figure 9.2:** (a) A simple example of a class schema with two inheritance hierarchies, abstract classes, multiple natural key bases, polymorphic associations, and both shared and joined inheritance mappings. The text in each class entry is close to the actual amount of code needed to specify this hierarchy. We use syntax that is similar to our Python-based reference implementation of the natural entity framework. (b) The relational schema generated by the natural entity framework from the class schema in (a). The foreign key constraints are shown.

morphic queries by using joins on the primary surrogate key to retrieve attribute values from all the relations that store an object's state. These differences lead to quantifiable performance and space trade-offs [60]. Modern ORMs allow the user to specify a mixture of these strategies within a single inheritance hierarchy [24]. When mixing strategies, the single table approach is called *shared* or horizontal mapping, while the class-table approach is called *joined* or vertical mapping [88]. Shared table inheritance works best when the cost of additional join operations needed to load rows is a limiting factor, or when a portion of the class hierarchy shares almost all of the same persistent attributes. Joined table inheritance works best when database space is constrained, or in portions of



www.manaraa.com

the hierarchy where few persistent attributes are shared between classes.

In the natural entity framework each class in a hierarchy only needs to specify if it will use the shared or joined inheritance strategy and the ORM can automatically derive the relational schema.

### 9.6.2 Natural keys and inheritance

Every concrete class that derives from `NaturalEntity` must define or inherit a natural key, so that the constructor can enforce the value-based uniqueness constraint. Abstract classes need not define a natural key, and any class that has no natural key must be declared as abstract.

Because of the option to use joined inheritance, an individual object can have its attributes stored in several relations, but there is always a relation that stores the attributes declared specifically in a class. This is the *primary relation* of the class.

Consider a class $C$ that defines a natural key and that has no superclass which also defines a natural key (i.e., it has only abstract superclasses). The natural key results in a uniqueness constraint which is implemented by the database. A constraint can typically only be defined on attributes in a single table and not on joined tables. It follows that exactly one of the relations representing $C$ must enforce this constraint. None of $C$'s superclasses could have a natural key constraint, as enforcing a uniqueness constraint on $\text{Super}(C)$'s primary relation would prevent other subclasses of $\text{Super}(C)$ from defining different natural keys. Hence, the natural key constraint for $C$ must be enforced in $C$'s primary relation. This implies that all $C$'s natural key attributes must be defined in $C$ and cannot be inherited, or they would not be present in $C$'s primary relation. Finally, note that any subclass of $C$ will inherit $C$'s natural key attributes, and because these attributes have a uniqueness constraint defined on the relation that stores them, the subclass must also inherit the natural key from $C$.



www.manaraa.com

Therefore in any inheritance chain, i.e., starting at a concrete class and following the super relation to a base class, there is exactly one class that declares a natural key. Such a class is called a *natural key base*, as all classes that inherit from the natural key base share the same natural key constraint and store their natural key attributes in the primary relation of the natural key base.

Hence, when mapping a class hierarchy to a relational schema, the mapping will require: (1) a single table for the root class to store the primary key and object type; (2) a table for each natural key base (unless the class is also the root); and (3) a table for each class that uses joined inheritance (unless the class is a natural key root or the base class).

### 9.6.3 Type as a natural key attribute

A natural key base will pass on its natural key to all of its subclasses, and thus only one object of *any* derived class may have a given natural key value. Sometimes this is too restrictive a condition on the classes. Because the natural key distinguishes objects based on their value, but not their type, it restricts cases where objects have identical values but different behavior because their respective classes have different methods.

For example, consider the class structure of the distributed simulation system in in Figure 9.2. The `Measurement` class defines a simple natural key as a foreign key relationship to the `Experiment` it measures. An experiment should be able to include both a `TimeMeasurement` and `DistanceMeasurement` instance. However, because these objects have the same natural key this becomes impossible. The two measurement subclasses have the same attributes, but the meaning of the attributes differs due to different method implementations. Thus, it can make sense to have more than one measurement object with the same natural key, provided they belong to different classes. This can be accomplished by adding the implicit type attribute to the natural key base's primary relation and thus adding the type to the uniqueness constraint. This allows multiple `Measurements` to belong to a single `Experiment`, provided they are from different classes.



www.manaraa.com

In the natural entity framework the type can optionally be declared to be part of the natural key of a class to allow this distinction when it is required. The type attribute is automatically managed by the ORM, since it is always present as an attribute of any object in the OO programming language.

## 9.7 Conclusion

The natural entity framework is composed of general OO concepts and semantics that can be implemented in any OO language that supports strong concept of object identity. Object and class introspection, and the ability to instrument object construction and destruction are helpful features in making the implementation easy to use. Our reference implementation in Python is built on top of the SQLAlchemy ORM, and the Elixir extension.

Any persistence library that attempts to enforce value-based uniqueness constraints through natural keys and that allows polymorphic queries and associations will have to share several properties: (1) the objects will have to use a dual key representation with surrogate primary key and auxiliary natural key; (2) the ORM must maintain an identity map using the natural keys to avoid creating duplicate objects in memory; (3) the ORM must restrict inheritance hierarchies so that at most one class defines a natural key in each inheritance chain; and (4) the ORM must keep all the natural key attributes for a natural key base in a single table so that the RDBMs can enforce a uniqueness constraint on them.

The constructor methods of natural entities provide a consistent interface which distinguishes the different mechanisms by which a persistent class may be created and initialized. These constructors prevent the ORM from representing the same conceptual entity with different in-memory objects by ensuring that the value-based natural key constraints are maintained for all natural entity objects in the execution environment.

Enforcing value-based object identity changes the semantics of object models in the context of OO languages. However, these constraints only apply to objects from classes



www.manaraa.com

that inherit from `NaturalEntity`. Thus natural entities can coexist with objects of other less-strict persistent classes, as well as normal transient objects. Hence the natural entity framework makes it easier for a programmer to reason about object uniqueness for those entities which require it, but does not otherwise constrain the expressiveness of programs or programming languages. Our experience tells us that a natural key is present in most situations, and easily enforcing it has been an invaluable tool in writing correct scientific software.



www.manaraa.com

# Part III

# Perspective and Conclusion



www.manaraa.com

# Chapter 10
# Executable Biology

Continuous-time Markov processes describe systems in terms of discrete states and state transitions—concepts that are ubiquitous in models of computation. The structure of the kinetic Monte Carlo approach is essentially a direct mapping between samples of a Markov process and execution traces of a computer simulation. The state of the simulation process encodes the state of the Markov process, and the execution path of the simulation emulates the transitions of the Markov process from state to state. This relationship becomes insightful for models such as the MVRW model where the states and transitions represent an approximation of the actual physical states and dynamics of the system. A computer simulation of a CTMP model such as the MVRW model represents a hypothetical *execution* of the physical or chemical system, rather than just an abstract computational solution to a mathematical function.

Fisher and Henzinger have recently introduced the idea of *executable biology* [41] as an approach to biological modeling focusing on *computational models* that "present a recipe—an algorithm—for an abstract execution engine to mimic a design or natural phenomenon." For Fisher and Henzinger, a computational model is described principally by operational semantics, and the execution of the model parallels the hypothetical physical and chemical evolution of the system. This is in contrast to *mathematical models* that describe a system as a set of equations, where the procedure for numerically estimating the solution of the equations has no semantic connection to the physical and chemical processes that give rise to the system dynamics. As an example of such a mathematical model, consider the standard deterministic model of the mass action kinetics of a chemical system (Section 3.1). This model uses differential equations derived from the law of mass action to describe the species concentrations over time. The model is accurate for most chemical reactions in large, well-mixed volumes of dilute solutions. Finding a solution to the equations allows one to accurately predict the dynamics of the system. However, any



www.manaraa.com

numerical estimation of the differential equations follows an execution pattern that has no direct relation to the actual molecular events that drive the system. The goal of computation in such a model is to obtain the best approximation to the solution of the mathematical system in the most efficient manner. This approach is functional and practical, but it does not shed any light on how the dynamics of the system result from the elementary chemical events.

Fisher and Henzinger's definitions of mathematical models and computational models are compelling, but suffer from a narrow focus on process calculi and interacting state machine models. For them, an essential feature of a computational model is that it is described algorithmically and is intrinsically executable with no ambiguity in the intended implementation. However, the real advantage of the computational model is not in the language of its description, but in the form of its assumptions. In a computational model, we view a system in terms of its physical and chemical constituents and assume that the state of the system is the state of its parts. Furthermore, we assume that temporal evolution of the system is governed by interactions of the constituents through a sequence of elementary events leading to discrete changes in the system state. Any model that characterizes a system in such a manner represents an *executable understanding* of the system, regardless of the language in which it is described. The real advantage of computational models is that their execution follows a sequence of events that correspond to an approximation of the real physical and chemical events driving the system at relevant time scales. Developing, running, and observing simulations gives us direct insight into the way natural laws give rise to complex effects through stochastic sequences of elementary events. Visualizing the execution of such a simulation is the virtual equivalent of watching a real physical system evolve, and allows the user to develop an operational understanding of the system in ways not possible with other models.

With a more fundamental description of a computational model in terms of the assumptions made about the system, we find that there is no longer a strict dichotomy between



www.manaraa.com

mathematical and computational models. In fact, most models based on CTMPs are both mathematical and computational. Clearly, a CTMP is a formal mathematical description of a system and hence a mathematical model. However, through the KMC algorithm, any such model also gives rise to an executable description of the system. Many famous examples of this class of models exist, for example, the Ising model of magnetism [18], Gillespie's stochastic model of chemical kinetics [48], and the random walk model of diffusion.

A model that is both mathematical and computational has the advantages of both classes. A purely mathematical model with no direct description of the elementary objects and events is conceptually opaque; we can use it to predict the system dynamics, but it provides no insight into the causes of the dynamics. On the other hand, a purely computational model described algorithmically lacks context. Such a model is difficult to compare with other models that do not share a common language of description in mathematics. Phrasing a model in terms of mathematics often helps to extract commonalities among models and suggest relationships and connections that may not be obvious from an executable algorithm. Also, a mathematical model will often admit some analytical results. Even if the full dynamics of the model is not analytically tractable, we can establish results regarding asymptotic behavior or identify conservation relations. Additionally, the formality with which mathematical models are described allows them to be derived from physical laws and assumptions in a rigorous and logically justifiable manner.

The ability to describe a system both mathematically and computationally represents a more fundamental understanding of the system than a model that fits in only one class or the other. Such a model requires understanding the fundamental mathematical relationships of the system, as well as how those relationships govern the execution of the system in terms of its constituent parts. It is easy to propose ad hoc computational models with no physical justification or to propose mathematical relationships observed empirically through experiment, but understanding how the relationships and dynamics result from



www.manaraa.com

the elementary interactions is much more enlightening. Thus we take this approach in our MVRW model of molecular transport.



www.manaraa.com

# Chapter 11
# Conclusion

In many ways the molecular spiders and the MVRW model exemplify the challenges of understanding the detailed kinetics of molecular-scale systems. While the overall structure of the spiders is simple and they interact with their environment under a set of simple rules, these rules are necessarily stochastic because of the natural thermal fluctuations on these scales. The stochasticity and the multitude of simultaneous physical and chemical processes operating make it very difficult to qualitatively or quantitatively understand how the spiders will move through analytical methods. In addition, the inadequacy of spatial-temporal resolution of current microscopy techniques also makes experimental investigations difficult. In such situations computational models and computer simulations are essential. Moreover, we shave shown that unanticipated effects and complex mechanisms emerge from the simple physical and chemical rules that govern the dynamics of the spiders.

Furthermore, computational thinking is required to arrive at a model that is both physically plausible and computationally feasible. The assumptions made in the MVRW are simple and are chosen to lead to a model that is practical to simulate with modern desktop computing resources. We take advantage of the Markovian nature of physical and chemical systems on the timescales of interest, which leads to an *executable understanding* of the MVRW system via the KMC algorithm.



www.manaraa.com

# Glossary of Symbols

| | | |
|---|---|---|
| $\boldsymbol{A}$ | The attached leg sites completely define the state of the walker. In general, $\boldsymbol{A} = [\boldsymbol{a}_i \in S \cup \{\odot\}]_{i=1}^{k}$. In the special case of point-bodied walkers all $k$ legs are identical and $\boldsymbol{A}$ can be replaced by the *unordered set* of attached legs written as $A$. | 34 |
| $\boldsymbol{B}$ | The random variable describing the body's equilibrium distribution over positions $(\boldsymbol{v}, \theta)$. This random variable depends on the current value of $\boldsymbol{A}$. | 37 |
| $\widehat{\mathbb{C}}$ | The set of *feasible canonical configurations*. | 65 |
| $C_b = \mathbb{R}^2$ | The space of 2D coordinates in the walker body's reference frame. | 34 |
| $C_e = \mathbb{R}^2$ | The space of 2D coordinates in the environment's reference frame. | 34 |
| $\chi : \Sigma \to \Sigma$ | The species transformation resulting from catalysis for each species. | 37 |
| $\odot$ | A detached state for a leg. | 34 |
| $\Delta E(\boldsymbol{b}) = \Delta E_f(\boldsymbol{b})$ | The change in potential energy of the walker at position $\boldsymbol{b}$ as it moves from original position $\boldsymbol{b}_0$ under the load of conservative force $\boldsymbol{f}$. | 45 |
| $\mathbb{E}$ | The set of environment states. | 32 |
| $\mathrm{Exp}(\lambda)$ | The exponential distribution with parameter $\lambda$. | 54 |
| $\mathcal{F}$ | The set of feasible body positions $(\boldsymbol{v}, \theta)$. | 38 |
| $f_{\boldsymbol{B}}^{i}(\boldsymbol{s})$ | The *feasibility probability* of leg $i$ attaching to site $\boldsymbol{s}$, given body distribution $\boldsymbol{B}$. | 43 |
| $\mathrm{Fpt}(d; t)$ | The first passage time—the distribution of the time to first reach distance $d$ from the origin. | 83 |
| $\Delta G$ | The change in Gibbs free energy | 17 |
| $\boldsymbol{H}$ | The hip locations $\boldsymbol{H} = [\boldsymbol{h}_i]_{i=1}^{k}$ in the body's coordinates $C_b$. | 34 |



www.manaraa.com

| | | |
|---|---|---|
| $I_{\boldsymbol{b}}^{i}(\boldsymbol{s})$ | The *feasibility indicator function* for leg $i$ binding to site $\boldsymbol{s}$ from fixed body position $\boldsymbol{b}$. This determines the attachment kinetics of an unattached leg from a fixed body position. | 43 |
| $k$ | The number of legs. | 34 |
| $k_B$ | Boltzmann's constant is $1.38 \times 10^{-23}\,\mathrm{J\,K^{-1}}$. At $T = 300\,\mathrm{K}$, we have $k_B T = 4.14\,\mathrm{pN\,nm}$, which is the average amount of thermal energy available to a walker. | 18 |
| $k_{\mathrm{cat}} : \Sigma \to \mathbb{R}^+$ | The rate of leg catalysis for each species. When working with $\Sigma = \{\mathrm{S}, \mathrm{P}\}$, only substrates can be catalyzed and we let $k_{\mathrm{cat}} = k_{\mathrm{cat}}(\mathrm{S})$. | 37 |
| $k^- : \Sigma \to \mathbb{R}^+$ | The rate of the leg dissociation reaction for each species. When working with $\Sigma = \{\mathrm{S}, \mathrm{P}\}$, we write $k_{\mathrm{P}}^- = k^-(\mathrm{P})$, and $k_{\mathrm{S}}^- = k^-(\mathrm{S})$. | 36 |
| $k^+ : \Sigma \to \mathbb{R}^+$ | The rate of the leg binding reaction for each species. When working with $\Sigma = \{\mathrm{S}, \mathrm{P}\}$, we write $k_{\mathrm{P}}^+ = k^+(\mathrm{P})$, and $k_{\mathrm{S}}^+ = k^+(\mathrm{S})$. | 36 |
| $\ell$ | The length of the legs. | 34 |
| $\Omega = \mathbb{E} \times \mathbb{W}$ | The set of all states for the Markov process defined by the MVRW model. | 36 |
| $p_B(\boldsymbol{v}, \theta)$ | The probability of the body position being $(\boldsymbol{v}, \theta)$ at equilibrium. | 38 |
| $\widehat{\Phi}_C$ | The canonical mapping for attached leg configuration $C$ takes the leftmost of the lowermost sites, $\boldsymbol{c}_{LL}$, and translates it to the origin. This gives the unique canonical representative for an attached leg configuration $C$. | 65 |
| $\pi : S \to \Sigma$ | The site species function: a mapping from a site to the species displayed at that site. | 32 |
| $\mathbf{P}[\,E\,]$ | The probability of event $E$. | 38 |
| $r_{\boldsymbol{B}}^{i}(\boldsymbol{s})$ | The *attachment transition rate* for leg $i$ attaching to site $\boldsymbol{s}$, given body distribution $\boldsymbol{B}$. | 44 |



www.manaraa.com

| | | |
|---|---|---|
| $S$ | The set of chemical sites in coordinates $C_e$. | 32 |
| $S_{\widehat{\mathbb{C}}}$ | The set of all possible canonical lattice coordinates, defined as the set of all coordinates from any canonical configuration in $\widehat{\mathbb{C}}$. | 65 |
| $S^i_{\mathcal{F}(\boldsymbol{A})}$ | The set of feasible sites $\boldsymbol{s} \in S$ that are within distance $\ell$ of $\boldsymbol{h}_i$ from some feasible body position in $\mathcal{F}(\boldsymbol{A})$. | 45 |
| $\Sigma$ | The set of chemical species. Typically we have $\Sigma = \{\mathrm{S}, \mathrm{P}\}$. | 32 |
| $T$ | Absolute temperature in Kelvin. We fix $T = 300\,\mathrm{K}$ for the isothermal walker systems modeled by the MVRW model. | 17 |
| $T(\boldsymbol{v}, \theta)$ | A 2D rigid body transform from the body's coordinates $C_b$ to the environment's coordinates $C_e$. This defines the location of the body. For point-bodied spiders $T(\boldsymbol{v}, \theta) = T(\boldsymbol{v})$, as they are rotationally symmetric. | 34 |
| $U(\boldsymbol{v}, \theta)$ | The energy associated with body position $(\boldsymbol{v}, \theta)$. This determines the equilibrium distribution via the Boltzmann distribution. | 38 |
| $\mathrm{Uniform}(a, b)$ | The uniform distribution over real interval $[a, b]$. | 54 |
| $\widehat{\mathbb{U}}$ | The set of *unique canonical configurations*, $\widehat{\mathbb{U}} \subseteq \widehat{\mathbb{C}}$ is the set of minimal elements of the unique canonical configuration ordering $\preceq$ over $\widehat{\mathbb{C}}$. | 67 |
| $\mathfrak{U} : \widehat{\mathbb{C}} \to \widehat{\mathbb{U}}$ | The *unique canonical mapping* takes a canonical configuration $\widehat{C}$ to its equivalent unique canonical configuration $\widehat{U} = \mathfrak{U}(\widehat{C})$, which is the minimal element of the chain of $\widehat{C}$ in the canonical equivalent mapping $\preceq$. | 67 |
| $\mathbb{W}$ | The set of walker states. | 35 |
| $w^\star(f)$ | The mean peak work a walker does moving against a force $f$. This is defined as $w^\star(f) = \max_{t \in [0, t_{\max}]} \langle \Delta E(t; f) \rangle - \Delta E_\infty(f)$, where $\Delta E_\infty(f)$ is the equilibrium energy of the spider system under force $f$. | 92 |



www.manaraa.com

| | | |
|---|---|---|
| $Z$ | The partition function in the Boltzmann distribution. | 39 |



www.manaraa.com

# References


[1] Leonard M. Adleman. Molecular computation of solutions to combinatorial problems. *Science*, 266(5187):1021–1024, 1994.

[2] Ashutosh Agarwal and Henry Hess. Biomolecular motors at the intersection of nanotechnology and polymer science. *Progress in Polymer Science*, 35(1–2):252–277, 2010.

[3] Francesc Alted and Mercedes Fernández-Alonso. PyTables: Processing and analyzing extremely large amounts of data in Python. In *PyCon 2003*, 2003.

[4] Scott W. Ambler. *Agile Database Techniques*. Wiley, Indianapolis, IN, 2003.

[5] Pier Lucio Anelli. A molecular shuttle. *Journal of the American Chemical Society*, 113:5131–5133, 1991.

[6] Tibor Antal and Paul L. Krapivsky. Molecular spiders with memory. *Physical Review E*, 76(2):021121, 2007.

[7] R. Dean Astumian. Design principles for Brownian molecular machines: how to swim in molasses and walk in a hurricane. *Physical Chemistry and Chemical Physics*, 9:5067–5083, 2007.

[8] R. Dean Astumian. Thermodynamics and kinetics of molecular motors. *Biophysical Journal*, 98(11):2401–2409, 2010.

[9] R. Dean Astumian and Imre Derényi. A chemically reversible brownian motor: application to kinesin and Ncd. *Biophysical Journal*, 77(2):993–1002, 1999.

[10] Heiko Bauke. *Tina's random number generator library*, August 2011.

[11] Heiko Bauke and Stephan Mertens. Random number generators for large-scale distributed Monte Carlo simulations. *Physical Review E*, 75(6):066701, 2007.

[12] Yaakov Benenson, Benyamin Gil, Uri Ben-Dor, Rivka Adar, and Ehud Shapiro. An autonomous molecular computer for logical control of gene expression. *Nature*, 429(6990):423–429, May 2004.

[13] Jorge Berger. From randomness to order. *Entropy*, 6(1):68–75, 2004.

[14] Veronika Bierbaum and Reinhard Lipowsky. Chemomechanical coupling and motor cycles of myosin V. *Biophysical Journal*, 100(7):1747–1755, 2011.

[15] Kurt Binder, Jürgen Horbach, Walter Kob, Wolfgang Paul, and Fathollah Varnik. Molecular dynamics simulations. *Journal of Physics: Condensed Matter*, 16(5):S429, 2004.




المنارة للاستشارات
www.manaraa.com

[16] Gerd Binnig, Calvin F. Quate, and Christoph Gerber. Atomic force microscope. *Physical Review Letters*, 56:930–933, 1986.

[17] Benjamin Block, Peter Virnau, and Tobias Preis. Multi-GPU accelerated multi-spin Monte Carlo simulations of the 2D Ising model. *Computer Physics Communications*, 181(9):1549–1556, 2010.

[18] Alfred B. Bortz, Malvin H. Kalos, and Joel L. Lebowitz. A new algorithm for Monte Carlo simulation of Ising spin systems. *Journal of Computational Physics*, 17(1):10 – 18, 1975.

[19] Ronald R. Breaker and Gerald F. Joyce. A DNA enzyme that cleaves RNA. *Chemistry & Biology*, 1(4):223–224, 1994.

[20] Ronald R. Breaker and Gerald F. Joyce. A DNA enzyme with $Mg^{2+}$-dependent RNA phosphoesterase activity. *Chemistry & Biology*, 2(10):655–660, 1995.

[21] Christian Brunner, Christian Wahnes, and Viola Vogel. Cargo pick-up from engineered loading stations by kinesin driven molecular shuttles. *Lab on a Chip*, 7(10):1263–1271, 2007.

[22] Petra Burgstaller and Michael Famulok. Synthetic ribozymes and the first deoxyribozyme. *Angewandte Chemie Int. Ed.*, 34(11):1189–1192, 1995.

[23] Carlos Bustamante, Yann R. Chemla, Nancy R. Forde, and David Izhaky. Mechanical processes in biochemistry. *Annual Review of Biochemistry*, 73:703–748, 2004.

[24] Luca Cabibbo. Managing inheritance hierarchies in object/relational mapping tools. In *CAiSE Conference on Advanced Information Systems Engineering*, pages 135–150, 2005.

[25] Siddhartha Chib and Edward Greenberg. Understanding the Metropolis-Hastings algorithm. *The American Statistician*, 49(4):327–335, 1995.

[26] Edgar F. Codd. A relational model of data for large shared data banks. *Communications of the ACM*, 13(6):377–387, 1970.

[27] Edger F. Codd. Extending the database relational model to capture more meaning. *ACM Transactions on Database Systems*, 4(4):397–434, 1979.

[28] Shirley Cohen, Patrick Hurley, Karl W. Schulz, William L. Barth, and Brad Benton. Scientific formats for object-relational database systems: A study of suitability and performance. *SIGMOD Record*, 35(2):10–15, 2006.



www.manaraa.com

[29] Chris M. Coppin, Jeffrey T. Finer, James A. Spudich, and Ronald D. Vale. Detection of sub-8-nm movements of kinesin by high-resolution optical-trap microscopy. *Proceedings of the National Academy of Sciences*, 93(5):1913–1917, 1996.

[30] Chris M. Coppin, Daniel W Pierce, Long Hsu, and Ronald D. Vale. The load dependence on kinesin's mechanical cycle. *Proceedings of the National Academy of Sciences*, 94(16):8539–8544, 1997.

[31] Enrique M. De La Cruz, Amber L. Wells, Steven S. Rosenfeld, E. Michael Ostap, and H. Lee Sweeney. The kinetic mechanism of myosin V. *Proceedings of the National Academy of Sciences*, 96(24):13726–13731, 1999.

[32] John R. Dennis, Jonathan Howard, and Viola Vogel. Molecular shuttles: directed motion of microtubules along nanoscale kinesin tracks. *Nanotechnology*, 10(3):232, 1999.

[33] Ramez Elmasri and Shamkant B. Navathe. *Fundamentals of Database Systems*. Addison Wesley, Boston, MA, 2004.

[34] Akihiro Enomoto, Michael Moore, Tadashi Nakano, Ryota Egashira, Tatsuya Suda, Atsushi Kayasuga, Hiroaki Kojima, Hitoshi Sakakibara, and Kazuhiro Oiwa. A molecular communication system using a network of cytoskelatal filaments. *NSTI-Nanotech*, 1:725–728, 2006.

[35] Baptiste Essevaz-Roulet, Ulrich Bockelmann, and Francois Heslot. Mechanical separation of the complementary strands of DNA. *Proceedings of the National Academy of Sciences*, 94(22):11935–11940, 1997.

[36] Richard Levery et. al. A systematic molecular dynamics study of nearest-neighbor effects on base pair and base pair step conformations and fluctuations in B-DNA. *Nucleic Acids Research*, 38(1):299–313, 2010.

[37] Evan Evans and Ken Ritchie. Strength of a weak bond connecting flexible polymer chains. *Biophysical Journal*, 76(5):2439–2447, 1999.

[38] Nick P. Ferenz, Alyssa Gable, and Pat Wadsworth. Mitotic functions of kinesin-5. *Seminars in Cell & Developmental Biology*, 21(3):255–259, 2010.

[39] Alan M. Ferrenberg, D. P. Landau, and Y. Joanna Wong. Monte Carlo simulations: Hidden errors from "good" random number generators. *Physical Review Letters*, 69(23):3382–3384, 1993.

[40] A Fischer, S Seeger, K H Hoffmann, C Essex, and M Davison. Modeling anomalous superdiffusion. *Journal of Physics A: Mathematical and Theoretical*, 40(38):11441, 2007.



www.manaraa.com

[41] Jasmin Fisher and Thomas A. Henzinger. Executable cell biology. *Nature Biotechnology*, 25(11):1239–1249, November 2007.

[42] Mike Folk, Gerd Heber, Quincey Koziol, Elena Pourmal, and Dana Robinson. An overview of the HDF5 technology suite and its applications. In *Proceedings of the EDBT/ICDT 2011 Workshop on Array Databases*, pages 36–47, 2011.

[43] Martin Fowler. *Patterns of Enterprise Application Architecture*. Addison-Wesley, Boston, MA, 2003.

[44] Ronald F. Fox. Rectified Brownian movement in molecular and cell biology. *Physical Review E*, 57(2):2177–2203, 1998.

[45] Charles J. Geyer. Practical Markov chain Monte Carlo. *Statistical Science*, 7(4):473–483, 1992.

[46] Charles J. Geyer. Introduction to Markov chain Monte Carlo. In *Handbook of Markov Chain Monte Carlo*, pages 3–47. CRC, 2011.

[47] Michael A. Gibson and Jehoshua Bruck. Efficient exact stochastic simulation of chemical systems with many species and many channels. *Journal of Physical Chemistry A*, 104(9):1876–1889, March 2000.

[48] Daniel T. Gillespie. A general method for numerically simulating the stochastic time evolution of coupled chemical reactions. *Journal of Computational Physics*, 22(4):403–434, 1976.

[49] Daniel T. Gillespie. Exact stochastic simulation of coupled chemical reactions. *The Journal of Physical Chemistry*, 81(25):2340–2361, 1977.

[50] Lawrence S. B. Goldstein and Alastair Valentine Philp. The road less traveled: emerging principles of kinesin motor utilization. *Annual Review of Cell and Developmental Biology*, 15:141–183, 1999.

[51] Solomon W. Golomb. *Polyominoes*. Princeton University Press, Princeton, NJ, 2nd edition, 1994.

[52] Jim Gray. Scientific data management in the coming decade. *SIGMOD Record*, 34(4):34–41, 2005.

[53] Eric Green, Mark J. Olah, Tatiana Abramova, Lance R. Williams, Darko Stefanovic, Tilla Worgall, and Milan N. Stojanovic. A rational approach to minimal high-resolution cross-reactive arrays. *Journal of the American Chemical Society*, 128(47):15278–15282, 2006.



www.manaraa.com

[54] W. Keith Hastings. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57(1):97–109, April 1970.

[55] Shlomo Havlin and Daniel Ben-Avraham. Diffusion in disordered media. *Advances in physics*, 51(1):187–292, 1987.

[56] Paul Helman. *The Science of Database Management*. Richard D. Irwin Inc., Burr Ride, IL, 1994.

[57] Niels E. Henriksen and Flemming Y. Hansen. *Theories of Molecular Reaction Dynamics*. Oxford University Press, New York, NY, 2008.

[58] Henry Hess, John Clemmens, Christian Brunner, Robert Doot, Sheila Luna, Karl-Heinz Ernst, and Viola Vogel. Molecular self-assembly of nanowires and nanospools using active transport. *Nano Letters*, 5(4):629–633, 2005.

[59] Kent L. Hill, Natalie L. Catlett, and Lois S. Weisman. Actin and myosin function in directed vacuole movement during cell division in *saccharomyces cerevisiae*. *Journal of Cell Biology*, 135(6):1535–1549, 1996.

[60] Stefan Holder, Jim Buchan, and Stephen G. MacDonell. Towards a metrics suite for object-relational mappings. In *Model-Based Software and Data Integration*, pages 43–54. Springer, 2008.

[61] Jiří Homola, Sinclair S. Yee, and Günter Gauglitz. Surface plasmon resonance sensors: review. *Sensors and Actuators B: Chemical*, 54(1–2):3–15, 1999.

[62] Joe Howard. Molecular motors: structural adaptations to cellular functions. *Nature*, 389(6651):561–567, 1997.

[63] Jonathan Howard. Mechanical signaling in networks of motor and cytoskeletal proteins. *Annual Reviews in Biophysics*, 38:217–234, 2009.

[64] Bo Huang, Mark Bates, and Xiaowei Zhuang. Super resolution fluorescence microscopy. *Annual Review of Biochemistry*, 78:993–1016, 2009.

[65] A. F. Huxley. Muscle structure and theories of contraction. *Progress in Biophysics and Biophysical Chemistry*, 7:255–318, 1957.

[66] Christopher Ireland, David Bowers, Michael Newton, and Kevin Waugh. A classification of object-relational impedance mismatch. In *Proceedings of the 2009 First International Conference on Advances in Databases, Knowledge, and Data Applications*, pages 36–43. IEEE Computer Society, 2009.



www.manaraa.com

[67] Jae-Hyung Jeon and Ralf Metzler. Inequivalence of time and ensemble averages in ergodic systems: exponential versus power-law relaxation in confinement. *Physical Review E*, 85(2):250602, 2012.

[68] Malvin H. Kalos and Paula A. Whitlock. *Monte Carlo Methods*. John Wiley & Sons, New York, NY, 1986.

[69] David Keller and Carlos Bustamante. The mechanochemistry of molecular motors. *Biophysical Journal*, 78(2):541–556, 2000.

[70] Setrag Khoshafian and George P. Copeland. Object identity. In *OOPSLA Object-Oriented Programming, Systems, Languages, and Applications*, pages 406–416, 1986.

[71] Donald E. Knuth. *Art of Computer Programming, Volume 2: Seminumerical Algorithms*. Addison-Wesley, 3rd edition, 1993.

[72] Anatoly B. Kolomeisky and Michael E. Fisher. Molecular motors: a theorist's perspective. *Annual Reviews of Physical Chemistry*, 58:675–695, 2007.

[73] Thomas Kowark, Robert Hirschfeld, and Michael Haupt. Object-relational mapping with squeaksave. In *Proceedings of the International Workshop on Smalltalk Technologies*, IWST '09, pages 87–100, New York, NY, 2009. ACM.

[74] Akihiro Kusumi, Yasushi Sako, and Mutsuya Yamamoto. Confined lateral diffusion of membrane receptors as studied by single particle tracking (nanovid microscopy). effects of calcium-induced differentiation in cultured epithelial cells. *Biophysical Journal*, 65(5):2021–2040, 1993.

[75] Pierre L'Ecuyer. Tables of linear congruential generators of different sizes and good lattice structure. *Mathematics of Computation*, 68(225):249–260, 1999.

[76] Chih-Kung Lee, Yu-Ming Wang, Long-Sun Huang, and Shiming Lin. Atomic force microscopy: determination of unbinding force off rate and energy barrier for protein-ligand interaction. *Micron*, 38(5):446–461, 2007.

[77] Steffen Liepelt and Reinhard Lipowsky. Kinesin's network of chemomechanical motor cycles. *Physical Review Letters*, 98(25):258102, 2007.

[78] Sebastian Link, Ivan Lukovic, and Pavle Mogin. Performance evaluation of natural and surrogate key database architectures. Technical report, Victoria University of Wellington, Wellington, NZ, 2010.



www.manaraa.com

[79] Reinhard Lipowsky and Steffen Liepelt. Chemomechanical coupling of molecular motors: Thermodynamics, network representations, and balance conditions. *Journal of Statistical Physics*, 130(1):39–67, 2008.

[80] Ariel Lubelski, Igor M. Sokolov, and Joseph Klafter. Nonergodicity mimics inhomogeneity in single particle tracking. *Physical Review Letters*, 100(5):250602, 2008.

[81] Kyle Lund. personal communication, 2008.

[82] Kyle Lund, Anthony J. Manzo, Nadine Dabby, Nicole Michelotti, Alexander Johnson-Buck, Jeanette Nangreave, Steven Taylor, Renjun Pei, Milan N. Stojanovic, Nils G. Walter, Erik Winfree, and Hao Yan. Molecular robots guided by prescriptive landscapes. *Nature*, 465:206–210, May 2010.

[83] Joanne Macdonald, Yang Li, Marko Sutovic, Harvey Lederman, Kiran Pendri, Wanhong Lu, Benjamin L. Andrews, Darko Stefanovic, and Milan N. Stojanovic. Medium scale integration of molecular logic gates in an automaton. *Nano Letters*, 6(11):2598–2603, 2006.

[84] Makoto Matsumoto and Takuji Nishimura. Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Transactions on Modeling and Computer Simulation*, 8(1):3–30, 1998.

[85] Stephan Mertens. Lattice animals: A fast enumeration algorithm and new perimeter polynomials. *Journal of Statistical Physics*, 58(5–6):1095–1108, 1990.

[86] Stephan Mertens. *Random number generators: A survival guide for large scale simulations*. Otto-von-Guericke University Magdeburg, 2009.

[87] Nicholas Metropolis, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller, and Edward Teller. Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21(6):1087–1092, 1953.

[88] Peter Mork, Philip Bernstein, and Sergey Melnik. Teaching a schema translator to produce O/R views. In *Conceptual Modeling - ER 2007*, volume 4801 of *LNCS*, pages 102–119. Springer, 2007.

[89] Dan V. Nicolau, Dan V. Nicolau, Jr., Gerardin Solana, Kristi L. Hanson, Luisa Filipponi, Lisen Wang, and Abraham P. Lee. Molecular motors-based micro- and nano-biocomputation devices. *Microelectronic Engineering*, 83(4-9):1582–1588, 2006.



www.manaraa.com

[90] Dan V. Nicolau, Jr., Kevin Burrage, and Dan V. Nicolau. Computing with motile bio-agents. In *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, volume 6416, Dec. 2007.

[91] Masayoshi Nishiyama, Hideo Higuchi, and Toshio Yanagida. Chemomechanical coupling of the forward and backward steps of single kinesin molecules. *Nature Cell Biology*, 4:790–797, 2002.

[92] Mark J. Olah, David Mohr, and Darko Stefanovic. Representing uniqueness constraints in object-relational mapping: The natural entity framework. In *Objects, Models, Components, Patterns*, volume 7304 of *Lecture Notes in Computer Science*, pages 236–251. Springer Berlin / Heidelberg, 2012.

[93] Mark J. Olah and Darko Stefanovic. Multivalent random walkers: A model for deoxyribozyme walkers. In *DNA Computing and Molecular Programming*, volume 6937 of *Lecture Notes in Computer Science*, pages 160–174. Springer Berlin / Heidelberg, 2011.

[94] Mark J. Olah and Darko Stefanovic. Superdiffusive transport by multivalent molecular walkers moving under load. *in submission* 2012.

[95] Mark J. Olah and Darko Stefanovic. Kinetic Monte Carlo simulation for multivalent random walkers. *in preparation*.

[96] Elizabeth J. O'Neil. Object/relational mapping 2008: Hibernate and the entity data model (EDM). In *Proceedings of the 2008 ACM SIGMOD international conference on management of data*, pages 1351–1356, 2008.

[97] Renjun Pei, Aihua Shen, Mark J. Olah, Darko Stefanovic, Tilla Worgall, and Milan N. Stojanovic. High-resolution cross-reactive array for alkaloids. *Chemical Communications*, pages 3193–3195, 2009.

[98] Renjun Pei, Steven K. Taylor, Darko Stefanovic, Sergei Rudchenko, Tiffany E. Mitchell, and Milan N. Stojanovic. Behavior of polycatalytic assemblies in a substrate-displaying matrix. *Journal of the American Chemical Society*, 39(128):12693–12699, 2006.

[99] Charles S. Peskin, Garry M. Odell, and George F. Oster. Cellular motions and thermal fluctuations: the Brownian ratchet. *Biophysical Journal*, 65(1):316–324, 1993.

[100] Richard D. Piner, Jin Zhu, Feng Xu, Seunghun Hong, and Chad A. Mirkin. "Dip-pen" nanolithography. *Science*, 283(5402):661–663, 1999.



www.manaraa.com

[101] Andre V. Pinheiro, Dongran Han, William M. Shih, and Hao Yan. Challenges and opportunities for structural dna nanotechnology. *Nature Nanotechnology*, 6:763–772, 2011.

[102] Hong Qian, Michael P. Sheetz, and Elliot L. Elson. Single particle tracking: analysis of diffusion and flow in two-dimensional systems. *Biophysical Journal*, 60(4):910–921, 1991.

[103] Adrian E. Raftery and Steven M. Lewis. Implementing MCMC. In Walter R. Gilks, Sylvia S. Richardson, and J. D. Spiegelhalter, editors, *Markov Chain Monte Carlo in Practice*, pages 115–130. Chapman and Hall, 1996.

[104] D. Hugh Redelmeier. Counting polyominoes: yet another attack. *Discrete Mathematics*, 36:191–203, 1981.

[105] Sidney Redner. *A Guide to First-Passage Times*. Cambridge University Press, Cambridge, UK, 2001.

[106] Ken Ritchie, Xiao-Yuan Shan, Junko Kondo, Kokoro Iwasawa, Takahiro Fujiwara, and Akihiro Kusumi. Detection of non-brownian diffusion in the cell membrane in single molecule tracking. *Biophysical Journal*, 88(3):2266–2277, 2005.

[107] Christian Robert and George Casella. A short history of MCMC: Subjective recollections from incomplete data. In *Handbook of Markov Chain Monte Carlo*, pages 49–61. CRC, 2011.

[108] Robert Ross, Daniel Nurmi, Albert Cheng, and Michael Zingale. A case study in application I/O on Linux clusters. In *Proceedings of the 2001 ACM/IEEE conference on Supercomputing*, New York, NY, USA, 2001. ACM.

[109] Paul W. K. Rothemund. Folding DNA to create nanoscale shapes and patterns. *Nature*, 440:297–302, 2006.

[110] Michael J. Saxton and Ken Jacobson. Single-particle tracking: Applications to membrane dynamics. *Annual Review of Biophysics and Biomolecular Structure*, 26(1):373–399, 1997.

[111] Tim P. Schulze. Efficient kinetic Monte Carlo simulation. *Journal of Computational Physics*, 227(4):2455–2462, 2008.

[112] Oleg Semenov, Mark J. Olah, and Darko Stefanovic. Mechanism of diffusive transport in molecular spider models. *Physical Review E*, 83(2):021117, Feb 2011.



www.manaraa.com

[113] Oleg Semenov, Mark J. Olah, and Darko Stefanovic. Multiple molecular spiders with a single localized source – the one-dimensional case. In *DNA Computing and Molecular Programming*, volume 6937 of *Lecture Notes in Computer Science*, pages 204–216. Springer Berlin / Heidelberg, 2011.

[114] Oleg Semenov, Mark J. Olah, and Darko Stefanovic. Cooperative linear cargo transport with molecular spiders. *Natural Computing*, 2012. In publication.

[115] Alexander Slepoy, Aidan P. Thompson, and Steven J. Plimpton. A constant-time kinetic Monte Carlo algorithm for simulation of large biochemical reaction networks. *The Journal of Chemical Physics*, 128(20):205101, 2008.

[116] Milan N. Stojanovic, Tiffany Elizabeth Mitchell, and Darko Stefanovic. Deoxyribozyme-based logic gates. *Journal of the American Chemical Society*, 124(14):3555–3561, 2002.

[117] Milan N. Stojanovic and Darko Stefanovic. A deoxyribozyme-based molecular automaton. *Nature Biotechnology*, 21(9):1069–1074, September 2003.

[118] Milan N. Stojanovic, Darko Stefanovic, Thomas LaBean, and Hao Yan. *Computing with Nucleic Acids*, pages 427–455. Wiley, 2005.

[119] Karel Svoboda, Christoph F. Schmidt, Bruce J. Schnapp, and Steven M. Block. Direct observation of kinesin stepping by optical trapping interferometry. *Nature*, 365(6448):721–727, 1993.

[120] Michio Tomishige, Nico Stuurman, and Ronald D Vale. Single-molecule observations of neck linker conformational changes in the kinesin motor protein. *Nature Structural & Molecular biology*, 13(10):887–894, 2006.

[121] Erdal Toprak, Ahmet Yildiz, Melinda Tonks Hoffman, Steven S. Rosenfeld, and Paul R. Selvin. Why kinesin is so processive. *Proceedings of the National Academy of Sciences*, 106(31):12717–12722, 2009.

[122] Ronald D. Vale and Ronald A. Milligan. The way things move: looking under the hood of molecular motor proteins. *Science*, 288:88–95, 2000.

[123] Koen Visscher, Mark J. Schnitzer, and Steven M. Block. Single kinesin molecules studied with a molecular force clamp. *Nature*, 400(6740):184–189, 1999.

[124] Arthur Voter. Introduction to the kinetic Monte Carlo method. In Kurt Sickafus, Eugene Kotomin, and Blas Uberuaga, editors, *Radiation Effects in Solids*. Springer, 2007.



www.manaraa.com

[125] Brian Walsh. Markov chain Monte Carlo and Gibbs sampling. Lecture Notes for University of Arizona class EEB 581, 2004. `http://nitro.biosci.arizona.edu/courses/EEB596/handouts/Gibbs.pdf`.

[126] Roel Wieringa and Wierbren de Jonge. Object identifiers, keys, and surrogates—object identifiers revisited. *Theory and Practice of Object Systems*, 1(2):101–114, 1995.

[127] Oscar H. Willemsen, Margot M. E. Snel, Alessandra Cambi, Jan Greve, Bart G. De Grooth, and Carl G. Figdor. Biomolecular interactions measured by atomic force microscopy. *Biophysical Journal*, 79(6):3267–3281, 2000.

[128] Torsten Wittmann, Anthony Hyman, and Arshad Desai. The spindle: a dynamic assembly of microtubules and motors. *Nature Cell Biology*, 3(1):E28–34, 2001.

[129] Wenxia Ying, Gabriel Huerta, Stanly Steinberg, and Martha Zúñiga. Time series analysis of particle tracking data for molecular motion on the cell membrane. *Bulletin of Mathematical Biology*, 71(8):1967–2024, 2009.

[130] Matthew A. Young, G. Ravishanker, and D. L. Beveridge. A 5-nanosecond molecular dynamics trajectory for B-DNA: analysis of structure, motions, and solvation. *Biophysical Journal*, 73(5):2313–2336, 1997.



www.manaraa.com