UNIVERSITY OF CALIFORNIA,
IRVINE

Stochastic Parameterized Grammars:
Formalization, Inference and Modeling Applications

DISSERTATION

submitted in partial satisfaction of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

in Computer Science

by

Guy Yosiphon

Dissertation Committee:
Professor Eric Mjolsness, Chair
Professor Rina Dechter
Professor Arthur D. Lander

2009

DEDICATION

This dissertation is dedicated to my wife, Tali. I could not have made this effort without her support. Tali encouraged me to pursue my dreams and continue even during rough times. I thank Tali for putting up with my constant complaints and directing me to a positive and constructive path (while raising our daughter Liya). I also dedicate this dissertation to my parents who encouraged me to continue my academic studies in the US. I thank them for their advice and for sharing their experience with us.

TABLE OF CONTENTS

LIST OF FIGURES

ACKNOWLEDGMENTS

I would like to thank my advisor Eric Mjolsness, for his support over the years, and for giving me guidance and the freedom to explore related areas of research. Eric suggested that I should investigate this novel computational formalism. To my great satisfaction, this research exposed me to number of fascinating areas of computer science, biology, chemistry, and astronomy. I have greatly benefited from Eric's mentoring and his in-depth knowledge in all these research topics. I would also like to thank Eric for his patience and confidence in my capabilities.

I would like to thank my previous advisor and member of my thesis committee, Rina Dechter. Rina introduced me to Artificial Intelligence and related topics. I learned from Rina the essential tools to conduct scientific research. I also thank her for the time and advice in reviewing this thesis.

I would like to thank, Arthur Lander, a member of my thesis committee and a collaborator. The collaboration with Arthur, Kimberly Gokoffski, and Anne Calof was invaluable in conducting the olfactory epithelium modeling research. I would like to thank Arthur, Kim, and Anne for their patience and assistance with all the biological aspects of our study.

I would like to thank our collaborators in the root development model: Victoria Mironova, Vitaly Likhoshvai and Nadya Omelyanchuk. I thank Aaron Barth for the collaboration on spiral galaxy modeling. I appreciate the help and the numerous stimulating discussions with my friends and colleagues at UCI and the Scientific Inference Systems Laboratory. These include: Shyam Srinivasan, Yuanfeng Wang, Todd Johnson, David Orendorff, Shaohua Zhou, Ben Compani, Li Zhang, Michael Duff, Pawell Kurpinski, and Alex Sadovsky.

The work was supported in part by the National Institutes of Health (NIH) awards numbers P50-GM76516, P20-GM66051 and GM086883, and the National Science Foundations Frontiers in Biological Research (FIBR) program, award number EF-0330786.

CURRICULUM VITAE

Guy Yosiphon

EDUCATION

Doctor of Philosophy in Computer Science **2009**
University of California, Irvine *Irvine, California*

Master of Science in Computer Science **2005**
University of California, Irvine *Irvine, California*

Bachelor of Science in Computer Science **2000**
Tel Aviv Academic College *Tel Aviv, Israel*

RESEARCH EXPERIENCE

Senior Researcher **2009**
Utopia Compression *Los Angeles, California*

Graduate Research Assistant **2003–2009**
University of California, Irvine *Irvine, California*

TEACHING EXPERIENCE

Teacher Assistant **2003**
University of California, Irvine *Irvine, California*

SELECTED HONORS AND AWARDS

Graduate Research Fellowship **2002–2003**
University of California, Irvine

Scholarship for outstanding achievements in studies **1998–1999**
Tel Aviv Academic College

REFEREED JOURNAL PUBLICATIONS

Eric Mjolsness and Guy Yosiphon
Stochastic Process Semantics for Dynamical Grammars
Annals of Mathematics and Artificial Intelligence **2006, August**

BOOK CHAPTERS

Guy Yosiphon and Eric Mjolsness
Towards the Inference of Stochastic Biochemical Network and Parameterized Grammar Models
In Learning and Inference in Computational Systems Biology
N. Lawrence, et al., ed. MIT Press **2009**

CONFERENCES

Guy Yosiphon, Kimberly K. Gokoffski, Anne L. Calof, Arthur D. Lander and Eric Mjolsness
Stochastic Multiscale Modeling Methods for Stem Cell Niches
Machine Learning in Computational Biology workshop (NIPS-MLCB) **2007**

Guy Yosiphon, Kimberly K. Gokoffski, Anne L. Calof, Arthur D. Lander and Eric Mjolsness
Dynamical Grammar Modeling of Cellular Proliferative Dynamics in the Olfactory Epithelium
The 8th International Conference on Systems Biology **2007**

Guy Yosiphon
Automated Galaxy Morphology using Stochastic Parameterized Grammar
Second International Conference on Space Mission Challenges For Information Technology (SMC-IT 2006) **2006**

ABSTRACT OF THE DISSERTATION

Stochastic Parameterized Grammars:
Formalization, Inference and Modeling Applications

By

Guy Yosiphon

Doctor of Philosophy in Computer Science

University of California, Irvine, 2009

Professor Eric Mjolsness, Chair

The Stochastic Parameterized Grammar (SPG) forms a unifying framework for modeling systems of stochastic nature and dynamical structure. The modeling language is based on grammar-like collections of rewrite rules that define local interactions between objects or features and may involve creation or annihilation of objects. Local interactions may be either in the form of stochastic events or deterministic continuous dynamics. Such broad expressiveness makes the framework particularly suitable for applications in machine learning and multi-scale scientific modeling. This thesis introduces the SPG framework, its appropriate simulation methods and their computational cost. The problems of parameter learning and inference in SPG models are addressed by deriving exact and approximate sampling algorithms. As an application of SPG inference, the thesis includes an automated technique for inferring galaxy structure from images. The thesis concludes with two biological applications of the SPG framework: modeling root development and modeling the regeneration of neurons in the olfactory epithelium. The SPG model of the olfactory epithelium is a comprehensive spatial representation of the tissue that includes cellular level stochastic events and diffusion of signaling molecules. The model recapitulates the observed behavior of the tissue and provides interesting predictions about the dynamics of cell

population.

Chapter 1

Introduction

1.1 Motivation

Computational modeling and simulation may provide insights into the functionality of complex biological and physical systems. A computational framework for modeling such systems should take into account the interactions between components of different scales, from the molecular level to the cellular and tissue levels. In this thesis, we present the Stochastic Parameterized Grammars (SPGs) as a computational framework for modeling a broad class of stochastic processes with emphasis on biological applications.

Depending on the application and the required level of accuracy, molecules or cells may be represented by discrete quantities or approximated as continuous concentrations in the cellular or extracellular medium. In many applications, cells are depicted as individual objects with different properties as opposed to global quantities. The distribution of molecular concentrations can be approximated being uniform and well-stirred over the relevant space. However, in some biological processes, the distribution of signaling molecules is far from uniform in space. Such inhomogeneous distribution may be important for the correct behavior and functionality of the system.

Stochastic models may capture the dynamics of an underlying process when dealing with noisy or incomplete data which is a common case for biological systems data. When the concentration or quantity of molecules or cells is large enough, a deterministic mathematical model can provide a fairly accurate description of the real process. This is because the standard deviation of the number of reactions in a stochastic process that can be approximated by Poisson process ($O(\sqrt{\lambda t})$, where λ is the total stochastic rate and t is time) is negligible in comparison to the mean ($O(\lambda t)$) in large systems (as $\lambda t \to \infty$). However, many physical and biological systems are not large enough so that stochastic fluctuations can be discarded. The probabilistic nature of cell division or differentiation may have significant impact over the dynamics of small cell populations.

Consider modeling the development of the bacteria *Anabaena catenula.* The *Anabaena* is a type of filamentous cyanobacteria. Such a bacterium is constructed of a long chain of cells and it obtains energy through photosynthesis. There are two types of cells in the *Anabaena*: vegetative cells and heterocyst cells [89]. A vegetative cell synthesizes carbohydrate and may differentiate into a heterocyst cell or divide into two vegetative cells. The heterocyst cells are terminally differentiated cells (cannot divide anymore) which are specialized for nitrogen fixation. In spite of continuous growth and cell divisions, heterocyst cells emerge in relatively constant spacing of vegetative cells. On average, there are intervals of 9-15 vegetative cells separating heterocyst cells.

Some mathematical models [43] postulate that the robust pattern of heterocyst cells is regulated by a signaling molecule (NtcA protein, see [8]) which is secreted by heterocysts and diffuses along the cellular chain. The probability of vegetative cell's differentiation is correlated with the molecule concentration level.

The *Anabaena* model exemplifies the requirements for a comprehensive computational framework of developmental biological systems. The filament can be represented as a chain of individual cell objects. Each cell object is parameterized by type (vegetative or heterocyst), size, location (or connections to neighboring cells) and its compound concentration. Vegetative cells can grow according to a continuous growth function, and divide or differentiate. The vegetative cell's division and differentiation are both stochastic discrete events with probabilities that depend on the cell's compound concentration. Another probability distribution controls the size of each daughter cell. In such multi-cellular system, the amount of signaling molecule can be approximated by continuous concentrations and the molecule's secretion, diffusion and decay are described by a set of time dependent ordinary differential equations (ODEs).

The SPG framework enables the design of models that include parameterized objects, time-varying structure and stochastic "jump" reaction dynamics. SPGs are defined as a rule based language and have a stochastic process semantics which is independent of a specific simulation algorithm. Hybrid models which are composed of stochastic discrete events and differential equations with time derivatives can be specified as Dynamical Grammars (DGs), a generalization of SPGs.

The SPG modeling language provides a comprehensive computational framework for modeling complex systems in biology or other scientific domains. This thesis demonstrates the expressive power and intuitive representation of the SPG language. Furthermore, the mathematical definition of SPGs, which is decoupled from specific simulation algorithm, allows derivations of various simulation and inference algorithms.

1.2 Overview of the contributions of the thesis

The SPG formalism, some extensions such as the non-exists operator, and the SPG simulation algorithm were introduced in [93]. This thesis provides a comprehensive presentation of the SPG framework and an implementation of the simulation algorithm including the algorithm's computational aspects. A simulation algorithm for dynamical grammars is derived in this thesis. The thesis defines a set of new SPG modeling motifs (Chapter 6): complex chemical reactions grammar rules, reduction to deterministic model, rule schema for memory dependent processes, and meta-grammar rules. Furthermore, the semantics of new subgrammar recursive calls (subgrammar over the same state space) is defined here.

The approximate inference algorithm was presented in [137]. The exact inference algorithm for context-free SPGs is introduced in this thesis.

The thesis presents two new SPG applications: spiral galaxy modeling and feature extraction from galaxy images, and model of neurogenesis in the olfactory epithelium. Moreover, this thesis explores the behavior of cell populations and lineage trees in the olfactory epithelium model and their biological implications.

1.3 Overview of the thesis

The rest of the thesis defines the SPG formalism, introduces the relevant sampling and inference algorithms, and presents SPG models from both biological and physical domains. Chapter 2 provides background on the SPG formalism and related work. The chapter introduces generative grammars, probabilistic extensions of formal grammars and relevant formalisms for modeling stochastic processes.

Chapter 3 defines the SPG syntax and its continuous-time stochastic process semantics. We further develop the basic SPG framework by adding some useful extensions such as: constraints over rule parameters, not-exist operator, and recursive calls to subgrammars. Moreover, Section 3.4 provides the definition of dynamical grammars (DGs) as a generalization of SPGs which include continuous state-transitions that are formulated as sets of ordinary differential equations (ODEs). Finally, we describe DG models that incorporate a discretization of Partial Differential Equations (PDEs) for continuous spatio-temporal dynamics.

Derivation of the simulation algorithms from the SPG probability dynamics is performed in Chapter 4. Subsequently, the simulation algorithm of DGs is presented and extended to DGs containing PDEs. Computational aspects of the SPG and DG simulation software are discussed in Chapter 5.

Chapter 6 outlines some SPG modeling schemas of wide applicability. The chapter describes rules schemas for modeling complex chemical reactions, memory dependent processes, spatial interactions and meta-grammar rules. Moreover, the chapter includes a reduction to a deterministic model of ODEs.

Inference and parameter learning problems are addressed in Chapters 7 and 8. An exact inference method for context-free SPGs is outlined in Chapter 7. An approximate inference algorithm based on a Markov Chain Monte Carlo (MCMC) sampling method, that is applicable for any SPG, is presented in Chapter 8.

Chapters 9, 10 and 11 present three applications for SPG modeling. Chapter 9 discusses an SPG model for spiral galaxy structures. The spiral galaxy SPG is a purely structure based model with no relation to the current physical theory of galactic spiral arms formation. However, it can be useful for automatic inference of galaxy structure from images. The chapter discusses a specialized learning algorithm which is based on an approximation of the SPG model.

We present SPG models that recapitulate the development of tissues in two domains: plant root (in the *Arabidopsis*) and stem cell niche of the olfactory epithelium. Chapter 10 presents a one-dimensional model of root development. Cells are represented as objects that may continuously grow, divide in a random manner, and communicate by diffusion of a growth hormone (Auxin). The SPG model recreates the observed auxin profile and distribution of cell divisions.

The one-dimensional root model is expanded to a two-dimensional cell model of the olfactory epithelium (OE) in Chapter 11. The OE model includes stochastic cell divisions and differentiations, and communication between cells by diffusion of signaling molecules in the two-dimensional space. A finite difference PDE solver is used to numerically integrate the diffusion equations. The SPG model recapitulates the development and spatial arrangement of cells in the OE. Furthermore, the model predicts an interesting interaction between different cell lineages.

Chapter 2

Background

The SPG language provides a unifying modeling framework that integrates distinct formalisms: generative grammars, stochastic processes, and dynamical systems. This chapter provides a background on generative grammars and their computational hierarchy [24, 25], and on the probabilistic extensions of grammars as applied to natural language processing (NLP) [84], protein sequence analysis [39, 41] and computer vision [140]. We further explore other relevant formalisms that model stochastic or deterministic processes. We discuss the related formalisms' computational capabilities and limitations in modeling complex behavior.

2.1 Generative grammars

Generative (or transformational) grammars were developed by linguists [24, 25] in order to study the structure of natural and formal languages. Grammars describe the syntactical structure of an acceptable sentence (or string) in a formal language. A grammar includes a set of transformation rules that recursively describe the language.

The generative grammar is formally defined as the 4-tuple $G = (N, \Sigma, P, S)$ where N and Σ are disjoint sets, N is a finite set of nonterminal symbols , Σ is a finite set of terminal symbols (eg. words, letters, pixels), and P is a set of production rules of the form: $\alpha \rightarrow \beta$ such that α and β are strings of terminal and nonterminal symbols and β can be the empty string. S is a nonterminal that is designated as the start symbol.

For example, consider the alphabet $\Sigma = \{a, b\}$, a single nonterminal symbol S, ϵ denotes an empty string, and the following rules:

1. $S \rightarrow aSb$
2. $S \rightarrow \epsilon$

Applying the first grammar rule results in a symmetric expansion of the string whereas the second rule terminates the expansion, i.e. $S \Rightarrow_1 aSb \Rightarrow_1 aaSbb \Rightarrow_1 aaaSbbb \Rightarrow_2 aaabbb$. The grammar generates the language $a^n b^n : n \geq 0$. As a shorthand, multiple rules with the same LHS can be denoted as a single rule with all the RHS parts separated by $|$, e.g. the above grammar is $S \rightarrow aSb|\epsilon$.

Chomsky identified four classes of grammars [24, 25], known as the Chomsky hierarchy: regular, context-free, context-sensitive and unrestricted. Each grammar class is increasingly more expressive, meaning that it can generate a larger class of languages. Regular grammars, which are the most constrained, can only have rules with one nonterminal on the LHS and one terminal on the RHS that may be followed by one nonterminal (i.e. $A \rightarrow a$ or $A \rightarrow aB$). Regular languages are decided by a finite state automaton.

The context free grammars (CFG) may have rules that are composed of only one nonterminal on the LHS and unconstrained RHS, i.e. $A \rightarrow \alpha$ where α is a string of terminals and nonterminals. Context free languages express nested languages such as palindromes and programming languages. The parsing of a context free string is denoted as a parsing tree where each inner node denotes a nonterminal, the children of each inner node are the RHS symbols of the corresponding rule, and the leaves denote the terminal symbols. Context free languages are decided by a nondeterministic push-down automaton that represents an inefficient exponential size solution. However, the Cocke-Younger-Kasami (CYK) algorithm [138] provides an efficient polynomial time algorithm for parsing context free languages. The CYK algorithm is based on bottom-up parsing and dynamic programming.

Context-sensitive grammars allow rules of the type: $\alpha A\beta \to \alpha\gamma\beta$ where A is a non-terminal, α, β and γ are strings of nonterminals and terminals, and γ cannot be an empty string. Context-sensitive rules have RHS with at least as many symbols as the LHS. Therefore, the context-sensitive grammar's productions never shrink. The decision problem of context-sensitive languages is solved by linear bounded automata which means that the decision problem is in PSPACE-Complete. In other words, the decision problem requires only polynomial space however there is no known polynomial time algorithm.

Finally, Unrestricted grammars represent the set of recursively enumerable languages. Hence, in general, finding whether a string was generated by an unrestricted grammar is an undecidable problem.

2.2 Stochastic grammars

A grammar is an *ambiguous grammar* if there are multiple parsing paths for a given string. For example, the CFG: $S \to S + S|S * S|a$, has two different parsing trees for $a + a * a$. Multiple explanations and ambiguities arise in many real-world applications such as parsing natural language text (eg. the ambiguous sentence "I saw the man with the telescope") or the identification of objects and their relations in vision. Assigning probabilities to strings and their different parsing paths can be used as a way to resolve multiple explanations.

Stochastic context-free grammar (SCFGs) is an extension of CFG that associates probabilities with each rule. The probabilities of all the rules with the same LHS sum to 1, i.e. for every nonterminal N^i: $\sum_j P(N^i \to \alpha_j) = 1$. For example, a verb phrase (VP) [84] may have two outcomes with probabilities 0.7 and 0.3, as denoted by the following stochastic context free grammar rules:

1. $VP \to V\ NP$, 0.7
2. $VP \to VP\ PP$, 0.3

SCFGs are used in NLP [84] to infer the most probable semantics of a sentence. Apart from handling the ambiguity in many natural language sentences, SCFGs form a robust modeling framework that can handle grammatical mistakes. While CFG models can only ignore grammatical errors, SCFG models can handle such errors by associating low probabilities. SCFG are used in the analysis of RNA secondary structure [39, 41]. The secondary structure involves the folding of the single stranded RNA due to intramolecular forces which forms base pair connections. Frequently, base pairs occur in a nested form [39] in which case SCFG can accurately model the RNA secondary structure.

The probabilistic extension of regular grammars is equivalent to Hidden Markov models (HMM). Context-sensitive and unrestricted grammars can also be associated with probabilities. However, some context-sensitive and unrestricted grammars may not have a valid probabilistic interpretation since the number of applicable rules over the same substring may change according to the context (the current string state). The Stochastic Parameterized Grammars (SPG), which are introduced in the thesis, avoid this problem since the probabilities are dynamically defined according to the context.

A variant of the CYK parsing algorithm [138] can infer the most probable alignment of a string to SCFG. This algorithm is an extension of the Viterbi algorithm [127] which is used for HMM (or regular grammars) inference. In order to calculate the probability of a sequence given an SCFG, the inside-outside algorithm [11, 75] is used. The inside-outside algorithm which is analogous to the forward-backward algorithm for HMMs, is a part of an Expectation Maximization (EM) [34] algorithm for estimation of the SCFG parameters, analogous to the Baum-Welch algorithm [70] for HMMs. These SCFG inference algorithms will be discussed further in chapter 7 which introduces an exact inference algorithm for context-free SPG models.

2.3 Other related formalisms

In this section we provide an overview of other SPG related formalisms and their modeling capabilities and limitations.

Dynamical systems frameworks include ordinary differential equation systems, stochastic differential equations, and partial differential equations, all of which are continuous in time and state spaces and constant in their structure; spatial birth and death processes [104], multitype and spatial branching processes [74] and [121] which have continuous time but discrete state-transitions including changes in their number of variables; and cellular automata and branching processes [9] which are limited to discrete time and discrete states. These continuous time frameworks do not come with a formal language nor do they come with the powerful model composition operations that such a formal language provide (such as parallel rule lists and recursive subgrammar calls). By contrast, the semantics of SPG's are not limited to branching processes, but can also express multiple-input/multiple-output events.

In Artificial Intelligence (AI), there has been considerable work to extend the Bayes Network (BN) [99] framework in order to handle time-dependence and dynamic model structure. The Continuous Time Bayesian Networks of [98] are a form of BN that expresses a graphical decomposition of a Markov Chain rate table. These networks cannot model a dynamical system structure as can SPG's. The Dynamical Bayesian Network [31] is an extension of BNs for iterative time evolution processes. DBN's depict the probabilistic relations between variables within a time-slice and between adjacent time slices. However, DBN's can only represent constant size structures, meaning that objects cannot be created or annihilated. Another limitation of DBN's is that they can represent only discrete temporal processes. In a DBN, time is depicted by fixed intervals and therefore systems which are composed of processes that evolve with different time granularities or systems that obtain evidence over irregular time intervals become intractable for simulation and inference.

The Relational Dynamic Bayesian Networks framework [115] extends DBN's to first order logic domains so as to deal with creation of objects over time and with dynamical structure. Another language for coding probabilistic dynamical systems is BLOG [87]. Both of these frameworks can represent only discrete-time processes and discrete state-transitions.

A different paradigm is represented by Probabilistic Constraint Nets (PCN) [122] which extends Constraint Networks to model uncertainty over time. PCNs subsume DBNs since they are not restricted to discrete time structure. But as with DBN's, PCN's cannot represent the dynamics and evolution over the number of objects in a system.

Arising in other parts of computer science, the L-systems [106] and MGS [50] frameworks were developed for modeling of biological processes. They both define transformation rules similar to SPG's. However the semantics of these frameworks are defined in terms of simulation algorithms rather than a parallel mathematical semantics as with SPG's. L-systems were extended to include differential equation dynamics as "differential L-systems" [105], but this extension is still deterministic. A similar integration of deterministic grammars and gene regulation networks was proposed as a framework for modeling biological development in [90].

Process calculi, which were introduced for the study of concurrent computation, have also been adapted for applications in biology. Phillips and Cardelli [102] presented an abstract machine mechanism, to perform stochastic simulations from stochastic pi-calculus (SPC) models. The SPG framework is more flexible in its semantics since it is not bound to any specified stochastic simulation technique as is the SPC abstract machine, which is based on the Gillespie stochastic simulation algorithm. Moreover, the SPC abstract machine has no extension for handling objects with continuously-changing parameters as do Dynamical Grammars.

One graphical formalism developed for the analysis of concurrent computation is the Petri Net (PN) [101] which depicts the structure of a distributed system as a directed bipartite graph. This framework models the creation and annihilation of objects by a set of "tokens" which are transmitted between "place" nodes by the firing of "transition" nodes. There are many generalizations of PNs such as Colored PNs [65] or the Predicate/Transition Nets [49] that enhance PN with First Order Logic predicates. Since the execution of most forms of PNs is nondeterministic, the most relevant PN extensions are the stochastic or stochastic-colored PNs [60] that augment PNs with rates of execution. Nevertheless, the syntax of stochastic PNs does not contain features such as rule variables, firing rate functions, and PNs are defined only for discontinuous state transitions.

A well-published approach to developmental modeling in biology is the Cellular Potts Model (CPM) [58] based on Potts models of statistical mechanics. CPM is a lattice-based computational modeling method that is used to simulate the behavior of cellular structures. However, lattice-based frameworks are inaccurate for modeling multi-object systems that exhibit range of velocities. A high computational cost is required for maintaining an accurate depiction of the underlying process (if that is even possible), since the time interval for a whole grid update must be small enough to capture the highest relative speed in the process.

Another computational framework for modeling multicellular development was introduced in [44]. The internal state of each cell, in this framework, is represented by a differential equation, which is formulated as a sum of contributions from multiple sources. The framework was used in synthetic biology modeling, study of artificial evolution of neural networks and computer rendering of multicellular development.

Chapter 3

The Stochastic Parameterized Grammar formalism

Stochastic Parameterized Grammars (SPGs)[93] comprise a formal modeling language based on grammar-like collections of rewrite rules. SPGs have a stochastic process semantics in continuous time, which can be extended to discrete time. The essential idea is that there is a "pool" (unordered set) of fully specified parameter-bearing objects such as {bacterium[x], macrophage[y], redbloodcell[z]} where x, y and z are parameters such as position vectors. A grammar may include rules of the following type:

$$\{\text{bacterium}[x], \text{macrophage}[y]\} \to \text{macrophage}[y] \quad \textbf{with} \quad \rho(\|x - y\|)$$

The rule specifies the probability per unit time, ρ, that a macrophage ingests and destroys a bacterium as a function of the distance between their centers, $\|x - y\|$. The left hand side (LHS) of the rule is comprised of terms that are matched to a set of input parameterized objects. The terms on the right hand side (RHS) constitute the output objects which are constrained by the parameter matchings of the LHS terms. An SPG rule has the following general form:

$$\{\mathrm{A_1}[x_1], \mathrm{A_2}[x_2], ..., \mathrm{A_n}[x_n]\} \rightarrow \{\mathrm{B_1}[y_1], \mathrm{B_2}[y_2], ..., \mathrm{B_m}[y_m]\} \quad \textbf{with} \quad \rho(x, y) \tag{3.1}$$

$\mathrm{A_i}$ and $\mathrm{B_j}$ are the object types and x_i or y_i are the uninstantiated parameters' vectors of the ith term on the LHS or RHS, respectively. The number of terms on the LHS or RHS is finite and can be zero. The function ρ is a nonnegative function of the input and output parameters.

The SPG language is a generalization of the probabilistic formulation of chemical reactions. The grammar rules are reaction schemas where each rule term matches any object of the same type and according to the parameters' constraints. The system's state (or the "pool") is represented by a vector of copy numbers for every unique parameterized object. Thus, the state space may be infinite or even uncountable. An instantiated grammar rule represents a possible reaction that removes a set of reactants and creates the products.

Assuming there are no conflicts with other reactions, the waiting time for a reaction is exponentially distributed. The exponential rate is defined as the rule's rate function ρ times the number of distinct reactants (input objects) combinations in a given state, which is denoted by n, i.e. $w = n\rho$.

The SPG semantics can be summarized as a set of time dependent differential equations over the states' probabilities [126]:

$$\forall a \qquad \frac{d}{dt}P_a(t) = \sum_{b \neq a} W_{b,a}P_b(t) - W_{a,a}P_a(t), \qquad \text{where} \quad W_{a,a} = \sum_{b \neq a} W_{a,b} \tag{3.2}$$

Here, $P_a(t)$ denotes the probability of state a at time t. The summation is over all possible neighboring states where $W_{b,a}$ is a reaction rate, as defined above, from state b to state a. If $P(t)$ denotes the probability function over the entire state space then the system can be written in an operator form:

$$\dot{P}(t) = WP(t) \tag{3.3}$$

which has the formal solution:

$$P(t) = \exp(tW) \cdot P(0) \tag{3.4}$$

W is the probability rate operator which is composed of the corresponding transition rates, $W_{a,b}$, and the diagonal entries, $-W_{a,a}$.

The following section provides a formal definition of SPG syntax and stochastic process semantics. Section 3.2 presents some useful extensions for the SPG language: embedding constraints over rule variables, an existential operator over grammar objects, and subgrammar execution via SPG rules. The inclusion of continuous rules that contain differential equations dynamics is a powerful extension of SPGs, named *Dynamical Grammars* (DG), which is defined in Section 3.4.

3.1 SPG formal definition

This section first defines the SPG probability space $\{\Omega, \mathcal{F}, P\}$. Subsequently, the basic syntax for SPGs is introduced, followed by a semantics map to the time evolution (Hamiltonian) operator W.

Let $\mathcal{T}$ be a set of object types and $D = \prod_{i=1}^{d} D_i$, a set of measure spaces , where D_i is a measure space such as the real numbers $\mathbb{R}$ or some finite set. For each object type

$\tau \in \mathcal{T}$, let $k[\tau] \geq 0$ be the number of related parameters. The measure space of each object type is defined as $X[\tau] = \prod_{j=1}^{k[\tau]} X_j$, where $X_j \in D$. We denote a parameterized object as $\phi = \tau(x_1, ..., x_{k[\tau]})$.

A state of the system (or an element of the state space Ω) is an attachment of a copy number for every possible parameterized object. The state space Ω is defined as follows:

$$\Omega \quad = \mathbb{N}^{\prod_{i=1}^{m} X[\tau_i]}$$

where $m = |\mathcal{T}|$ (the number of object types), and τ_i is the ith object type.

Denote an element in the sample space as $\{n_\phi\} \in \Omega$, where n_ϕ is the copy number of object ϕ. $\mathcal{F}$ is defined as the appropriate σ-algebra for the set Ω.

The syntax of a grammar rule is defined as follows:

$$\{\tau_1[x_{1,1}, ..., x_{1,k[\tau_1]}], ..., \tau_n[x_{n,1}, ..., x_{n,k[\tau_n]}]\} \rightarrow \{\tilde{\tau}_1[y_{1,1}, ..., y_{1,k[\tilde{\tau}_1]}], ..., \tilde{\tau}_n[y_{m,1}, ..., y_{m,k[\tilde{\tau}_n]}]\}$$
$$\textbf{with} \quad \rho(x, y) \tag{3.5}$$

The SPG rule has a tuple (ordered list) of left hand side (LHS) terms, that represent the input items, followed by a tuple of right hand side (RHS) terms, that represent the output items. A rule term is a template for matching a parameterized object of the same type τ_i. Each parameter of input or output term (x,y) is an uninstantiated variable that is defined over the corresponding space of the object type parameters space, $X[\tau_i]$.

The rule is associated with a nonnegative *rate* function ρ over the ordered lists of input (LHS) x and output (RHS) y parameters:

$$\rho : \quad \left(\prod_{i=1}^{n} X[\tau_i] \right) \times \left(\prod_{i=1}^{m} X[\tilde{\tau}_i] \right) \longrightarrow \mathbb{R}^+$$

If ρ is integrable over the output parameters then it can be decomposed to: a rate function over only the input parameters, $\rho(x)$, and a conditional probability function over the output parameters:

$$\rho(x) \equiv \int \rho(x,y)dy \quad ; \quad P(y|x) \equiv \frac{\rho(x,y)}{\rho(x)}; \qquad \rho(x,y) = P(y|x) * \rho(x) \tag{3.6}$$

This decomposition is essential for an SPG time-forward simulation algorithm where the input parameters are known and the output parameters are sampled according to $P(y|x)$. The simulation algorithm will be described in detail in the next chapter.

An SPG is simply a collection of such rules defined over a common sample space, with the following syntax:

grammar-name = **grammar**[

$\{\tau_i^1[x_i^1]\} \rightarrow \{\tilde{\tau}_i^1[y_i^1]\}$ **with** $\rho^1(x^1, y^1)$

...

$\{\tau_i^n[x_i^n]\} \rightarrow \{\tilde{\tau}_i^n[y_i^n]\}$ **with** $\rho^n(x^n, y^n)$

]

The order of rules, which are indexed by the superscript, has no effect on the semantics, as defined below. The SPG header contains the keyword **grammar** followed by a grammar name and an optional header rule.

Now we formally define the semantics of the probability evolution operator W that was introduced in Equations 3.2 - 3.4. The evolution operator is constructed from elementary creation and annihilation operators which are similar to the creation and annihilation operators in physics and Quantum Field Theory [18]. In physics the creation and annihilation operators increase and decrease the number of particles or quantum of energy in a state. In the probabilistic settings of SPGs, these operators are used to direct the probability flow between states.

Prior to the creation and annihilation operators, we define the delta functions:

Definition: (delta functions) The Kronecker delta function is defined for discrete spaces as:

$$\delta_K(a, b) = \begin{cases} 1 & \text{if } x = y \\ 0 & \text{otherwise} \end{cases} \tag{3.7}$$

The Dirac delta (generalized) function, $\delta(x, y)$, with the appropriate measure μ is used in continuous spaces. The Dirac delta has the following property:

$$\int_{-\infty}^{\infty} f(x)\delta(x, y)dx = f(y)$$

A delta function between two objects ϕ and $\hat{\phi}$ is defined as:

$$\delta_o(\phi, \hat{\phi}) = \delta_K(\tau[\phi], \tau[\hat{\phi}]) * \prod_{i=1}^{k[\phi]} \delta(x_i[\phi], x_i[\hat{\phi}]) \tag{3.8}$$

where $\tau[\phi]$ is the object type and $x_i[\phi]$ is the ith parameter of the object.

Definition: (creation operator $\hat{a}(\phi)$) Given two objects ϕ and $\hat{\phi}$, and the previously defined probability space, the creation operator $\hat{a}(\phi)$ has the following effect:

$$\hat{a}(\phi)P(\{n_{\hat{\phi}}\}) = P(\{n_{\hat{\phi}} - \delta_o(\phi, \hat{\phi})\})$$

The creation operator shifts the probability from one state to a state that has an additional copy of the object ϕ.

Definition: (annihilation operator $a(\phi)$)

$$a(\phi)P(\{n_{\hat{\phi}}\}) = (1^T \cdot \{\delta_o(\phi, \hat{\phi})(n_{\hat{\phi}} + 1)\})P(\{n_{\hat{\phi}} + \delta_o(\phi, \hat{\phi})\})$$

The annihilation operator shifts the probability from one state to a state that has one less copy of the object ϕ. In addition, the probability is multiplied by the number of copies of object ϕ in the source state.

Definition: (copy-number operator $N(\phi)$)

$$N(\phi)P(\{n_{\hat{\phi}}\}) = \hat{a}(\phi)a(\phi)P(\{n_{\hat{\phi}}\}) = 1^T \cdot \{\delta_o(\phi, \hat{\phi})n_{\hat{\phi}}\} \cdot P(\{n_{\hat{\phi}}\})$$

The copy-number operator multiplies the state probability with the number of copies of object ϕ. This operator is equivalent to an application of the annihilation operator followed by the creation operator.

If the sample states are comprised of only a single object, the previous operations can be represented in the following matrix form:

$$\hat{a} = \begin{pmatrix} 0 & 0 & 0 & 0 & \cdots \\ 1 & 0 & 0 & 0 & \\ 0 & 1 & 0 & 0 & \\ 0 & 0 & 1 & 0 & \\ \vdots & & & \ddots & \ddots \end{pmatrix} \quad a = \begin{pmatrix} 0 & 1 & 0 & 0 & \cdots \\ 0 & 0 & 2 & 0 & \\ 0 & 0 & 0 & 3 & \\ 0 & 0 & 0 & 0 & \ddots \\ \vdots & & & & \ddots \end{pmatrix} \quad \tilde{n} = \begin{pmatrix} 0 & 0 & 0 & 0 & \cdots \\ 0 & 1 & 0 & 0 & \\ 0 & 0 & 2 & 0 & \\ 0 & 0 & 0 & 3 & \ddots \\ \vdots & & & & \ddots \end{pmatrix}$$

The transition of probability due to an SPG rule can be described as the application of annihilation operators followed by the application of creation operators. The rule operator given a set of input and output parameterized objects is defined as follows:

$$\hat{O}(x, y) = \rho(x, y) \left[\prod_{i=1}^{m} \hat{a}(\tilde{\tau}_i(y_i)) \right] \quad \left[\prod_{i=1}^{n} a(\tau_i(x_i)) \right] \tag{3.9}$$

The operator, $\hat{O}(x, \tilde{x})$, establishes the rate of probability transition from any source state into a target state where the LHS objects were removed and the RHS objects were added. The rate value is a product of the rate function, $\rho(x, y)$, and the number of matching input objects available in the source state. Note that when the rule's LHS matches k copies of the same object and the source state has n copies of the object then the rate function is multiplied by $n!/(n-k)!$, which is the number of ways of choosing sorted sets of k object copies. This definition agrees with the chemical Law of Mass Action and is valid for large n. In order to count only unsorted sets, $\binom{n}{k}$, the rate function should be multiplied by the factor $1/k!$, which is independent of n and the source state.

A rule operator is defined as the integral of the operator from Equation 3.9 over the entire parameter space:

$$\hat{O} = \int_{X[\tau_1]} ... \left(\prod_{i=1}^{n} dx_i \right) \int_{X[\tilde{\tau}_1]} ... \left(\prod_{i=1}^{m} dy_i \right) \hat{O}(x, y) \tag{3.10}$$

The operator addition or integration corresponds to independently firing processes, whereas the operator multiplication corresponds to obligatory event co-occurrence. The rule operator expresses only the flow of probabilities into new states and not the flow of probabilities out of old states. Including the probability outflow in our dynamical system is essential for probability conservation. Therefore, the rule operator is adjusted by subtracting the sum of outflow rate from the current state using diagonal matrix elements:

$$O = \hat{O} - \mathrm{diag}(1^T \cdot \hat{O}) = \hat{O} - \hat{D} \tag{3.11}$$

We can define $\hat{D}$ alternatively by using the copy-number operator:

$$\hat{D}(x, \tilde{x}) = \rho(x, \tilde{x}) \prod_{i=1}^{n} N(\tau_i(x_i)) \tag{3.12}$$

The time evolution operator of the entire SPG is simply a sum over the generators of each rule:

$$W = \sum_r O_r \tag{3.13}$$

This superposition implements the basic principle that every possible rule firing is an exponential process, happening in parallel with all the others until a firing event occurs. The operator semantics does not, in general, impose a unique preferred execution order on rule-firing events.

3.2 Language extensions

3.2.1 Constraints over rule parameters

Constraints over the input and output variables (x, y) in a grammar rule (Equation 3.5) may be incorporated in the rate function $\rho(x, y)$. Equality constraints, such as $f(x, y) = 0$, may be incorporated by multiplying ρ by the appropriate Dirac delta or Kronecker delta (Equation 3.7). Inequality constraints, such as $f(x, y) \leq 0$ and other Boolean combinations, can be similarly converted to numerical functions using the Heaviside function Θ:

$$\Theta(A) \equiv \begin{cases} 1 & \text{if predicate } A \text{ is true} \\ 0 & \text{otherwise} \end{cases} \tag{3.14}$$

As a convenient syntactic extension, equality constraints may be embedded in the rule terms in place of the constrained parameters. Given the constraint $\delta(x_{i,n} - c_{i,n})$ where $c_{i,n}$ is an expression over the rule's variables and constant values, the following shorthand notation is used:

$$\left\{\tau_i[x_{i,1}, ..., c_{i,n}, ..., x_{i,k[\tau_i]}]\right\} \rightarrow \left\{\tilde{\tau}_j[y_{j,1}, ..., y_{j,k[\tilde{\tau}_j]}]\right\} \quad \textbf{with} \;\; \rho(x, y)$$

The same notation is applicable for equality constraints in the RHS of the rule. All the auxiliary variables are integrated over in the rate operator as defined in Equation 3.10 for the rule's parameters.

Example 1 *The following rule matches only Cell objects such that their first parameter equals 1. If the first parameter denotes a cell type then this rule represents a cell differentiation event. A second equality constraint exists between the second parameter, x, of the input object and the second parameter of the output object.*

$$\mathrm{Cell}[1,x] \rightarrow \mathrm{Cell}[2,x] \quad \textbf{with} \ \ \rho(x)$$

Example 2 *This example demonstrates a constraint between variables in the LHS of the rule. The edge term's parameters must match the first parameters of the Node terms. This rule represents an event of detachment between two nodes whereas the stochastic rate is proportional to the squared distance between the nodes.*

$$\{\mathrm{Node}[i_1,x_1],\mathrm{Node}[i_2,x_2],\mathrm{edge}[i_1,i_2]\} \rightarrow \{\mathrm{Node}[i_1,x_1],\mathrm{Node}[i_2,x_2]\}$$
$$\textbf{with} \ \ \alpha(x_1-x_2)^2$$

The **subjectTo** keyword provides another syntactic extension which may be used to introduce inequalities or other complex Boolean constraints. Given a set of Boolean constraints, c, over the rule's parameters, the following notation is defined:

$$\left\{\tau_i[x_{i,1},...,x_{i,k[\tau_i]}]\right\} \rightarrow \left\{\tilde{\tau}_j[y_{j,1},...,y_{j,k[\tilde{\tau}_j]}]\right\} \quad \textbf{with} \ \ \rho(x,y) \ \ \textbf{subjectTo} \ \ c$$

Example 3

The following rule eliminates a Cell object as a function of the inverse cell size, r. Such rate function is only valid for positive cell size as declared by the **subjectTo** *constraint.*

$$\mathrm{Cell}[i,x,r] \rightarrow \{\} \quad \textbf{with} \ \ r^{-1} \ \ \textbf{subjectTo} \ \ \{r>0\}$$

3.2.2 Existential operator

Some events are triggered by the absence of an object. The basic SPG language can only support rules that require the existence of some objects. Here, the SPG syntax and semantics is extended to include a "Not Exists" quantifier, $\nexists$. In the following rule the ith term in the LHS indicates that the existence of any matching object (copy counter greater than 0) will render this rule as inapplicable.

$$\left\{\tau_1[x_{1,1}, ..., x_{1,k[\tau_1]}], ..., \nexists\tau_i[x_{i,1}, ..., x_{i,k[\tau_i]}], ..., \tau_n[x_{n,1}, ..., x_{n,k[\tau_n]}]\right\} \rightarrow \{\tilde{\tau}_j\,[y_j]\} \quad \textbf{with}\ \rho(x, y) \tag{3.15}$$

Definition: (*not-exists* operator $\varepsilon(\phi)$)

$$\varepsilon(\phi)P(\{n_{\hat{\phi}}\}) = (1^T \cdot \{\delta_o(\phi, \hat{\phi})\delta_K(n_{\hat{\phi}}, 0)\})P(\{n_{\hat{\phi}}\})$$

The *not-exists* operator is non-zero for states that do not contain a copy of the object, ϕ. The *not-exists* operator is applied inside the rule operator (Equation 3.9) as part of the product of annihilation operators. However, the annihilation and *not-exist* operators do not commute (where A and B commute if $AB = BA$). Therefore, the order in which *not-exists* operators are applied may affect the outcome. If the *not-exist* term precedes a regular input term that matches the same object then the rule is not applicable (the rule has no set of matching objects). If the *not-exist* term follows a regular input term that matches the same object then the rule is applicable to some states. In general, applicable states must have exactly the same number of copies of the *not-exist* object as the number of preceding regular terms that match the same object.

Example 4

The following rule describes an event of connecting two adjacent nodes. In order to avoid reconnecting the same nodes repeatedly, this rule is applicable only when there is no existing edge object that connects the two nodes.

$$\{\mathrm{Node}[i_1, x_1], \mathrm{Node}[i_2, x_2], \nexists \mathrm{edge}[i_1, i_2]\} \to \{\mathrm{Node}[i_1, x_1], \mathrm{Node}[i_2, x_2], \mathrm{edge}[i_1, i_2]\}$$
$$\quad \textbf{with} \ \ (x_1 - x_2)^{-2} \quad \textbf{subjectTo} \ \ \{x_1 \neq x_2\}$$

3.2.3 Subgrammar calls

An SPG prescribes a probability function and therefore can be used as part of a rate function in another calling SPG. A rate function that depends on another SPG is defined as:

$$\rho(\{\tau_i(x_i)\}, \{\tau_j(y_j)\} \quad ; t, W) = \delta(\{\tau_j(y_j)\}) \exp(tW) \delta(\{\tau_i(x_i)\}) \tag{3.16}$$

The rate function expresses the joint probability of an initial state $\{\tau_i(x_i)\}$ and final state $\{\tau_j(y_j)\}$ at time t according to the subgrammar (the called grammar), which is denoted by its operator W. This method is useful for multiscale modeling where the subgrammar may describe a set of stochastic reactions that occur in a faster time scale than the calling grammar. For example, an SPG that encodes cellular or tissue level stochastic events may call a subgrammar that encodes chemical reactions.

We define the **via** keyword as a syntactic extension for calling a subgrammar Γ. A single rule may have both a **via** and **with** clauses, in which case their firing rates are multiplied.

$$\left\{\tau_i[x_{i,1}, ..., x_{i,k[\tau_i]}]\right\} \to \left\{\tilde{\tau}_j[y_{j,1}, ..., y_{j,k[\tilde{\tau}_j]}]\right\} \quad \textbf{via} \ \ \Gamma(t) \ \ [\ \textbf{with} \ \ \rho(x,y)\ ] \tag{3.17}$$

The subgrammar called by a **via** clause operates on a different sample space than the calling grammar. We propose another scheme in which the subgrammar operates over the same sample space. The grammar rules, described thus far, encode probability transitions between states that differ by some set of objects matching a single rule. A subgrammar, that encapsulates many reactions over the same space, can express direct probability transitions between states that differ by dynamic sets of objects matching multiple reactions.

The syntax of a subgrammar call over a common sample-space is "**call** $\tilde{\Gamma}(t)$". This command may either replace a grammar rule, thus a standalone call, or follow a grammar rule. When we use a standalone **call** command with a subgrammar operator $\tilde{W}$, the operator, $\hat{O}$, in the calling grammar becomes: (instead of Equation 3.10)

$$\hat{O} = \exp(t\tilde{W}) \tag{3.18}$$

If the subgrammar call follows a grammar rule then the operator $\exp(t\tilde{W})$ is applied immediately after the rule operator:

$$\hat{O} = \exp(t\tilde{W}) \int_{X[\tau_1]} ... \left(\prod_{i=1}^{n} dx_i \right) \int_{X[\tilde{\tau}_1]} ... \left(\prod_{i=1}^{m} dy_i \right) \hat{O}(x, \tilde{x}) \tag{3.19}$$

This way, the subgrammar performs multiple reactions that can take place immediately after the rule execution. The rule may provide input data to the subgrammar in the form of RHS matching objects.

A generalization of the common sample-space subgrammar calls has the following syntax:

$$\textbf{call} \;\; \tilde{\Gamma}_1(t_1), \tilde{\Gamma}_2(t_2), ...$$

Subgrammars operators are applied according to their respective order. Such command may encode a sequence of events that occur immediately after the primary rule (in the calling grammar) has executed.

Example 5

This example shows the use of subgrammar calls for removing a variable number of objects in one event. The following rule removes a Node object and all of its edge connections. The Node object is replaced by a delNode object whereas the edges are removed by calling two subgrammars. The first subgrammar uses delNode as an input to remove all the edge. The second subgrammar removes the unused delNode object. Calling the two subgrammars with infinite time results in a sample from the steady state which is concentrated in a single point, hence a deterministic state. Using finite time may result in non-zero probabilities for invalid states, i.e. states that have edges for non-existing nodes.

$$\mathrm{Node}[i,x] \to \mathrm{delNode}[i,x]$$
$$\textbf{with}\ \ \rho(x)\ \ \textbf{call}\quad \mathrm{EdgeDelGrammar1}(\infty), \mathrm{EdgeDelGrammar2}(\infty)$$

EdgeDelGrammar1 $=$ ***grammar[***

$$\{\mathrm{delNode}[i,x], \mathrm{edge}[i,i_2]\} \to \{\mathrm{delNode}[i,x]\} \quad \textbf{with}\ \ k$$

$$\{\mathrm{delNode}[i,x], \mathrm{edge}[i_1,i]\} \to \{\mathrm{delNode}[i,x]\} \quad \textbf{with}\ \ k$$

]

EdgeDelGrammar2 $=$ ***grammar[***

$$\mathrm{delNode}[i,x] \to \{\} \quad \textbf{with}\ \ k$$

]

3.3 SPG illustration - clustering

This section provides an illustration of the expressive power of SPGs. The example has two parts: a generative process of hierarchical data structure creations and hierarchical clustering inference. The following single-rule grammar produces a hierarchical tree structure of clusters which is bounded by L, the tree depth. The deepest clusters, at level L, are the data items. The cluster distributions are Gaussians as encoded in the rate function. The **subjectTo** constraints limit the number of subclusters generated in each supercluster according to the size control variable, S.

ClusterGenerator = **grammar** [

(*Cluster generates a subcluster*)

$$\text{Cluster}[x, \sigma, l, s] \to \{\text{Cluster}[x, \sigma, l, s+1], \text{Cluster}[\tilde{x}, \sigma/V, l+1, 0]\}$$
$$\textbf{with} \;\; N(\tilde{x}; x, \sigma) \;\; \textbf{subjectTo} \;\; \{s \leq S \quad \text{and} \quad l < L\}$$

]

The clustering grammar represents an iterative procedure which is implicit in many clustering algorithms such as K-Means [82] and EM algorithm [34]. Each subcluster or data item is associated with a superlcuster (cluster of lower level). Subsequently, the new mean locations and variances are calculated. This sequence of steps is performed using the call operation to two subgrammars.

Each Cluster object is augmented with two additional parameters, b and χ . The first parameter, b, is a Boolean indicator for the assignment of the cluster to some super-cluster, whereas χ is a place holder for the sum of positions of all the assigned subclusters. The subgrammars are called with infinite time in order to obtain steady states which are concentrated in single (deterministic) points.

ClusteringGrammar $=$ **grammar**[

call Step1Grammar(∞), Step2Grammar(∞)

]

Step1Grammar $=$ **grammar**[

(*associate a subcluster to cluster*)

$$\{\text{Cluster}[x, \sigma, l, s, b, \chi], \text{Cluster}[\tilde{x}, \tilde{\sigma}, l+1, \tilde{s}, \tilde{b}, \tilde{\chi}]\} \rightarrow$$
$$\{\text{Cluster}[x, \sigma, l, s+1, b, \chi+\tilde{x}], \text{Cluster}[\tilde{x}, \tilde{\sigma}, l+1, \tilde{s}, 1, \tilde{\chi}]\}$$
$$\textbf{with} \quad N(\tilde{x}; x, \sigma) \quad \textbf{subjectTo} \quad \{s \leq S \quad \text{and} \quad l < L \text{ and } \tilde{b} = 0\}$$

]

Step2Grammar $=$ **grammar**[

(*compute new mean *)

$$\text{Cluster}[x, \sigma, l, s, b, \chi] \rightarrow \text{Cluster}[\chi/s, \sigma, l, 0, 0, 0]$$

$$\textbf{with} \quad 1 \quad \textbf{subjectTo} \quad \{s > 0 \quad \text{and} \quad l < L\}$$

]

3.4 Dynamical Grammar

Many complex processes, such as molecule signaling by diffusion, transport and decay, or object motion due to physical forces, are approximated by differential equations (DEs): ordinary DE (ODEs), partial DE (PDEs) or stochastic DE (SDEs). *Dynamical Grammars* (DGs) are defined as a generalization of SPGs that supports differential equations. DEs are encoded in **solving** clauses that follow a regular rule structure where the LHS and RHS terms are identical. The following rule is denoted as a *continuous-time rule*:

$$\left\{\tau_i[x_{i,1}, ..., x_{i,k[\tau_i]}]\right\} \to \left\{\tau_i[x_{i,1}, ..., x_{i,k[\tau_i]}]\right\} \quad \textbf{solving} \quad \left\{\frac{dx_{i,j}}{dt} = f_{i,j}(x)\right\} \tag{3.20}$$

The **solving** clause contains differential equation for each input variable, $x_{i,j}$, whereas x denotes the set of all input variables $x_{i,j}$. Therefore, each variable may be dependent on any or all of the input variables. Rule constraints, either in the form of **subjectTo** clause or inline equality constraints, are still applicable. A rule constraint is translated into the appropriate delta (or Heaviside) function which is multiplied with each gradient function, $f_{i,j}(\chi)$.

The continuous rules' semantics are defined by considering an SDE or equivalent Langevin equation which specializes to an ordinary differential equation when the stochastic term vanishes: $(\eta(t) = 0)$:

$$du_i = v_i(u)dt + \sigma_i(u)dw \quad \text{or} \tag{3.21}$$

$$\frac{du_i}{dt} = v_i(u) + \eta_i(t) \tag{3.22}$$

Under some conditions on the noise term $\eta(t)$ the dynamics can be expressed [113] as a Fokker-Planck equation for the probability distribution $P(u, t)$:

$$\frac{\partial P(u,t)}{\partial t} = -\sum_i \frac{\partial}{\partial u} v_i(u)P(u,t) + \sum_{ij} \frac{\partial^2}{\partial u_i \partial u_j} D_{ij}(u)P(u,t) \tag{3.23}$$

Therefore, the probability rate is comprised of the drift and diffusion rate operators:

$$\rho_{\text{drift}}(u) = -\sum_i \frac{\partial}{\partial u_i} v_i(u) \tag{3.24}$$

$$\rho_{\text{diffusion}}(u) = \sum_{ij} \frac{\partial^2}{\partial u_i \partial u_j} D_{ij}(u) \tag{3.25}$$

The drift rate operator can be expressed as an infinite dimension matrix in the limit $du \to \infty$:

$$\rho_{\text{drift}}(u) = -\begin{pmatrix} -\frac{v(u)}{du} & \frac{v(u+du)}{du} & 0 & 0 \\ 0 & -\frac{v(u)}{du} & \frac{v(u+du)}{du} & 0 \\ 0 & 0 & -\frac{v(u)}{du} & \ddots \\ 0 & 0 & 0 & \ddots \end{pmatrix}$$

Instead of using a derivative operation over the probability function we can define the drift and diffusion operators differently, by using the following relation:

$$-\frac{\partial}{\partial u} v(u) \;\; P(u,t) = -\int_{-\infty}^{\infty} \frac{\partial v(y)P(y,t)}{\partial y} \delta(y-u)dy = \int_{-\infty}^{\infty} v(y)P(y,t)\frac{\partial}{\partial y}\delta(y-u)dy$$

The delta function's fundamental derivative property [17] is used in the equation above. The drift and diffusion operators can be expressed by using the creation and annihilation operators over the continuous space:

$$O_{\text{drift}} = \int\int \hat{a}(y)a(u)v(y)\sum_i \frac{\partial}{\partial y_i}\delta(y-u)dy\; du \tag{3.26}$$

$$O_{\text{diffusion}} = \int\int \hat{a}(y)a(u)D_{ij}(y)\sum_{ij} \frac{\partial^2}{\partial y_i \partial y_j}\delta(y-u)dy\; du \tag{3.27}$$

The continuous rule operator, O^c, is defined as the sum of the drift and diffusion operators:

$$O^c = O_{\text{drift}}(u) + O_{\text{diffusion}}(u) \tag{3.28}$$

Example 6

A rule that describes continuous exponential growth for a cell:

$\mathrm{Cell}[x, r] \rightarrow \mathrm{Cell}[x, r]$

solving $\{\frac{dr}{dt} = kr\}$

Example 7

The following rule describes a repulsion force between two cells as a function of their distance. According to the defined semantics above, if there are multiple cells in the system, then, for each cell, the repulsion forces, from all other cells, will add up.

$\{\mathrm{Cell}[x_1, r_1], \mathrm{Cell}[x_2, r_2]\} \rightarrow \{\mathrm{Cell}[x_1, r_1], \mathrm{Cell}[x_2, r_2]\}$

solving $\{\frac{dx_1}{dt} = (x_1 - x_2)^{-1}\}$

3.4.1 Parabolic partial differential equations

In some modeling applications, the time dependent continuous concentration, $u(t)$, may vary as a function of the spatial coordinates, $u(x, t)$. Examples of such continuous spatial dynamics include fluid flow and the propagation of heat and sound. Parabolic Partial Differential Equations (PDEs) are used to formulate diffusion-dominated spatio-temporal dynamics (eg. heat and particle diffusion), as the following PDE for diffusion and drift (∇ is the differential operator over the spatial coordinates, x):

$$\frac{\partial u}{\partial t} = v(u) + D\nabla^2 u(x,t)$$

In general, time dependent PDEs have the following form:

$$\frac{\partial u}{\partial t} = F(\nabla, x, u)$$

The syntax of Dynamical Grammars, defined in the previous section, can be expanded to time dependent PDEs. When declaring continuous rules for PDEs, the parameter that corresponds to the dependent variable u represents a function over the spatial coordinates rather than a point.

The probability rate operators for PDEs can be derived by replacing each spatial function $u(x)$ with a set of variables, u_i, over the vertices of a uniform grid discretization of space. x_i denotes the positions of the grid for the one dimensional case, whereas h is the grid spacing. Hence, $x_i = ih$ and $u_i = u(x_i)$. When $h \to 0$, the spatial derivatives, ∇, may be approximated using the Taylor expansion:

$$\begin{aligned} u'(x_i) &= \frac{u(x_{i+1}) - u(x_{i-1})}{h} + O(h^3) \\ u''(x_i) &= \frac{u(x_{i+1}) - 2u(x_i) + u(x_{i-1})}{h^2} + O(h^4) \end{aligned} \tag{3.29}$$

Using Equation 3.29, PDEs can be approximated by sets of ODEs over the grid variables u_i. Subsequently, the previous section's operator derivation is applicable for this set of ODEs. Such discretization is directly expressed by a set of grammar elements, $\Gamma(u_i, i)$, representing the grid points and the following continuous rule between neighboring elements:

$$\{\Gamma[u_{i-1}, i-1], \Gamma[u_i, i], \Gamma[u_{i+1}, i+1]\} \rightarrow \{\Gamma[u_{i-1}, i-1], \Gamma[u_i, i], \Gamma[u_{i+1}, i+1]\}$$
$$\textbf{solving} \quad \left\{ \frac{d}{dt} u_i = g(u_{i-1}, u_i, u_{i+1}, h) \right\} \tag{3.30}$$

The function g is the finite difference discretization of the RHS part of Equation 3.29. We use the Crank-Nicolson discretization scheme [28] which is an implicit method in time and numerically stable (in most cases) for large timestamps [23]. Crank-Nicolson is second order accurate in both time and space.

Chapter 4

SPG simulation

4.1 Time ordered product expansion

It is impossible to derive a general algebraic solution of the probability function (Equation 3.4, also known as the master equation) or its moments since the state space is infinite or even uncountable. Sampling techniques are used in order to approximate the probability distribution. The Time-Ordered Product Expansion (TOPE) [40, 113] is a valuable tool for studying similar stochastic processes in physics and can be used to create samples from the master equation. The following is the TOPE of the master equation:

$$
\begin{aligned}
P(t) &= \exp(tW)\,P(0) = \exp\left(t\left(W_0 + W_1\right)\right)\cdot P(0) \\
&= \sum_{n=0}^{\infty} \Big[\int_0^t dt_1 \int_{t_1}^t dt_2 \cdots \int_{t_{n-1}}^t dt_n \\
&\qquad \exp((t-t_n)\,W_0)W_1\exp((t_n-t_{n-1})\,W_0)\cdots W_1\exp(t_1W_0)\Big]\cdot P(0) \qquad (4.1)
\end{aligned}
$$

W_0 is usually a solvable or easily computable part of W. An obvious choice is to take W_0 to be the diagonal part of W, in which case we derive Gillespie's well-known Stochastic Simulation Algorithm (SSA) for simulating chemical reaction networks [52]. The SSA algorithm generates a multi-reaction path by sampling in each iteration the next waiting time and reaction event. The TOPE with a diagonal W_0 interprets the probability of an arbitrary state a at time t as the sum of probabilities over all possible multi-reaction paths that arrive at state a and time t. The probability of a multi-reaction path can be written as:

$$P(.|a,k) = \mathcal{W}^k \circ P(.|a,0)$$

where

$$\mathcal{W}(b,t'|a,t) \approx W_{b,a} \exp(-\,(t'-t)\,D_{a,a})\mathbf{1}\,(t' \geqslant t)$$
$$D \equiv \mathrm{diag}(W)$$

In these equations we have defined $\Pr(a,t|b,k)$ to be the "just-reacted state probability": the probability of being at state b and time t immediately after the k'th reaction event, given the initial state a. These equations explicitly express the SSA algorithm as a discrete-time Markov chain representing a randomized algorithm. This expression is in accord with, for example, Theorem 10.1 of [131].

4.2 Dynamical Grammar simulation

A simulation algorithm of Dynamical Grammars (DGs) should take into account the continuous rules' operators of drift and diffusion (Equation 3.24 and Equation 3.25).

The TOPE can be used by including the continuous rules' operators as part of the operator H_0 which appears in the exponential. Following the simulation scheme of previous section, we denote $H_0 = D + \sum_r O_r^c$ where D is the discrete rules' diagonal operator and O_r^c is a continuous rule's operator (indexed by r) which is defined in Equation 3.28. Using this definition of H_0, the waiting time for the next discrete event in the sampling scheme is distributed according to:

$$\exp\left(t(D + \sum_r O_r^c)\right) \cdot \delta(\{n_\phi\}) \tag{4.2}$$

Let x denote a state space and $\rho(x)$ denote the diagonal element in D that corresponds to state x, i.e. the outflow rate in state x. Assuming the continuous operators involve only drift operators ($\eta(t) = 0$ in Equation 3.22), the resulting distribution (which is derived in Appendix A) is:

$$\exp(-\hat{\rho}(t))\delta(x - \hat{x}(t)) \tag{4.3}$$

where $\hat{x}$ and $\hat{\rho}$ are defined as:

$$\frac{d\hat{x}}{dt} = v(\hat{x}) \quad , \hat{x}(0) = x_0 \tag{4.4}$$

$$\hat{\rho}(t) = \int_0^t \rho(\hat{x}(\tau))\, d\tau \tag{4.5}$$

The sampling scheme of previous section is modified by integrating the differential equations (expressed by the continuous rules) between successive discrete events. Furthermore, the total outflow rate during the waiting time is expressed as an integration of the outflow rate over the continuous state path.

When dealing with SDEs, the expression in Equation 4.2 becomes analytically complicated since the diffusion term (Equation 3.25) expands the variance of the state distribution. However, for each simulation there is a single realization of the random function $\eta(t)$ (Equation 3.22). Therefore, the simulation algorithm is modified by initially generating a random realization of $\eta(t)$ and replacing Equation 4.4 with:

$$\frac{d\hat{x}}{dt} = v(\hat{x}) + \eta(t)$$

4.3 Simulation of Dynamical Grammars with PDEs

In Section 3.4.1, Dynamical Grammars were generalized to allow continuous rules with PDEs that include time and spatial derivatives (in particular, parabolic PDEs). The semantics of continuous rules with PDEs is defined by discretizing the space and replacing each PDE with a set of ODEs for the grid variables.

The Dynamical Grammar simulation algorithm, described in the previous section, applies to this space discretization. Numerical solution of the PDEs with time derivative can be computationally expensive. However, in many applications, the steady state solution (no time derivative), which is a boundary value problem, is sufficient and may provide substantial computational speedup. In biological modeling applications signaling molecules diffuse in space in a faster time scale than cell level events such as cell division or movement. The production, decay and diffusion of signaling molecules may reach steady state before any cell level event occurs.

The MultiGrid method [130] is an efficient numerical PDE solver method that is used primarily for boundary value problems. The finite differences discretization of a boundary value problem PDE (steady state solution) is denoted by this system of linear equations:

$$Au = f \tag{4.6}$$

Here u is the unknown spatial function whereas f is a source function. The matrix A is sparse and therefore Equation 4.6 can be computed by iterative methods such as Jacobi or Gauss-Seidel. The error term of the iterative method is $e = u - u^m$ where u^m is the solution vector of the m'th iteration. The error term can be represented by a Fourier series. It can be shown that iterative methods do converge ($e \to 0$ as $m \to \infty$) [132]. The rate of convergence is the slowest for the first Fourier mode (longest wavelength) whereas short wavelength Fourier modes converge rapidly. The MultiGrid method uses the differences in rates of convergence by using coarser grids in order to approximate the smooth error of long wavelengths. The correction is then propagated to finer grids where the short wavelengths error is reduced.

The MultiGrid method is further described in [109], which provides MultiGrid solver software that is used in our Dynamical Grammar simulation software *Plenum*.

A different method that is applicable for both steady state and time dependent problems is the Finite Element Method (FEM) [63]. FEM works as Finite Differences methods by discretizing the space. However, Finite Differences calculates the dependent variable over the grid points whereas FEM defines nodes and element trial functions that cover the whole space. The elements and the trial functions can be designed according to the complexity of the underlying medium.

As a simple example, we define the following linear one-dimensional elements:

$$N_1(x) = \frac{x_{j+1} - x}{x_{j+1} - x_j} \quad , \quad N_2(x) = \frac{x - x_j}{x_{j+1} - x_j}$$
$$u^*(x,t) = N_1(x)u_1(t) + N_2(x)u_2(t) \quad , \quad \text{for } x_j \leq x \leq x_{j+1}$$

Given a PDE that describes production, diffusion and decay:

$$\frac{\partial u}{\partial t} = v(x) + D\nabla^2 u(x,t) - ku \quad ,$$

we can derive the following equation for each node, u_i , where L is the element size.

$$L(\dot{u}_{i-1} + 4\dot{u}_i + \dot{u}_{i+1}) + \frac{D}{L}(-u_{i-1} + 2u_i + u_{i+1}) + kL(u_{i-1} + 4u_i + u_{i+1}) = v_i$$

This set of ordinary differential equations can be solved using any finite difference method.

The FEM formulation is appealing for biological modeling since elements may correspond to cells or organs. As opposed to grid methods that naively discretize the underlying space, the FEM elements may change and develop along with their corresponding physical entity, such as cells.

The formulation of FEM and Finite Differences can be declared internally by the DG simulator or externally by the DG model. A grammar rule for Finite Differences was described in Equation 3.30.

Chapter 5

Simulator implementation

A Dynamical Grammar interpreter, called *Plenum*, was developed in *Mathematica*, the computer algebra system. *Plenum* includes the simulation algorithm that was derived in Chapter 4 and a PDE Finite difference solver. This chapter discusses algotihmic design issues that emerged while developing the interpreter and DG models that are described in the following chapters. Section 5.1 describes the constraint processing methods for matching sets of objects to the LHS of grammar rules. Section 5.2 provides the overall complexity analysis of the simulation algorithm. The simulation of SPGs, that include output parameters in the rate functions, is discussed in Section 5.3. Simulation of continuous-time rules is described in Section 5.4.

5.1 Matching objects tuples to rules

After every discrete event, the simulation algorithm attempts to match sets of objects from the pool to the LHS of each grammar rule. The procedure which finds all the sets of objects that match the rule's LHS elements and satisfy their constraints (as defined in Section 3.2.1) can be divided to two phases. The first phase matches a tuple of arbitrary objects to some or all of the elements in the LHS, whereas the second part enumerate all candidate objects' tuples for complete or partial matching of the LHS. In order to match an objects' tuple to an elements' tuple, *Plenum* uses, in most cases, *Mathematica*'s built-in pattern matching method. This procedure works for simple elements that have only a variable or constant value for each parameter. However, if an LHS element's parameter is a mathematical expression then this direct procedure is insufficient. For example:

$$\{\text{TreeNode}[l, x_1], \text{TreeNode}[l{+}1, x_2]\} \to \{\text{TreeNode}[l{+}1, x_1], \text{TreeNode}[l, x_2]\} \quad \textbf{with} \quad k$$

The first parameter of a TreeNode object denotes its level in the tree. The rule switches tree levels of two TreeNode's that differ by one level. In order to process constraints such as the level parameter of the second LHS element (l + 1), *Plenum* generates a set of equations in which the equations' LHS comprise the set of elements' parameters, whereas the equations' RHS is the set of candidate object parameters. The set of objects matches if and only if there are one or more solutions for this set of equations. *Plenum* utilizes *Mathematica*'s equation solver(Solve [133]) for this task.

The full enumeration of all possible object tuples for each rule may be prohibitive and unnecassery. Consider the following example of a rule that eliminates a set of three connected nodes.

$$\{\text{Node}[i_1, x_1], \text{edge}[i_1, i_2], \text{Node}[i_2, x_2], \text{edge}[i_2, i_3], \text{Node}[i_3, x_3]\} \to \{\} \quad \textbf{with} \quad k$$

A naive algorithm may construct a list of all node triplets, whereas the only node triplets that satisfy the rule's constraints form a connected and directed row. For any graph that is not fully connected, such implementation has an unnecessarily high computational cost. If there is a mismatch between a partial object tuple and the corresponding LHS elements, then there is no need to test the complete tuples.

After each reaction event, every new tuple of matching objects contains at least one new object. Therefore, the tuple enumeration scheme should not reconsider tuples that consists of existing objects only.

The following Generate Tuples Algorithm is implemented in *Plenum*. The input to the algorithm is a set of new and existing matching objects per LHS element. The algorithm maintains a list of partial tuples over the first i elements which are then extended or eliminated by considering the objects of the next (i + 1'st) element. In addition, the algorithm enumerates only tuples that have some new objects by combining the lists of new and existing objects.

Generate Tuples Algorithm:

Input: $LHS[r]$*: set of LHS elements of rule* r*.*

$LHS[r]_{1 \ldots i}$ *denotes the list of the first* i *elements.*

$|LHS[r]|$ *denotes the number of elements.*

Θ_i^r *: List of matching objects in the pool for each LHS element* i *of rule* r*.*

$\hat{\Theta}_i^r$ *: List of matching new objects for each LHS element* i *of rule* r*.*

Initialize: For each r, i *- concatenate the two lists :* $\bar{\Theta}_i^r = \Theta_i^r \cup \hat{\Theta}_i^r$

Set of new tuples : $\bar{T} = \{\}$

Iteration: For each rule r*, For* $i = 1$ *to* $|LHS[r]|$

initialize list of new tuples $T = \{\}$

For $j = 1$ *to* $|LHS[r]|$

If $j < i$ *then* $a = \Theta_i^r$*, else If* $j = i$ *then* $a = \hat{\Theta}_i^r$ *, else* $a = \bar{\Theta}_i^r$

create new set of tuples by appending a new object to each candidate

$T = T \times a$*, where* $\times$ *denotes the Cartesian product*

For each tuple τ *in* T

If τ *matches* $LHS[r]_{1 \ldots j}$ *continue, else remove* τ *from* T

add the new tuples : $\bar{T} = \bar{T} \cup T$

return $\bar{T}$

In the above algorithm, the elements' ordering of tuple construction, indexed by j, is predetermined by the elements' order in the LHS. However, this ordering may re many partial tuples that do not include new objects and can not be extended to full tuples when considering the new objects. Therefore, as a heuristic, the ordering may be modified so that, in every partial tuple, the first object is new, i.e. change the ordering statement such that $a = \hat{\Theta}_i^r$ for $j = 1$. This heuristic is implemented in *Plenum.*

If a rule includes *not-exists* elements (see Section 3.2.2) then a tuple that matches all the regular elements may trigger a reaction only if there are no objects that complete the tuple for the remaining *not-exists* elements. For every such regular elements tuple, *Plenum* maintains a list of matching *not-exists* tuples. A regular elements tuple is considered valid whenever its *not-exists* tuples' list is empty.

The Rete algorithm [45] is a pattern matching algorithm for performing forward chaining in production rule systems such as Ops5 [19] and Soar [71]. The algorithm maintains a list of objects matching each rule element, as in the above algorithm. In addition, Rete stores the partial tuples that were discarded in the foregoing Generate Tuples Algorithm. Storing the partial tuples may save considerable processing time when generating new tuples and partial tuples. However, the increased space complexity may be prohibitive and there is additional processing time for deleting the relevant partial tuples of removed objects.

The problem of generating tuples over the set of constraints imposed by the rule's elements is essentially a constraint satisfaction problem (CSP) [33]. The Bucket Elimination (BE) [32] algorithm is an optimal dynamical programming type algorithm for solving CSPs. In this algorithm, the variables, which are the rule elements in our application, are arranged in sequential order. Each variable is associated with a bucket (abstract container), whereas the constraints are assigned to the first bucket in the ordering that includes one of their variables. Each bucket is processed thus creating a set of partial tuples that satisfy all these constraints. The set of partial tuples constitutes a new discrete constraint which is propagated to the next bucket that includes one of its variables.

The time and space complexity of the BE procedure is bounded by the size of the largest discrete constraint that was generated. The constraint size is exponential in the number of variables involved. Every variable ordering may produce a different maximal constraint size, which is also known as the induced width of the ordered constraint graph (network) [33]. Finding the minimum induced width of a graph is hard (NP-complete). However there are some heuristics [33] that usually produce close approximations.

The BE algorithm which generates tuples in a tree-like structure that may provide computational speedup over the sequential approach of Generate Tuples Algorithm. BE implementation can be embedded in the inner loop of Generate Tuples Algorithm by replacing the sequential construction of tuples. Since the BE algorithm stores all the partial tuples that are associated with each bucket, the space complexity might be prohibitive in some cases. In addition, the computational time that is gained by using the BE algorithm is insignificant for constraint problems with few variables whereas most SPG rules used in this thesis have at most four LHS elements, which are the constraints' variables.

5.2 Computational complexity of the simulation algorithm

This section analyzes the overall computational complexity of the simulation algorithm's implementation in *Plenum*. The simulation algorithm maintains a data structure of pending reactions and their rates. Each reaction step consists of: a search for the new executed reaction, execution of the reaction, creation of new pending reactions, and the modification of affected pending reactions. Let $\hat{r}(t)$ be the set of pending reactions at time t and $M(t)$ is the set of unique objects that exist at time t. Note that $|\hat{r}(t)|$ is bounded by $\sum\limits_k \binom{M(t)}{|\,\text{reactant}(k)|}$, where $\text{reactant}(k)$ is the set of LHS reactant elements (or input elements) of the kth grammar rule. The unique objects could be stored in an array where each entry holds a link to the set of relevant pending reactions (all the reactions for which the current object is the first reactant). In many stochastic processes (for example, various gene regulation network models), the total number of unique objects over the entire process, $M = \cup_t(M(t))$, is finite, relatively small and known in advance. Thus, the objects' array can include the entire set of unique objects and the space complexity is $O(\max\limits_t |\hat{r}(t)| + |M|)$. In other cases, where $|M|$ is too large or unpredictable, an alternative for the array is a hash table of fixed size.

The search for a random reaction may take $O(|\hat{r}(t)|)$, but using a binary tree reduces the search time to $O(\log |\hat{r}(t)|)$. The tree leaves refer to pending reactions, whereas the inner nodes store the total rate and number of reactions (leaves) in their subtree. New pending reactions are inserted into the subtree (right or left) that has fewer leaves. That way, the binary tree is balanced throughout the simulation with a logarithmic height.

Once a reaction, denoted as $\overset{*}{r}$, is executed, the algorithm updates the affected pending reactions which are accessed directly by the array (or hash table) structure. The set of affected reactions is $\tilde{r} = \{r_i | a \in \overset{*}{r} \wedge a \in r_i\}$, or in other words, the set of reactions that share an object, a, with the executed reaction. Therefore this procedure takes $O(|\tilde{r}|)$ time. The number of new reactant tuples and Generate Tuples Algorithm is bounded exponentially by the size of the LHS of the largest rule.

Finally, the total number of reactions events depends on the total time T, the rates of each reaction $\rho(\hat{r}_i)$, and the number of possible reactants' tuples per reaction $\prod_{j \in \hat{r}_i} N_j$ (where N_j is the number of identical objects that match the jth reactant). A bound on the expected number of reaction events is : $O(T \cdot \max_i(\rho(\hat{r}_i) \prod_{i \in \hat{r}_j} N_j))$.

The computational bounds of the simulation algorithm are summarized in the following theorem.

Theorem 5.2.1 *The expected time complexity of the SPG simulation algorithm is* $O(\Psi \cdot \hat{r}_{max})$ *where* $\Psi = T \cdot \max_i(\rho(\hat{r}_i) \prod_{i \in \hat{r}_j} N_j)$ *and* $\hat{r}_{max} = \max_t |\hat{r}(t)|$. *The space complexity of the SPG simulation algorithm is* $O(\hat{r}_{max})$.

5.3 Handling output parameters

The rate function is defined over both input and output parameters. During a simulation, the input (or LHS) parameters are assigned according to the tuple of matching objects, but the output parameters are unknown. A rule with output parameters represents infinitely many potential transitions from a single input state to the whole space of output objects. We decompose the rate function into the product of: a rate function over the input parameters and a conditional probability of the output parameters:

$$\rho(x,y) = P(y|x) * \rho(x) \qquad \text{where} \qquad \rho(x) \equiv \int \rho(x,y)dy \quad ; \quad P(y|x) \equiv \frac{\rho(x,y)}{\rho(x)}$$

The simulation algorithm chooses the input objects tuple, i, according to the discrete distribution of the normalized input rate functions: $\rho(x_i)/\sum_i \rho(x_i)$. Given the input tuple, the algorithm samples from the conditional probability, $P(y|x_i)$, and creates the output objects. An applicable sampling method is the probability integral transform [35]:

- Sample $u \sim \text{Uniform}[0,1]$
- Return $x = F^{-1}(u)$

Here, F is the cumulative distribution function (CDF). The simulation algorithm, in *Plenum*, computes the CDF of the conditional probability by numerical integration over the relevant domain. All the integrations that create the conditional probability function and later its CDF, are performed as a preprocessing step. The algebraic transformations are performed using built-in *Mathematica* functions.

5.4 Integration of Dynamical Grammar rate functions

In case there are matching tuples to continuous rules, the simulator has to integrate (solve) all the relevant object parameters according to the **solving** time-dependent differential equations. In the following equations x is the set of continuous parameters,

v is the relevant set of rate functions, and x_0 is the initial condition:

$$\frac{dx}{dt} = v(x) \quad , x(0) = x_0$$

Plenum uses *Mathematica*'s numerical ODE solver (NDSolve). The rate functions of discrete events may depend on continuously changing parameters, so the simulator integrates the rate functions as well. An additional ODE describes the exponential decay of the survival probability for avoiding the next reaction firing event:

$$\frac{dP}{dt} = -P \sum_i \rho_i(x(t)) \quad , \quad P(0) = 1$$

ρ_i is the rate function of the ith candidate reaction. A derivation of these equations from the TOPE is described in Section 4.2.

Sampling a reaction time from P is similar to the previous section's probability integral transform. Generate a uniform number, u, from [0,1] and find the time, t, for which $P(t) = u$. The ODE solver integrates until it reaches the event time. Finally, the discrete event is sampled from the distribution of the normalized rates at the time of event: $\rho_j(x(t))/\sum_i \rho_i(x(t))$.

Chapter 6

SPG modeling motifs

This chapter presents several SPG modeling "motifs". An SPG motif is a schema of grammar rules that represents recurring patterns of use. Section 6.1 shows SPG schema for modeling complex chemical reactions. In Section 6.2, we describe a dynamical grammar schema for modeling processes with memory in contrast to the SPG semantics that results in memoryless exponential distributions for elementary events. Section 6.3 presents dynamical grammar rules that define the basic kinetics of movement caused by forces between adjacent cells in accord with a weak spring potential. The weak spring rules were incorporated in the spatial biological models of the following chapters. A procedure that transforms an SPG model to a deterministic approximation based on ODEs is presented in Section 6.4. Finally, Section 6.5 discusses the capability to create SPGs that manipulate other SPG models.

6.1 Complex chemical reactions

Ordinary chemical reactions involve sets of rules between unparameterized objects that denote different molecules, such as:

$$2H_2O \rightarrow 2H_2, O_2$$

However, in many biological applications, chemical reactions involve numerous chemical complexes which may affect the local interactions. For example, a ligand that has n binding sites may bind to a receptor molecule that has m binding sites. The rate of such reactions may depend on the number of free binding sites. This results in an exponential scaling of the number of different reactions between ligands and receptors. Modeling these reactions in the conventional way would be impractical.

The following is an SPG rule for the binding of ligand, 'L', and receptor, 'R', over one of their free binding sites, 'B'. The ligand and receptor objects are associated with a set of binding site objects. The ligand-receptor models, described in this section, are adapted from the models published in [27]. The ligand-receptor models in [27] were described using the BioNetGen language (BNGL) [15].

$$\begin{aligned}\{r = \mathrm{R}[i_1], \mathrm{B}[i_1, ib_1, 0, 0], l = \mathrm{L}[i_2], \mathrm{B}[i_2, ib_2, 0, 0]\} \rightarrow \\ \{r, \mathrm{B}[i_1, ib_1, i_2, ib_2], l, \mathrm{B}[i_2, ib_2, i_1, ib_1]\} \\ \textbf{with} \quad k\end{aligned} \tag{6.1}$$

The i parameters denote identification (ID) numbers for the relevant objects. Object

ID numbers, as part of SPG models, are discussed in [93] as a way to define graph grammars. Binding site objects have ID numbers for the source object, source binding site, target object, and target binding site. The last binding site object in the rule represents a free binding site since a '0' ID number is not associated with any target object. The SPG rule describes a binding event between two free binding sites of a ligand and a receptor such that the ligand has an additional binding site which is already occupied (as defined by the inequality constraint).

The probability of chemical reactions may depend on the intricate structure of underlying chemical complexes. For example, ligand and receptor may belong to the same complex not because of direct connection but via intermediate connection with other ligands and receptors. In such cases, the possibility of ligand-receptor binding is diminished due to the structural constraints of their mutual complex. Modeling this type of reaction requires a recursive language that can identify dynamic structures. The following SPG rules describe a ligand-receptor model in which a ligand and a receptor may bind only if they belong to different complexes.

(*binding ligand-receptor: *)

$$\begin{aligned}&\{\mathrm{R}[i_1, c_1], \mathrm{B}[i_1, ib_1, 0, 0], \mathrm{L}[i_2, c_2], \mathrm{B}[i_2, ib_2, 0, 0]\} \rightarrow \\ &\qquad \{\mathrm{R}[i_1, max(c_1, c_2)], \mathrm{B}[i_1, ib_1, i_2, ib_2], \mathrm{L}[i_2, max(c_1, c_2)], \mathrm{B}[i_2, ib_2, i_1, ib_1]\} \\ &\qquad\qquad \textbf{with} \quad k_1 \quad \textbf{subjectTo} \quad c_1 \neq c_2\end{aligned} \tag{6.2}$$

(*disassociation of ligand-receptor: *)

$$\begin{aligned}&\{\mathrm{R}[i_1, c_1], \mathrm{B}[i_1, ib_1, i_2, ib_2], \mathrm{L}[i_2, c_1], \mathrm{B}[i_2, ib_2, i_1, ib_1]\} \rightarrow \\ &\qquad \{\mathrm{R}[i_1, c_1], \mathrm{B}[i_1, ib_1, 0, 0], \mathrm{L}[i_2, c^*], \mathrm{B}[i_2, ib_2, 0, 0]\} \\ &\qquad\qquad \textbf{with} \quad k_2\end{aligned} \tag{6.3}$$

(*propagate new complex ID: *)

$$\{\mathrm{R}[i_1, c_1],\mathrm{B}[i_1, ib_1, i_2, ib_2], \mathrm{L}[i_2, c_2], \mathrm{B}[i_2, ib_2, i_1, ib_1]\} \rightarrow$$
$$\{\mathrm{R}[i_1, max(c_1, c_2)], \mathrm{B}[i_1, ib_1, i_2, ib_2], \mathrm{L}[i_2, max(c_1, c_2)], \mathrm{B}[i_2, ib_2, i_1, ib_1]\} \quad (6.4)$$
$$\textbf{with} \quad \infty \quad \textbf{subjectTo} \quad c_1 \neq c_2$$

Each ligand and receptor object includes a reference to its complex ID. The rule of Equation 6.2 denotes a binding event of ligand and receptor from different complexes. Both objects are assigned with the same complex ID ($max(c_1, c_2)$). Equation 6.3 defines the reverse event of disassociation. c^* denotes a new complex ID. An ID generator to support c^* can be defined by adding an object that produces new ID number in every disassociation event.

Equation 6.3 defines a rule that updates all the complex objects whenever there is a mismatch of complex ID. The mismatch occurs immediately after binding or disassociation events. The rule uses $max(c_1, c_2)$ to resolve the conflict between the two connected objects. This deterministic resolution is required since a random assignment to one of the complex IDs may cause an infinite update loop (A updates B, then C updates B).

The ID propagation rule (Equation 6.3) may be replaced by a call to an ID propagation subgrammar (see Equation 3.18). Such subgrammar will include only the propagation rule and the subgrammar call should be included in both the binding and disassociation rules.

6.2 Rule schema for memory dependent processes

SPGs model memoryless processes since reaction waiting times are distributed according to exponential distributions. However, in many domains, the processes are not memoryless. Power law distributions such as the Pareto distribution are used for approximation of computer network traffic [80, 79] and various applications in biology and neuroscience [124, 78]. A Pareto distribution has the following cumulative distribution function (CDF) and probability density function (PDF):

$$F(t) = 1 - (\frac{t_0}{t})^k \tag{6.5}$$

$$p(t) = k(\frac{t_0^k}{t^{k+1}}) \tag{6.6}$$

where $t_0 > 0$ is the minimum possible value of t.

The following dynamical grammar rules describe an event (cell death) where the waiting time follows a Pareto distribution. Each Cell object is associated with a life span parameter, τ, which evolves according to the first continuous rule. τ is initialized in the creation of the Cell object. This initialization is equivalent to the minimal time parameter t_0 in Equation 6.5.

$\mathrm{Cell}[x, \tau] \rightarrow \mathrm{Cell}[x, \tau]$

  solving $\{\frac{d\tau}{dt} = 1\}$

$\mathrm{Cell}[x, \tau] \rightarrow \{\}$

  with k/τ

In contrast to the above Pareto rule in which the rate decreases over time, we may define rules that have increasing firing rates. A delayed rule has a firing rate that increases abruptly from 0 to some constant after a predefined delay time. Such rule can be declared using the Heaviside Step function from Equation 3.14. For example, see the following delayed rule for a cell death event after t_{delay} time:

$\text{Cell}[x, \tau] \rightarrow \{\}$

$\quad\quad$ **with** $k\ \Theta(\tau - t_{delay})$

The expected waiting time for a delayed rule is: $t_{delay} + \frac{1}{k}$, whereas the variance, which results only from the homogeneous exponential part, is: $\frac{1}{k^2}$. Thus, as k increases to infinity, the rule converges to a deterministic delayed process.

In general, the cumulative rate of a valid rule must converge to infinity: $\int_0^\infty \rho(t)dt = \infty$; otherwise, the waiting time's CDF will not converge to 1.

6.3 Spatial modeling - the weak spring

In the biological spatial models which are presented in this thesis, cell objects grow, divide, and push one another or other tissues. Such spatial interaction is modeled by a weak (breakable) spring potential function, as described in [120]. The spring potential is 'breakable' since the attraction force is broken after some distance between the cells. Equation 6.7 describes a weak spring potential where k is the interaction strength, Δ is the Euclidean distance between the two objects, Δ_{opt} is the optimal distance, and Δ_{max} is the maximum interaction distance.

$$V(\Delta, \Delta_{\text{opt}}) = \frac{1}{2}k(\text{Min}(\Delta, \Delta_{\text{max}}) - \Delta_{\text{opt}})^2 \tag{6.7}$$

Spatial objects attempt to minimize the potential thus each object moves along the

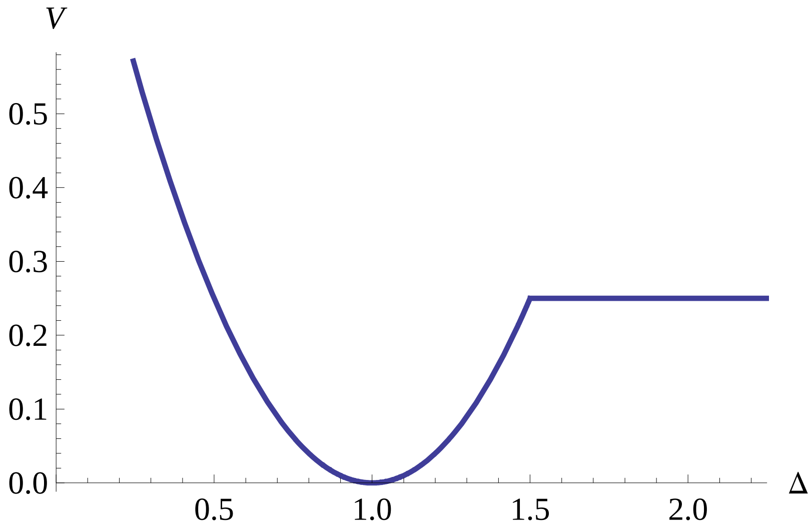


Figure 6.1: Plot of the spring potential according to Equation 6.7 for $k = 1$, $\Delta_{\text{max}} = 1.5$ and $\Delta_{\text{opt}} = 1$

spring's gradient. The following rule describes a spring potential between two cells where the optimal distance is their total radii.

$$\{\text{c1} = \text{Cell}[x_1, r_1], \text{c2} = \text{Cell}[x_2, r_2]\} \rightarrow \{\text{c1}, \text{c2}\}$$
$$\textbf{solving} \ \ \{\tfrac{dx_1}{dt} = -\nabla V(d(x_1, x_2), r_1 + r_2)\}$$

Another possible potential function that may represent the breakable spring is the Lennard-Jones function for the interaction of two atoms [76].

The breakable springs may be generalized to stochastic breakable springs that disconnect according to a discrete stochastic event as the following rules:

$$\{\text{c1} = \text{Cell}[x_1, r_1], \text{c2} = \text{Cell}[x_2, r_2], \text{spring}[\text{c1}, \text{c2}]\} \rightarrow \{\text{c1}, \text{c2}, \text{s12}\}$$
$$\textbf{solving} \ \ \{\tfrac{dx_1}{dt} = -\nabla V(d(x_1, x_2), r_1 + r_2)\}$$

(*disconnect spring*)

$$\{\text{c1} = \text{Cell}[x_1, r_1], \text{c2} = \text{Cell}[x_2, r_2], \text{spring}[\text{c1}, \text{c2}]\} \rightarrow \{\text{c1}, \text{c2}\}$$
$$\textbf{with} \quad \rho_d(|x_1 - x_2|)$$

(*connect spring*)

$$\{\text{c1} = \text{Cell}[x_1, r_1], \text{c2} = \text{Cell}[x_2, r_2], \nexists\text{spring}[\text{c1}, \text{c2}]\} \to \{\text{c1}, \text{c2}, \text{s12}\}$$
$$\textbf{with} \quad \rho_c(|x_1 - x_2|)$$

Both the stochastic and deterministic breakable spring rules create matchings between every pair of cells or other spatial objects. For every discrete event, the simulator integrate $O(n^2)$ differential equations or rate functions if there are n spatial objects. However, the number of interacting objects, in spatial models, is only $O(n)$. In order to save computational time, we implemented a spatial data structure in *Plenum* which assigns grammar rules between spatial elements only if they are considered neighbors. The space partitioning data structure is a KD tree [13], [29]. Generating a KD tree requires $O(n \log(n))$ operations whereas inserting or removing an element requires only $O(\log(n))$. Grammar rules apply only for proximate spatial objects (according to a threshold distance). Spatial object types are declared in *Plenum* using the rule '*spatialObject* $\to$ *obj*' that is augmented to the input grammar.

6.4 Reduction to deterministic model

In many applications, the model's parameters are unknown and have to be estimated from observations or according to some desired behavior. Inference in stochastic models and SPGs, which is discussed in Chapters 7 and 8, is computationally intensive and often intractable. Deterministic models such as ODEs are amenable for parameter space exploration. Here, we describe a mechanism that converts an SPG to an approximate ODE model.

The first step is to convert an SPG to an equivalent SPG where the rates express the number of reactions for each unique object tuple as in ODE models. The equivalent grammar is denoted here as the aggregate SPG form. Every element, in the aggregate SPG form, is augmented with a parameter that counts the number of existing objects in the pool for each configuration of the other parameters. The rate functions are multiplied by the combinatorial number of reactions according to the counting parameters. For example, the following is a rule that eliminates two parameterized objects, A and B, and creates a new object C:

$$\{\mathrm{A}[x_1], \mathrm{B}[x_2]\} \to \{\mathrm{C}[q(x_1, x_2)]]\}$$
$$\quad\quad \textbf{with} \;\; f(x_1, x_2)$$

This rule is converted to the following aggregate form:

$$\{\mathrm{A}[x_1, i_1], \mathrm{B}[x_2, i_2], \mathrm{C}[q(x_1, x_2), i_3]\} \to \{\mathrm{A}[x_1, i_1 - 1], \mathrm{B}[x_2, i_2 - 1], \mathrm{C}[q(x_1, x_2), i_3 + 1]\}$$
$$\quad\quad \textbf{with} \;\; i_1 i_2 f(x_1, x_2)$$

In the ODE approximation, the discrete counters of the aggregate form are replaced by continuous quantities. The rate of change of the quantities in a continuous rule is approximated as the stochastic rate times the number of objects created (positive counter) or removed (negative counter). The approximate continuous rule of the aggregate rule above is:

$$\{\mathrm{A}[x_1, i_1], \mathrm{B}[x_2, i_2], \mathrm{C}[q(x_1, x_2), i_3]\} \to \{\mathrm{A}[x_1, i_1], \mathrm{B}[x_2, i_2], \mathrm{C}[q(x_1, x_2), i_3]\}$$
$$\quad\quad \textbf{solving} \;\; \{\tfrac{di_1}{dt} = -i_1 i_2 f(x_1, x_2), \tfrac{di_2}{dt} = -i_1 i_2 f(x_1, x_2), \tfrac{di_3}{dt} = i_1 i_2 f(x_1, x_2)\}$$

The aggregate grammar form maintains one object instance with counter parameter for every set of object copies in the original form. A rule, in the original form, that contains multiple elements that match the same object is mapped to multiple rules in the aggregate form. Each rule in the aggregate form is equivalent to a different possible combination of objects. For example, the following rule may match three equal objects (same parameter values, $x_1 = x_2 = x_3$) or two equal objects and one different object or three different objects:

$$\{\mathrm{A}[x_1], \mathrm{A}[x_2], \mathrm{A}[x_3]\} \rightarrow \{\}$$
$$\textbf{with} \quad f(x_1, x_2, x_3)$$

Mapping this rule to the aggregate grammar form results in the following three rules:

$$\{\mathrm{A}[x_1, i_1]\} \rightarrow \{\mathrm{A}[x_1, i_1 - 3]\}$$
$$\textbf{with} \quad \binom{i_1}{3} f(x_1, x_2, x_2)$$

$$\{\mathrm{A}[x_1, i_1], \mathrm{A}[x_2, i_2]\} \rightarrow \{\mathrm{A}[x_1, i_1 - 1], \mathrm{A}[x_2, i_2 - 2]\}$$
$$\textbf{with} \quad i_1 \binom{i_2}{2} f(x_1, x_2, x_2)$$

$$\{\mathrm{A}[x_1, i_1], \mathrm{A}[x_2, i_2], \mathrm{A}[x_3, i_3]\} \rightarrow \{\mathrm{A}[x_1, i_1, i_1 - 1], \mathrm{A}[x_2, i_2 - 1], \mathrm{A}[x_3, i_3 - 1]\}$$
$$\textbf{with} \quad i_1 i_2 i_3 f(x_1, x_2, x_3)$$

The “Not Exists” quantifier can be mapped to the aggregate form by including the constraint as a factor in the rate function. An example is the following “Not Exists” rule:

$$\{\mathrm{A}[x_1], \nexists \mathrm{B}[x_1]\} \rightarrow \{\mathrm{C}[x_1]]\}$$

with $f(x_1)$ This rule is converted to the following aggregate form rule where Θ is the Heaviside Step function (Equation 3.14):

$$\{\mathrm{A}[x_1, i_1], \mathrm{B}[x_1, i_2], \mathrm{C}[x_1, i_3]\} \rightarrow \{\mathrm{A}[x_1, i_1 - 1], \mathrm{B}[x_1, i_2], \mathrm{C}[x_1, i_3 + 1]\}$$
$$\textbf{with} \quad \Theta(i_2 = 0) i_1 f(x_1)$$

The rules in the aggregate grammar form do not create or remove objects but, instead, modify the counters. Therefore, the initial pool of the aggregate grammar consists of all the possible unique objects that are created in the original grammar. Yet, SPGs define variable structure systems with a possibly unbounded number of unique objects. SPGs with unbounded or even large number of unique objects can not be mapped to equivalent aggregate grammars. The modeler may approximate the behavior of the original grammar by defining the aggregate grammar's initial pool with a finite number of unique objects that are most likely to dominate the system over time.

The conversion rules described in this section can be encoded and applied automatically. However, such an automated conversion tool was not developed, yet, as part of *Plenum*. A similar conversion tool is Cellerator [119], which provides an automatic translation of reactions to ordinary differential equations. Cellerator handles chemical reactions with unparameterized objects but does not support parameterized objects. Subsequently, Cellerator does not include the expanded conversion (see the foregoing discussion) that is required when multiple parameterized terms may match the same or different objects.

6.5 Meta-Grammar rules

Since the SPG framework extends unrestricted grammars, we can design SPGs that perform manipulations on other SPGs. Such meta-grammars include, for example, an SPG that interpreters or reverse all the reaction rules of different SPGs.

Assume that SPG rules can be represented as a tree where the root node is connected to a LHS elements node, RHS elements node, and a rate node. Each rule element is associated with a tree node that is connected to either the LHS or the RHS node. The following simple rule reverses the direction of a reaction that is expressed as a tree, modifies the rate function, and flags it as 'reversed'.

$$\{\mathrm{LHS}[ruleID, lhsRootID], \mathrm{RHS}[ruleID, rhsRootID], \mathrm{Rate}[ruleID, r], \nexists \mathrm{reversed}[ruleID]\} \to$$
$$\{\mathrm{LHS}[ruleID, rhsRootID], \mathrm{RHS}[ruleID, lhsRootID], \mathrm{Rate}[ruleID, k{\cdot}r], \mathrm{reversed}[ruleID]\}\}$$

Chapter 7

Exact inference in context-free SPGs

Previous chapters discussed the generative process of SPGs and DGs. A generative process is useful for prediction of future events. However, in some cases the modeler's task is to infer the past system's states or learn the model's parameters given some set of observations. Chapter 2 discusses the Chomsky hierarchy of generative grammars [24, 25] and their stochastic extensions. Since context-sensitive and unrestricted grammars are generally infeasible for inference tasks, regular and context free grammars (CFG) and their stochastic extensions are commonly used for modeling and inference purposes. This chapter presents inference algorithms for context-free SPGs, i.e. SPGs with only one term on the LHS.

The context-free SPGs inference algorithms are derived from the inference algorithms for stochastic context-free grammars (SCFG). The SCFG inference algorithms are based on the inside-outside scheme [11, 75], which calculates the probability of an observed sequence. The inside-outside scheme is a dynamic programming algorithm which is analogous to the forward-backward algorithm for Hidden Markov Models (HMMs). Inferring the most likely parsing can be done by modifying the inside algorithm. This modification is analogous to the Viterbi algorithm [127] for HMMs. Lastly, the inside-outside algorithm is used as part of an Expectation Maximization (EM) [34] algorithm in order to estimate the grammar's parameters.

Context-free SPGs are similar to SCFGs except that: 1) the semantics includes the time dimension which is either discretized or continuous, and 2) there is no predefined linear order of the objects as in the strings produced by formal grammars.

Section 7.1 provides a background on the inside-outside algorithm and the EM scheme for learning parameters in SCFG. The inference algorithms for context-free SPGs are developed in Section 7.3.

7.1 The inside-outside algorithm

The inside-outside algorithm is used for calculating the probabilities of SCFGs. The inside part calculates the probability that a grammar, G, generates a subsequence, w_{ij} (from terminal i to j), that originated from a nonterminal N_k, i.e. the probabilities:

$$P(w_{ij}|root = N_k, grammar = G) \ \ \forall i, j, k$$

The algorithm uses a dynamic programming scheme where the probabilities of small sequences are cached and used for calculating the probabilities of enclosing sequences. This approach is valid since the grammar is context free, and therefore, the probability of a sequence is independent of the rest of the string, given the sequence's root element.

The inside-outside procedures are defined for SCFGs in Chomsky Normal form, meaning that the rules are of the form $A \rightarrow BC|a$ where a is a terminal symbol. However, the inside-outside procedures could be redefined for general SCFGs. Furthermore, any CFG can be efficiently [62] transformed to an equivalent CFG in Chomsky Normal form. The outline of the inside algorithm is the following [75]:

Inside Algorithm:

Input: String w, $l = length(w)$, *set of nonterminals* N_k, $S = N_0$.
Initialize: For all i, k: $\alpha(k,i,i) = P(N_k \rightarrow w_i)$
Iteration: For $\delta = 1$ *to* l, *For* $i = 1$ *to* $l - \delta + 1$, *For all* k

$$\alpha(k,i,i+\delta) = \sum_{j=i}^{i+\delta-1} \sum_{m,n} \alpha(m,i,j)\alpha(n,j+1,i+\delta)P(N_k \rightarrow N_m N_n)$$

Output: $\alpha(k,i,j) = P(w_{ij}|root = N_k, G)$

The probability of the whole string given the grammar is: $P(w|G) = \alpha(0,1,l)$. The most likely parse tree can be inferred by a modification of the inside algorithm such that all the summation operators are replaced by maximization operators. In addition, for each $\alpha(k,i,j)$ we store the rule with the maximal probability in order to traceback in top-down order the most likely parsing tree.

The outside procedure calculates the probability of a nonterminal as the root of some subsequence and the string outside of the subsequence. Let w_{ij}^{-1} be the complete string except the sequence w_{ij} (the outside string). The outside probability can be stated as:

$$P(w_{ij}^{-1}, root = N_k|G)$$

The outside probability $\beta(k,i,j)$ is calculated by the following algorithm [75]:

Outside Algorithm:

Input: String w, $l = length(w)$, set of nonterminals N_k, $S = N_0$, α functions.

Initialize: $\beta(0, 1, l) = 1$; *For all* $k \neq 0$, $\beta(k, 1, l) = 0$

Iteration: For $\delta = l$ *to* 1, *For* $i = 1$ *to* $l - \delta + 1$, *For all* k

$$\beta(k, i, i+\delta) = \sum_{j=i}^{i+\delta-1} \sum_{m,n} \alpha(n, i, j)\beta(m, j+1, i+\delta)P(N_m \to N_k N_n)$$
$$+ \sum_{j=i}^{i+\delta-1} \sum_{m,n} \beta(m, i, j)\alpha(n, j+1, i+\delta)P(N_m \to N_n N_k)$$

Output: $\beta(k, i, j) = P(w_{ij}^{-1}, root = N_k|G)$

The probability of a nonterminal as the root of some subsequence given the entire string is:

$$\begin{aligned} P(root = N_k|w_{ij}, w_{ij}^{-1}, G) &= P(w_{ij}|root = N_k, G)P(w_{ij}^{-1}, root = N_k|G)/P(w) \\ &= \alpha(k, i, j)\beta(k, i, j)/P(w) \end{aligned} \tag{7.1}$$

Here, $P(w)$ can be regarded as a normalization factor.

If L is the number of terminals and M is the number of nonterminals, the time complexity of the inside-outside algorithm is $O(L^3M^3)$, and the space complexity is $O(L^2M)$ [11, 75].

7.2 Parameter training in SCFGs

In case the generative process of the data is known, the rules' probabilities are learned according to a maximum likelihood estimation. However, in most cases the generative process of the observed data is not available. The Expectation Maximization (EM) [34] algorithm is used for maximizing the log-likelihood, L, of the latent parameters when the generative process is unknown. The following equation describes the objective of an EM iteration. θ^i denotes the set of latent parameters, x is the observed data (set of strings), and z is a set of hidden variables that fully describes the unknown generative process.

$$\theta^{i+1} = \arg\max_{\theta} \sum_{z} P(z|x, \theta^{i}) L(x, z|\theta) \tag{7.2}$$

Each iteration estimates the latent parameters that maximize the expected log-likelihood with respect to the conditional probability of the hidden variables. The EM algorithm is guaranteed to converge to some local maximum [34] which is dependent on the initial parameters values.

In SCFGs, the generative process is described by the parse tree, i.e. the set of rule executions. The conditional probabilities of the hidden variables, $P(z|x, \theta^{i})$, are the following two equations: the probability that a rule is the root of a sequence, w_{ij} [75]:

$$P(root = N_k \rightarrow N_m N_n | w_{ij}, w_{ij}^{-1}) = \sum_{o} \alpha(k, i, o)\alpha(k, o, j)\beta(k, i, j) P(N_m N_n | N_k)/P(w) \tag{7.3}$$

Also, the probability of a rule that generates a terminal is the following [75]:

$$P(N_k \rightarrow w_i | w_i, w_i^{-1}) = \beta(k, i, i) P(w_i | N_k)/P(w) \tag{7.4}$$

Let P_t denote the probability estimation at iteration t. The maximization of Equation 7.2 results in this reestimation of a rule probability [75]:

$$P_{t+1}(N_m N_n | N_k) = \frac{\sum_{ij|i<j} \sum_{o} \alpha(k, i, o)\alpha(k, o, j)\beta(k, i, j) P_t(N_m N_n | N_k)}{\sum_{ij|i<j} \alpha(k, i, j)\beta(k, i, j)} \tag{7.5}$$

Likewise the probability of a terminal generating rule is re-estimated as follows [75]:

$$P_{t+1}(a|N_k) = \frac{\sum_{i|w_i=a} \beta(k,i,i) P_t(a|N_k)}{\sum_{ij|i<j} \alpha(k,i,j)\beta(k,i,j)} \quad . \tag{7.6}$$

7.3 Inference algorithm for context-free SPGs

Context-free SPGs differ from SCFG in two major ways: 1) the SPG rule firing probability is time dependent, and 2) there is no imposed linear ordering of the terminal objects. For now, suppose that there is an imposed linear ordering on the objects generated by an SPG. The discussion about handling the unordered sets of objects and its complexity is deferred to Section 7.4.

We assume that all or some of the observations (the terminals) are associated with known time points and the task is to infer probabilities of parsing trees or the most likely parsing tree. The following Inside Algorithm is modified for SPG inference. The same modifications apply for the outside algorithm.

SPG-Inside Algorithm:

Input: *Set of terminals* w, $l = length(w)$, $\hat{t}(i)$*- observed time of terminal object* w_i

$\hat{t}$*- max observed time*

$\rho(k,m,n)$*- rate of rule* $N_k \to \{N_m, N_n\}$

$\rho(k,a)$*- rate of rule* $N_k \to a$

$\rho(k)$*- total rates of rules for object* N_k

Initialize: *For all* i,k: $\alpha(k,i,i,t) = if\ \ t < \hat{t}(i) : \ \rho(k,w_i)\exp(-\rho(k)(\hat{t}(i)-t)), \ else : \ 0$

Iteration: *For* $\delta = 1 \ to \ l$, *For* $i = 1 \ to \ l-\delta+1$, *For all* k

$\alpha(k,i,i+\delta,t) =$

$\sum_{j=i}^{i+\delta-1} \sum_{m,n} \int_t^{\hat{t}} \alpha(m,i,j,\bar{t})\alpha(n,j+1,i+\delta,\bar{t})\rho(k,m,n)\exp(-\rho(k)(\bar{t}-t))d\bar{t}$

In many cases, numerical integration is the only option for the foregoing integral since analytical solution is infeasible. A numerical method for estimating the $\alpha(k,i,j,t)$ function over the t axis, requires multiple evaluations over different t values. However, there is no need to compute the complete integral from t to $\hat{t}$ in every $\alpha(k,i,j,t)$ evaluation. Previous integral calculations can be stored and used in subsequent $\alpha(k,i,j,t')$ evaluations.

7.4 Handling SPGs' unordered objects

As opposed to SCFG, SPGs do not generate objects in a predefined linear order. Therefore, the previous SPG-Inside Algorithm has to take into account all possible objects groups.

The following 'Build parse tree' routine constructs a dataset of all the relevant object sets. The routine maintains a list of subtrees that are represented by triplets $(k, \eta, Size(\eta))$ where k is the index of a nonterminal symbol, and η is the set of all terminal nodes in the subtree. The routine builds the data structure bottom-up by constructing subtrees of size one up to a full tree. For each size, s, the routine finds new subtrees by combining two subtrees that have a unifying rule and a total size s.

Build parse tree Algorithm:

Input: set of terminals w, $l = length(w)$, set of nonterminals N_k, $S = N_0$.

Initialize: $Q = \{\}$; For all i, k if there is a rule $N_k \to w_i$ then add $(k, w_i, 1)$ to the set Q.

Iteration: For $s = 2$ to l, For $s_1 = 1$ to $s - 1$, For all k, m, n

retrieve $(m, \eta_1, s_1) \in Q$ and $(n, \eta_2, s - s_1) \in Q$

if there is a rule $N_k \to N_m N_n$ and $\eta_1 \cap \eta_2 = \varnothing$

then add $(k, \eta_1 \cup \eta_2, s)$ to Q.

Once the list of all the relevant object sets is established, the inside and outside routines can use it. The number of unordered sets can be exponentially large, depending on the grammar. Consider a simple grammar with recursive rules such as $A \rightarrow AA|a$. For this grammar, the number of sets with m terminals is $\binom{n}{m}$, and each set can be composed of 2^m different pairs of sets. Using the inside-outside method for such SPGs will be infeasible. Yet, SPGs with a more constrained structure and no recursive rules may be used efficiently. In order to reduce the size of the tuples list, the Build Algorithm may save only tuples that were formed due to grounded rules with a probability rate over some threshold.

7.5 Discussion

The SPG Inside-Outside algorithm can be modified for inference in context-free dynamical grammars that describe memory dependent processes (non-homogeneous processes). The calculation of the α functions is changed to this form:

$$\alpha(k, i, i+\delta, t) = \\ \sum_{j=i}^{i+\delta-1} \sum_{m,n} \int_t^{\hat{t}} \alpha(m, i, j, \bar{t})\alpha(n, j+1, i+\delta, \bar{t})\rho(k, m, n, \bar{t}-t) \exp(-\int_0^{\bar{t}-t} \rho(k, \tau)d\tau)d\bar{t} \tag{7.7}$$

Note the dependence of the rate functions over time. The additional integral for the non-homogeneous exponential process could be computed once in advance over the entire time range. Since the factors that depend on t can not be extracted from the outer integral, calculation of the outer integral cannot be used for evaluations of α at other t values.

The inference algorithm is designed for context-free SPGs that may generate a finite number of nonterminals (unique objects). Yet, some SPGs may generate an infinite number of unique objects because of rules that have output parameters, i.e. parameters of RHS objects which are not given as input on the LHS of the rule. For example, the following grammar rule generates a new item near the cluster location:

$$\text{cluster}[x] \to \{\text{cluster}[x], \text{item}[y]\}$$
$$\quad \textbf{with} \quad (x-y)^2$$

According to this rule, there is a continuous range of cluster objects (each with a different location x) that have a positive reaction rate. A possible solution would be to generate a fixed number of nonterminal cluster objects given an observed item object. The input parameters, which are the locations of the cluster objects in the above example, can be optimized or sampled according to the conditional probability of the rate function.

Chapter 8

Approximate inference in SPGs

Chapter 7 presents an inference algorithm for context-free SPGs. The algorithm, which is based on the inside-outside method [11, 75], utilizes the tree structure dependencies between objects in a context-free grammar. However, when there are context-sensitive rules, which are rules that have more than a single term on the LHS, the inside-outside inference algorithm does not apply. Moreover, every possible object in some parse tree is represented as a node in the inside-outside scheme. This limits the applicability of the inside-outside scheme for SPGs that generate limited number and size of parse trees (multi-reactions paths).

Here, we present an approximate inference algorithm for unrestricted SPGs (having no context-free constraint) that is based on a Markov Chain Monte Carlo (MCMC) sampling method. This inference algorithm uses the simulation algorithm to evaluate multi-reactions path probabilities, and therefore, avoids the explicit parse trees structure that was required for the exact method of the previous chapter. The approximate inference algorithm was published in [137].

Parameter inference may be achieved by evaluating the likelihood function $P(\Theta|\Gamma)$, where Γ denotes the stochastic grammar and its parameters, and Θ denotes the observations or evidence. Prior information about the model's parameters may be incorporated, using Bayes' theorem. The posterior distribution is:

$$P(\Gamma|\Theta) = \frac{P(\Theta|\Gamma)P(\Gamma)}{P(\Theta)} = \alpha P(\Theta|\Gamma)P(\Gamma) \tag{8.1}$$

The marginal probability, $P(\Theta)$, is equivalent to a normalizing factor which will be discarded in our sampling algorithm (the factor is canceled out in the MCMC Metropolis-Hastings method, see next section). The evidence, Θ, may be in the form of full or partial observation of the underlying system state in certain time points. The likelihood function is decomposed according to the multi-reactions path and the observations probability (the error term):

$$P(\Theta|\Gamma) = \sum_r P(\Theta|a[r,\vec{\tau}])P(r|\Gamma) \tag{8.2}$$

where $\vec{\tau}$ denotes the observations time points, r denotes a multi-reaction path, and $a[r,\vec{\tau}]$ denotes the system state at the observation time points as realized by the path r.

Usually, deriving an analytical solution for the likelihood function is not possible. We resort to a sampling algorithm that is based on the MCMC principle.

8.1 MCMC algorithm

An MCMC algorithm [7] generates a sequence of samples, according to a Markov chain transition function that converges to the desired (target) probability. The Markov chain transition function, $M(x|x')$, is designed so that its invariant probability is the target probability, $\pi(x)$:

$$\pi(x) = \int M(x|x')\pi(x')dx'$$

Furthermore, the Markov chain must be ergodic, i.e. it must converge to the invariant probability regardless of the initial condition [81]. The chain satisfies the ergodic condition if it is both irreducible and aperiodic. A chain is irreducible when there is a non-zero probability of reaching every state from any other state. A reducible chain has more than one invariant probability. An aperiodic (with period of 1) chain is established when, for every state, there is a non-zero probability to stay.

A sufficient but not necessary condition ([81],[7]) for $\pi(x)$ to be the invariant probability of the chain is the following "detailed balance condition":

$$\pi(x)M(x'|x) = \pi(x')M(x|x')$$

For our sampling task, we use the Metropolis-Hastings (MH) algorithm [61] which is an MCMC method. An MH step works as follows:

1) Generate a candidate sample x given the current sample, $x^* = x^{(i-1)}$, according to a proposal distribution $q(x|x^*)$.

2) (To maintain detailed balance) The candidate sample is accepted according to the probability:

$$\text{Min}[1, \frac{\pi(x)q(x^*|x)}{\pi(x^*)q(x|x^*)}]$$

If accepted $x^{(i)} = x$, otherwise $x^{(i)} = x^*$.

The choice of the proposal distribution, q, is important for the convergence of the algorithm. On the one hand, a proposal distribution with small variance might constrain the MH method to a local region of the target distribution space. On the other hand, a proposal distribution with high variance will result in many rejections, which subsequently increase the correlation between samples. As a rule of thumb, the proposal distribution should be adjusted so that the rejection rate is approximately 0.5 [72] [81].

In most parameter inference applications, we are interested in the global maximum (or maxima) of the target distribution. For this task, a simulated annealing like strategy [48] can be integrated in the MH algorithm. The target distribution is modified to $\pi^{1/T_i}(x)$ where T_i obeys a decreasing temperature cooling schedule $\lim_{i \to \infty} T_i = 0$. The algorithm is expected to initially explore a broad region of the state space and gradually to confine the search to lower-energy areas.

8.2 Grammar sampling algorithm

We define the invariant (target) distribution of the MH method as the posterior distribution in Equation 8.1. A sample is an instantiation of the grammar parameters and a multi-reaction path. A possible choice for the proposal distribution can be obtained by omitting the observations probabilities from the target distribution:

$$q = P(r|\Gamma)P(\Gamma)$$

This proposal distribution may exhibit high variance since the candidate sample is independent of the current sample which may lead to high number of rejections. Since the grammar parameters are a fixed size set, a conditional probability on the current parameters values may be easily included in q:

$$q = P(r|\Gamma)f(\Gamma|\Gamma^*) \tag{8.3}$$

A Normal distribution which is centered around the current parameters values, Γ^*, is a natural choice for continuous variables.

Still, the multi-reaction path r is independent of the current path. We offer a modified proposal distribution for which each candidate path has the same set of reactions as the current path except for a randomly chosen time window (t', t''). The reactions in the chosen time window are sampled according to the SPG probability conditioned on the state at time t'. In other words, the SSA algorithm is executed over the subinterval (t', t'') where the initial state is the state reached by the current path at time t', $a(r^*, t')$. The probability of the candidate path in the subinterval window (denoted as $r'[t', t'']$) is:

$$P(r'[t', t'']|\text{initial time} = t', \text{end time} = t'', \text{initial state} = a(r^*, t'), \text{parameters} = \Gamma) \tag{8.4}$$

The candidate multi-reaction path over the whole time interval, r, is defined as follows (where + denotes concatenation of multi-reaction paths):

$$r = r^*[0, t'] + r'[t', t''] + r^*[t'', t_{\text{end}}]$$

Note that although the reactions in the last subinterval (t'', t_{end}) are identical to the current path, r^*, the states may be different, due to the changes in the intermediate subinterval. The deterministic scheme for setting the reactions in the last subinterval may lead to improbable paths. For example, the reactions in $r'[t', t'']$ could remove all water molecules by decomposition ($2H_2O \longrightarrow 2H_2 + O_2$) even though there is another decomposition reaction that occurs in $r^*[t'', t_{\text{end}}]$ before the production of any other water molecules. However, the target probability (π) of such an improbable path will be zero and it will be rejected.

Now, the modified proposal distribution is defined as:

$$q(r'; t', t'') = P(r'[t', t'']|t', t'', a(r^*, t'), \Gamma)P(t', t'')f(\Gamma|\Gamma^*) \tag{8.5}$$

Note that for a proper MH proposal distribution, the time window (t', t'') should be integrated out. However, the MCMC method would preserve detailed balance even when picking a random time window (according to $P(t', t'')$) instead of integrating over them. This construction of an MCMC sampler is known as a mixture of transitions [81]. In general, an MH with mixture of transitions can be applied in order to modify each variable, or block of variables, separately. In that case, a variable or block of variables is randomly selected (usually according to uniform distribution), followed by an MH transition over the chosen variable space. Such an algorithm is applicable when the proposal distribution can be decomposed according to individual transitions. The current proposal distribution (Equation 8.5) can be decomposed to a transition over a random window in the multi-reaction path and a transition over the grammar parameters (or individual parameters):

$$q_1(r'; t', t'') = P(r'[t', t'']|t', t'', a(r^*, t'), \Gamma^*)P(t', t'') \qquad q_2(\Gamma|\Gamma^*) = f(\Gamma|\Gamma^*) \tag{8.6}$$

The proposal distributions (q_1,q_2) should be designed according to the Markov chain ergodicity requirement. If q_2 is a Normal distribution which is centered in the current parameter value, then the chain is aperiodic and irreducible (since each parameter value is reachable). The window proposal is aperiodic since there is a non-zero probability to stay in the same path, i.e. $q_1(r^*; t', t'') > 0$. If the set of possible windows, according to $P(t', t'')$, covers the whole time interval, then the proposal distribution is irreducible (since each path is reachable from any other path).

8.3 Illustrative example

As an illustrative example of the inference algorithm, we use a simple model of chemical reactions for synthesis and decomposition $\mathrm{A} + B \underset{k_d}{\overset{k_s}{\rightleftharpoons}} C$, which can be expressed in SPG syntax:

Chemical-Reaction= **grammar**[

$\{A, B\} \rightarrow \ C$ **with** k_s

$C \rightarrow \{A, B\}$ **with** k_d

]

After setting the rate variables, we generated multi-reaction paths using an SPG interpreter and simulator [136]. Figure 8.1 plots the time sequence of molecule states for a single stochastic multi-reaction path.

The input for the inference algorithm is a sequence of molecule-number states in 20 evenly spaced time points. The observations conditional probability (data error term) has a Normal distribution shape with an input width (variance) parameter σ_i. Figures 8.2 and 8.3 present five trajectories of the MCMC inference algorithm for a single observations sequence. Figure 8.2 shows the convergence of the rate parameter

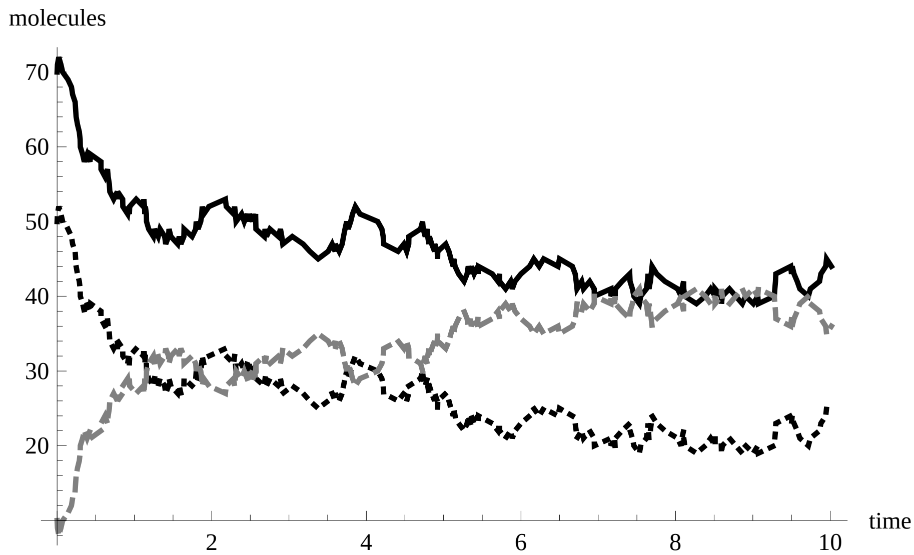


Figure 8.1: Plots of the time sequence of molecule states for a single stochastic multi-reaction path. Y axis - molecule quantity (A-solid black line,B-dotted black line, and C-dashed gray line). X axis - time. The synthesis and decomposition rates are $k_s = 0.01$,$k_d = 0.3$

where the true parameter value is 0.3. Figure 8.3 shows the convergence of the multi-reaction path in each MCMC iteration to the observed molecules states.

The inference algorithm uses different transitions for the rate variable and for the multi-reaction path, as defined in Equation 8.6. This has the benefit of potentially higher acceptance probability. The variance of the parameters proposal distribution and window size distribution were adjusted during the execution, in order to maintain a low rejection rate (below 0.5).

The algorithm implements an exponentially decreasing cooling schedule ($T_i = pT_{i-1}$ where $0 < p < 1$) for the rate parameter transition. Initial attempts without a cooling schedule have resulted in rapid convergence to low-probability samples.

We have performed inference for different values of the decomposition rate k_d. The results are shown in Figure 8.4. For each rate value, five random multi-reaction paths were generated. We performed 25 inference runs where each is given one of the

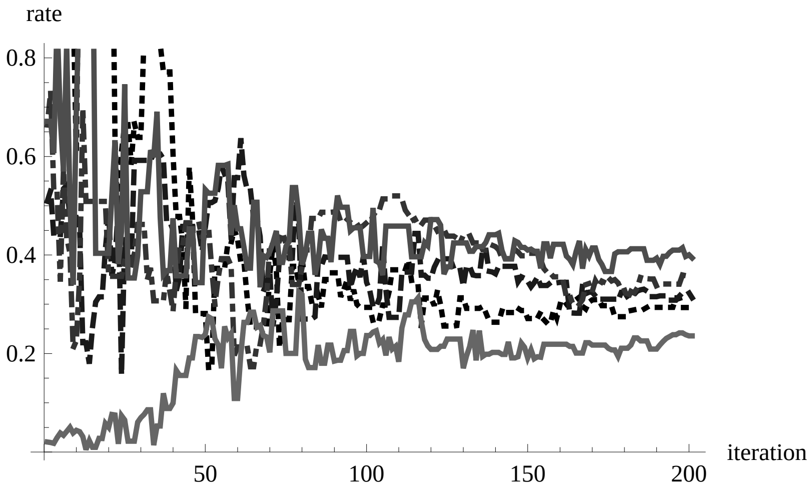


Figure 8.2: Trajectories of the MCMC inference algorithm: Y axis - inferred decomposition rate k_d. X Axis - iterations counter. The inferred decomposition rate converges toward the target rate value, $k_d = 0.3$

generated multi-reaction paths and a random initial rate value.

The variance of the observations conditional probability is important for the accuracy of the inference method. A search algorithm is more susceptible to getting trapped in local minima when the variance is too low. High variance may lead to the opposite effect since the penalty for improbable samples is insufficient. Figure 8.5 displays the increase in standard deviation of inferred rates as we increase observations' probability variance.

8.4 Related work

Numerous optimization approaches have been applied to parameter inference in deterministic dynamical systems. Among the first fully automatic parameter estimations for biological network dynamics was [110], which used the Lam-Delosme variant of simulated annealing to fit the 33 parameters, including a 5x5 matrix of connection strengths in a phenomenological gene regulation network (Artificial neural network

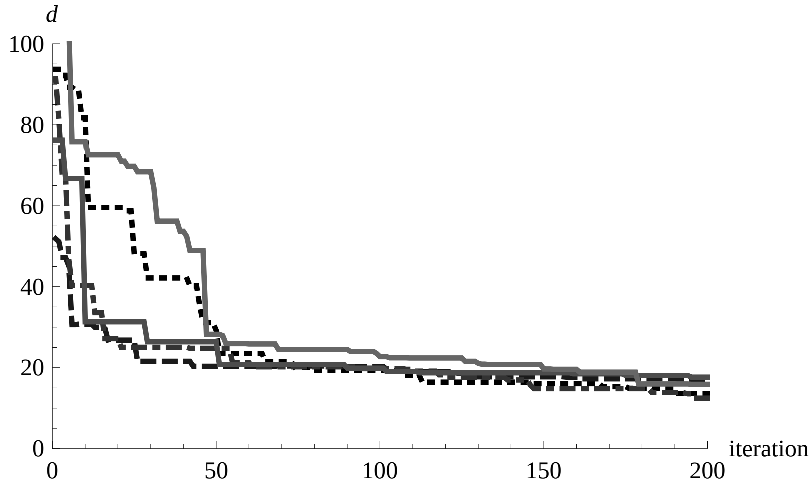


Figure 8.3: Trajectories of the MCMC inference algorithm: Y axis - Euclidean distance between the current sample's molecule state and the input. X Axis - iterations counter.

or ANN style GRN) model to image-derived expression domain data in Drosophila development. This is still a cutting-edge problem today, 15 years after the original computer experiments. A similar algorithm was later parallelized [26]. Genetic algorithms were compared to simulated annealing on these GRN models in [85]. Furthermore, [85] presents the results of applying these optimization methods to variable-structure dynamical systems, specified by a deterministic grammar as defined in [92]. The resulting model maintains a changing number of cells each of which contains a set of chemical kinetic differential equations representing a gene network. Simulated annealing, quasi-Newton methods, and a variety of other deterministic and stochastic optimization methods (including Levenberg-Marquardt, Nelder-Mead, and Hooke-Jeeves) were compared in their applications to more standard chemical kinetic differential equation models in [86]. Lam-Delosme simulated annealing compared favorably with several alternative methods for optimizing signal transduction models in [139].

There has been a recent revival of interest in developing parameter optimization

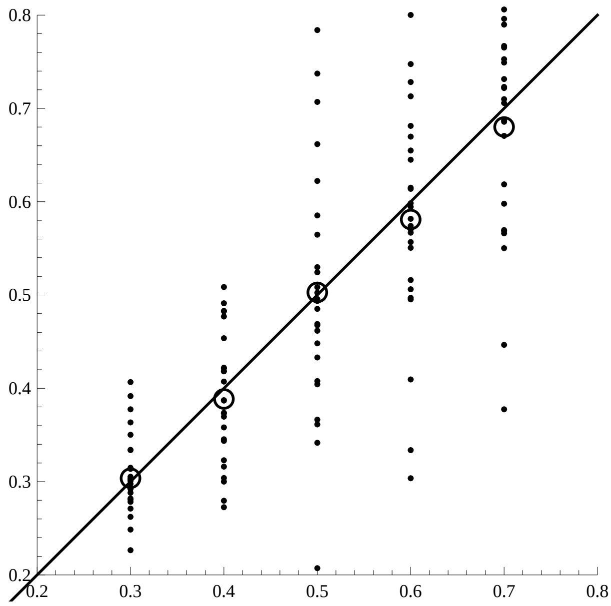


Figure 8.4: The distribution of inferred rate values for each target value, $k_d = \{0.3, 0.4, 0.5, 0.6, 0.7\}$: Each column of dots represents the 25 inference outcomes per target value (x position). The average inferred rate for each value is denoted by the circles.

methods. For example [68] applied parallelized differential evolution to the ANN GRN models. [108] applied the adjoint method, a continuous-time method analogous to backpropagation through time in discrete-time neural networks, to efficiently compute gradients for gradient algorithms and applied them to mass action kinetic models of Drosophila flies' wing development. [97] applied recent developments in evolutionary strategies to ANN GRN's. Another major category of research looks to Bayesian inference for a firm statistical foundation in structure and parameter estimation. An essential requirement is a practical prior distribution on structures and/or parameters. [129] demonstrates a GRN inference method for feed-forward transcriptional networks using an approximate but tractable graph prior. Other graph priors are possible, as in [14], [107].

Turning to parameter estimation in stochastic dynamics, [56] developed a time-subdivision approach to learning reaction rates in small reaction networks under an

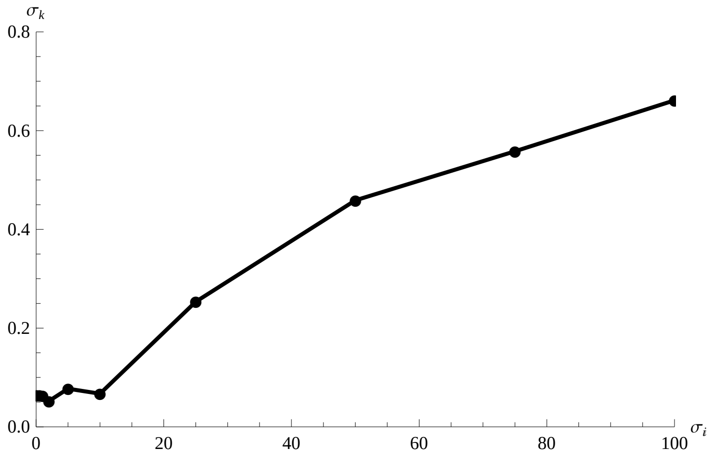


Figure 8.5: Standard deviation of inferred rates, σ_k, as a function of the observations' probability variance, σ_i. The synthesis and decomposition rates are $k_s = 0.01$, $k_d = 0.3$

SDE approximation to the usual stochastic dynamics. A statistical mechanics perspective is applied in [42]. Others could be cited e.g. from recent workshops [1], [2]. All currently have strong restrictions on their domain of applicability, especially in the size of the inferable system.

Also for parameter inference in stochastic dynamical systems, Green [59] has introduced a sampling method for probability distributions that are defined over spaces of varying dimensionality such as the variable-structure systems considered above. The method, known as reversible jump Markov Chain Monte Carlo (MCMC), is based on the Metropolis-Hastings (MH) [61] scheme (which will be described in a later section). The algorithm performs "jumps" between models (or states) of different dimensionality. Therefore, it is suitable to handle transitions between reaction paths of different length (hence, different number of variables). The reversible jump method requires an application-based design of jump transitions. The jump transitions that are commonly defined are birth, death, split, or merge of random variables.

The reversible jump method was generalized to a block updating method which was used for inference in stochastic reaction models [16]. The block update scheme modifies multiple variables, which represent the number of reactions (of each type) and the reactions times in a single MCMC step. A previous work on stochastic epidemic compartment models [51] has also used multiple jump events in each iteration in order to increase the convergence rate of an MCMC sampler. A Simulated Annealing cooling schedule was integrated into the reversible jump method in [6].

Reinker et al. [111] have taken an approximate maximum likelihood approach for parameter estimation in stochastic biochemical reactions. The method approximates the number of reactions in each interval and assumes that the total rate does not change inside an interval. A similar approach was taken for simulating chemical reactions by the approximate tau-leap method [53].

An algorithm for inferring grammar-based structure models was introduced in [117]. The grammars, which are based on an L-system formulation [106], describe recursive structures and are used to model images of multi-cellular bacteria. Their grammars resemble SPGs but are only context free (only a single element on the LHS of rule) and cannot model continuous-time processes. The inference procedure is an MCMC method that includes reversible jumps for adding or removing cells and branches from the multicellular structure.

8.5 Discussion

The method presented here does not require the construction of specialized jump transitions, which was required in [59], [16], [51], [6] and [111]. The modeling assumptions are all integrated in the declared grammar. The methods described in [16] and [51] require exact molecule counts over discrete time points, whereas in our method, the data is assumed to be noisy and the multi-reaction paths are weighted accordingly. Since the penalty for matching the data is less strict (and tunable), this method can be more flexible in traversing the multi-reaction path space.

As described before, the simulation algorithm for DG is an extension of the SPG simulation algorithm and is derived directly from the Time-Ordered Product Expansion (TOPE). The presented inference algorithm is based on the simulation from the TOPE and therefore can encompass even DG models.

The current inference method is most effective for SPGs that have a finite number of unique objects which are all present throughout the simulations. Using forward simulations to generate new multi-reactions paths might be ineffective for SPGs that produce numerous unique objects, e.g. SPGs that include output parameters. In these grammars, there is a small probability of generating a multi-reactions path, by forward simulation that includes all the unique objects of the observations. A possible future direction is to incorporate, in this approximate method, a reverse approach for generating reactions from the observations, as in the inside-outside algorithm.

Another possible direction is to incorporate exact marginalization over some parts of the multi-reactions path. A method that combines Monte-Carlo sampling and marginalization is known as Rao-Blackwellisation [22]. The Rao-Blackwellised particle filtering algorithm for Dynamic Bayesian Networks was presented in [38]. Exact marginalization is feasible only for some context-free SPGs, as was discussed in Chapter 7. Therefore, the marginalization part should be confined to context-free and independent parts of the generative process. In a Rao-Blackwellised sampling algorithm for SPGs, the sampling part generates a multi-reactions path over some internal (or hidden) objects. Given the sampled internal objects and a context-free decomposition of the remaining possible reactions, the exact marginalization may calculate the probability of the observations.

Chapter 9

Galaxy morphology grammar

Galaxies are often classified according to Edwin Hubble's classification scheme that is known colloquially as the Hubble tuning-fork, shown in Figure 9.1. Hubble's galaxy classification distinguishes between elliptical and spiral galaxies, which are further divided into normal spirals and barred spirals. Still, this is a very broad classification scheme that does not capture the apparent variety of shapes and components of galaxies that belong to the same class. In order to gain insight into the formation and evolution of galaxies, we need to understand the structure of galaxies based on rigorous, quantitative criteria. Quantitative properties of galaxies include disk and bulge luminosities and radii, bar lengths, and spiral arm pitch angles (e.g., the tightness of spiral arm winding). These quantities, when measured accurately for nearby galaxies, can give us information on the structure and mass of the halo of "dark matter" around galaxies and insights into the history of how galaxies formed through mergers, since mergers tend to disrupt disks and build bulges.

Recent advancements in telescope technology have produced an inflating size of image datasets of both nearby and distant galaxies. The task of analyzing the images and feature extraction becomes impractical without the aid of machine learning tools.

Spiral galaxies are complex objects which are intrinsically “fuzzy” since the bulge, disk, and bar regions do not have definite boundaries. In addition, images taken by ground-based telescopes are smeared due the atmospheric distortion of rays of light. This smearing effect is known as Point Spread Function (PSF). Spiral arms present a particularly complex problem for automated classification, and no previous scheme of automated morphological measurement has ever successfully found a way to measure spiral arm properties such as pitch angle or arm length. These are of particular importance since dynamical models predict that spiral arm pitch angle is directly related to the degree of mass concentration in galaxies, which is a fundamentally important quantity.

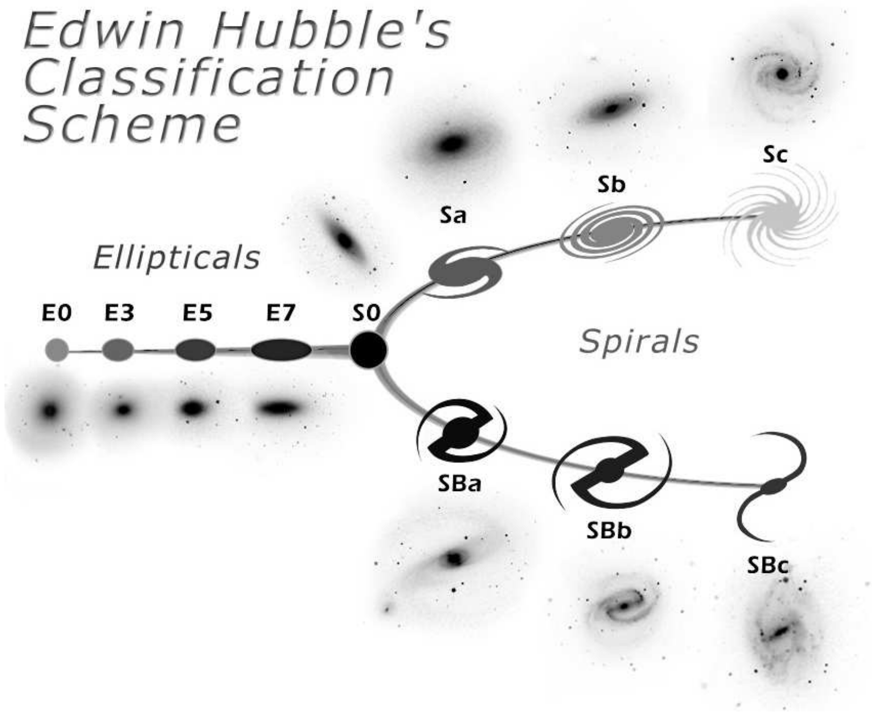


Figure 9.1: Hubble Classification Scheme. Retrieved from http://hubblesite.org/newscenter/archive/releases/1999/34/image/o

Related work on automated galaxy morphology was mostly focused on classification of distant galaxies. An Expectation-Maximization (EM) [34] algorithm for automatic identification of "bent-double" morphology was introduced in [67]. Another regression algorithm that identifies the bulge-to-disk ratio was presented in [5]. However, neither of the papers discusses the identification of individual galaxy components such as spiral arms and bars.

An initial attempt at inferring the spiral structure for galaxy images was done in [112]. The spiral arms and bars were modeled as a collection of segments, where the distribution of an angle between two segments *i* and *i+1* depends on the angle between previous segments *i-1* and *i*. An algorithm for fitting the structural parameters to galaxy images using the Levenberg-Marquardt method for optimization was introduced in [100]. GALFIT supports a detailed parameterization of galaxy structure: however, spiral arms are not modeled directly.

9.1 Spiral galaxy grammar

This section presents a DG model for generating spiral galaxies structures. The model is based on the spiral galaxies' structure with no relation to the current physical theory of galactic spiral arms formation. The objects form a hierarchy of four levels: 1) galaxy object 2) bulge and spiral arms 3) knots and 4) stars. A spiral arm object is constantly growing according to a continuous rule along a spiral trajectory that originates in the center of the galaxy. A discrete/stochastic rule creates the knot objects from each arm, thus generating a dynamic number of knots along the spiral arms. Subsequently, the knot objects create clusters of star objects according to a Normal distribution formulation.

grammar[

*(*creation of major components*)*

spiralGalaxy$[x] \rightarrow \{\text{bulge}[x], \text{armGenerator}[x]\}$ **with** 1

armGenerator$[x] \rightarrow \{\text{armGenerator}[x], \text{arm}[x, \theta_0, 0], \text{arm}[x, \theta_0 + \pi, 0]\}$

with $\rho_g \,\text{Uniform}(\theta_0; 0, \pi)$

*(*create a knot in the bulge*)*

bulge$[x] \rightarrow \{\text{bulge}[x], \text{knot}[x_1]\}$

with $\rho_b * N(x_1; x, \sigma_b)$

*(*extend the spiral arm - arm length (r), angle (θ) , and position (x)*)*

arm$[x, \theta, r] \rightarrow \text{arm}[x, \theta, r]$

solving $\{\frac{dr}{dt} = k, \frac{d\theta}{dt} = s(r), \frac{dx}{dt} = \{\text{Cos}(\theta), \text{Sin}(\theta)\}$

*(*stop the spiral arm*)*

arm$[x, \theta, r] \rightarrow \{\}$

with d_a

*(*create a knot in a spiral arm*)*

arm$[x, \theta, r] \rightarrow \{\text{arm}[x, \theta, r], \text{knot}[x_1]\}$

with $\rho_a * N(x_1; x, \sigma_a)$

*(*create a star in a knot*)*

knot$[x] \rightarrow \{\text{knot}[x], \text{star}[x_1]\}$

with $\rho_k * N(x_1; x, \sigma_k)$

*(*stop the knot*)*

knot$[x] \rightarrow \{\}$ **with** d_k

]

We define the change in spiral arms' angle as:

$$s(r) = \alpha * r^{\beta}$$

In order to add the bar component, the arm angle is held constant for a predefined distance from the galaxy center, denoted as b. This is modeled by a sigmoidal function:

$$s(r) = \sigma(r - b)\alpha * r^{\beta}$$

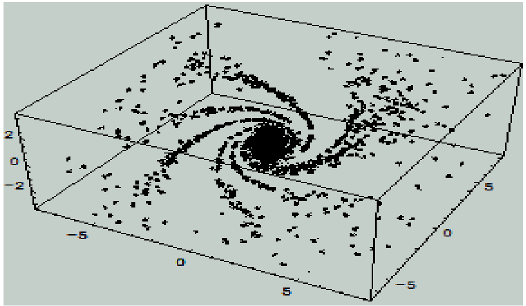
An extension of the current model to 3-Dimensional space is straightforward. Furthermore, the location of stars may be translated according to the viewing angle and the Point Spread Function (PSF). Different results of 3-Dimensional simulations are plotted in Figure 9.2. The simulations differ in the number of generated arms, rate of angle change (α,β), initial arm angle θ_0, and viewing angle.

9.2 Galaxy model inference

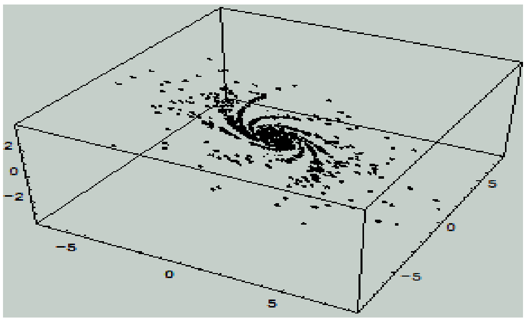
The objective is to infer the model's latent variables from a galaxy image. The latent variables in the model include: the initial angle of each arm θ_0, variables for spiral angle change (α,β), the bar length b, and the rates of creation, ρ, and removal, d, of knots, arms and stars.

Using Bayes theorem (Equation 8.1), we define the inference objective function. The inference scheme of Chapter 8 is applicable here since it is based on deriving the

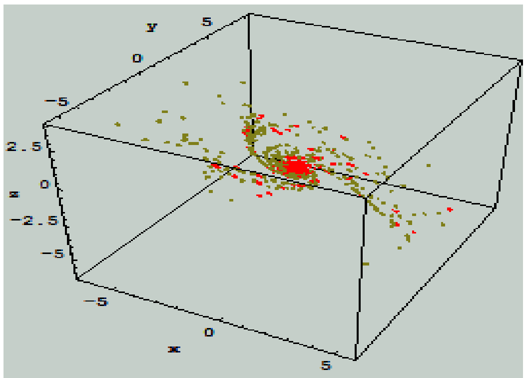
(a)



(b)



(c)



(d)

Figure 9.2: Results of different 3-Dimensional simulations

likelihood function from the TOPE, Equation 4.1. The inference scheme in Chapter 8 involves regenerating a multi-reactions path (or part of the path according to a window) for every MCMC sample. Such sampling algorithm is appropriate for grammar models that do not have random output parameters, as in Section 8.3. However, the galaxy grammar rules include random output parameters for the knots and stars locations and therefore generate a multi-reactions path from the TOPE (likelihood function) that will produce the exact same output objects, is improbable.

Instead of using the exact likelihood function of a galaxy grammar multi-reactions path, we use an approximation. First, a predetermined number of arms is assumed. In addition, the number of knots is predetermined, although their time of creation and location are still unknown. These assumptions simplify the model so that there is a static number of star clusters. Taking into account only the arm-knot creation process, the following equation is the likelihood function:

$$L_{\mathrm{arm}:k} = \left(\prod_{i=1}^{n} e^{-\Delta t_i \hat{\rho}_i} \rho_a * N(w_{\mathrm{knot}_i}; arm(\theta_0, t_i), \sigma_a) \right) * e^{-\Delta t_{n+1} \hat{\rho}_{n+1}} d_a \tag{9.1}$$

$\hat{\rho}_i$ denotes the total rate at time interval i, which is $\rho_a + d_a$ for a single arm process. t_i denotes the time of the ith reaction whereas $\Delta t_i = t_{i+1} - t_i$. $arm(\theta_0, t_i)$ is the solution for the spiral arm position ODE given the initial angle and time, which is based on the optimized variables α, β. The final product results from the ending arm removal event.

Given an input galaxy image, the stars objects are assumed to be represented by the white pixels. Suppose the knot origin of each star is known and is determined by the mapping. knot$[j]$. The following product is the relevant part of the grammar reaction path's likelihood function for the creation of the observed stars:

$$\prod_{j=1}^{m} \rho_k * N(x_j; w_{\text{knot}[j]}, \sigma_k)$$

However, the knot origin of each star is an unknown that should be summed over, as in the following:

$$\prod_{j=1}^{m} \sum_{i=1}^{n} \rho_k * N(x_j; w_i, \sigma_k) \tag{9.2}$$

The summation is over all the combinations of paths for generating the stars from different knots. This summation is based on the assumption of equivalence between the different knots. There is such equivalence only when the life time of each knot is the same. The knot's life time is the time between the knot's creation and removal. When the life time of an arbitrary knot is longer than the rest, there are more multi-reaction paths that create stars from that knot. A varied predefined life time for each knot may be expressed in the model by setting a weight or prior function $P(w_i)$ for each knot, which will get multiplied by the star's conditional (Normal) probability.

Instead of optimizing the likelihood function, we use the log-likelihood since the logarithmic (or any monotone) transformation does not affect the maximum. This is beneficial for optimization since the logarithm simplifies the expression. However, there is no advantage in using the logarithm on the product expression in Equation 9.2 because of the inner summation. To overcome this problem, we use a formulation originated from the Expectation-Maximization (EM) algorithm [34]. The EM algorithm is popularly used for maximum likelihood estimation of mixture of Gaussians models. It maximizes a lower bound of the objective function which is based on

Jensen's Inequality:

$$\text{Log}\left(\sum_{i=1}^{n} N(x_j; w_i, \sigma_k)\right) = \text{Log}\left(\sum_{i=1}^{n} \frac{q_{\text{ji}}}{q_{\text{ji}}} N(x_j; w_i, \sigma_k)\right) \geq \sum_{i=1}^{n} q_{\text{ji}} \text{Log} \frac{N(x_j; w_i, \sigma_k)}{q_{\text{ji}}}$$

whenever $q_{\text{ji}} > 0$ and $\sum_{i=1}^{n} q_{\text{ji}} = 1$.

The variables q_{ji} provide a weight for the relation between data item (star) and cluster (knot). EM iteratively optimizes the q_{ji} variables on the E-step and the model's latent variables on the M-step. The E-step has a simple closed form solution:

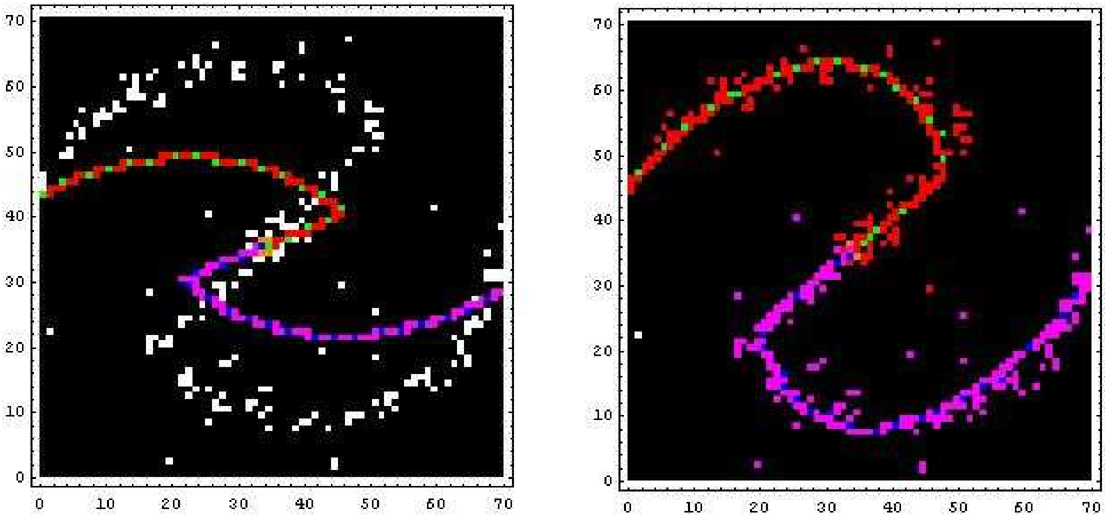
$$q_{\text{ji}} = \frac{N(x_j; w_i, \sigma_k)}{\sum_{i=1}^{n} N(x_j; w_i, \sigma_k)}$$

There is no closed form solution for the latent arm's variables in the M-Step. Therefore, the M-Step is solved by a numerical method.

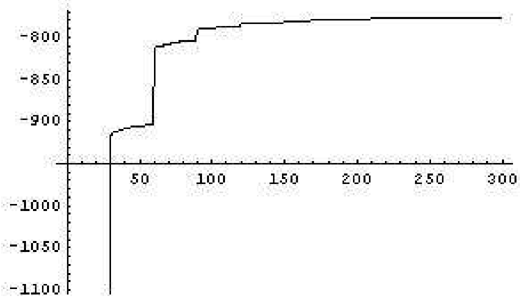
9.3 Results

We performed inference on images of a simulated barred-spiral galaxy. The input stars' locations are shown as white pixels in Figure 9.3 and Figure 9.4. The figures illustrate the initial and final configurations of the knots' locations and the inferred spiral arm parameters. The two figures represent two inference runs from different initial configurations. The knots' locations and the spiral structure variables (α, β, θ_0) converge to the target values. However, the spiral arm length does not change significantly throughout an EM inference run. Figure 9.3(b) and Figure 9.4(b) show the convergence of the log-likelihood during each inference execution.

Figure 9.5 illustrates the initial and final states of the model for a real galaxy (NGC-

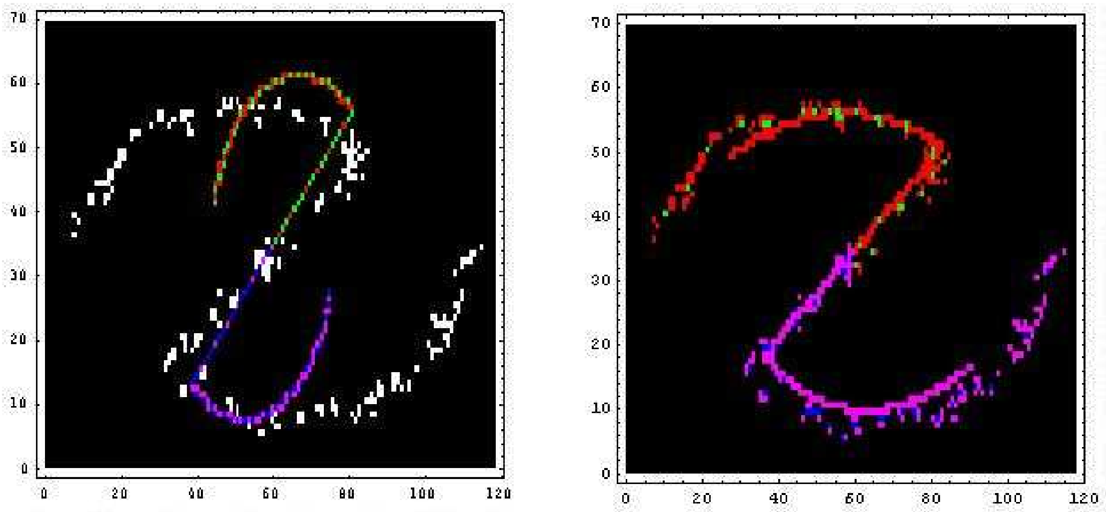


(a)

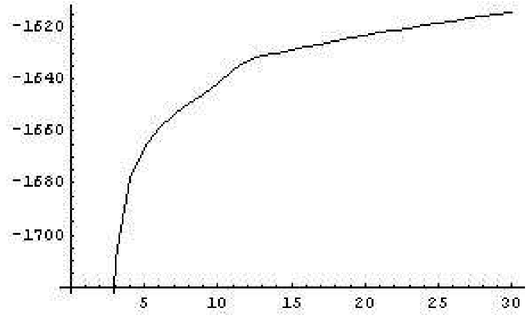


(b)

Figure 9.3: 9.3(a) The input barred-galaxy and the initial and final states of the model. There are 20 knots per arm. 9.3(b) Plot of the convergence of log-likelihood function



(a)



(b)

Figure 9.4: 9.4(a) The input barred-galaxy and the initial and final states of the model. There are 20 knots per arm. 9.4(b) The convergence of log-likelihood function for the data

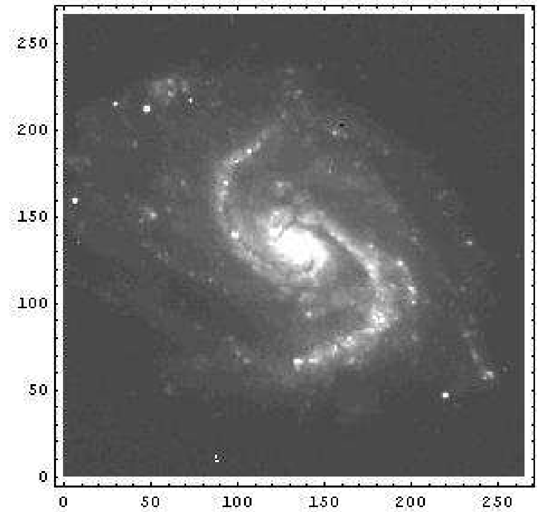
895) image, shown in Figure 9.5(a). In this model, we incorporate an affine transformation matrix in order to capture the angle in which the galaxy is tilted. The transformation matrix is also optimized, as seen by the elliptic shape of the final spiral arms. Still, the optimized shape of the spiral structure does not fully capture the underlying structure.

9.4 Future directions

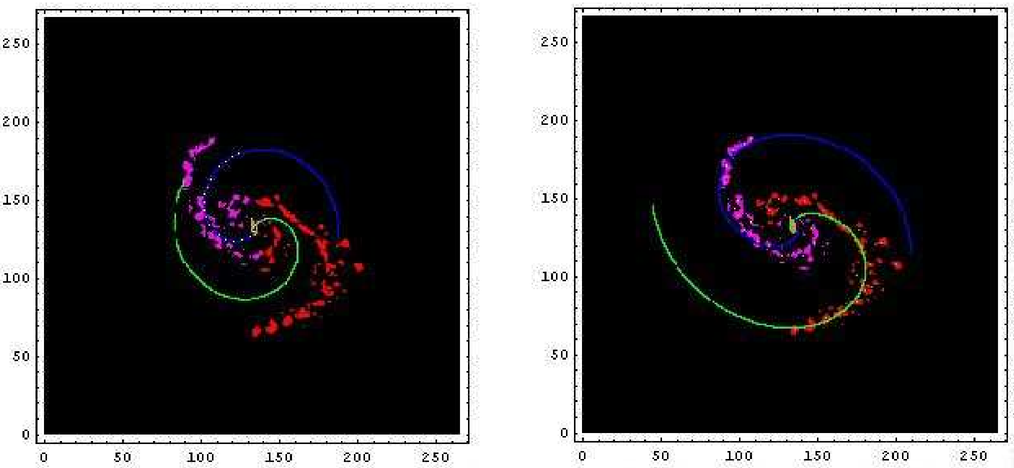
The results for NGC-895 demonstrate the complexity of inferring the correct model from these real galaxies' images. The galaxy's structure is more complex than our simplified model, and the telescopic images are limited in quality due to physical constraints. A possible remedy for the inference difficulties is to allow freedom in the development of spiral structures, similar to the model of [112]. This can be achieved by adding a stochastic rule that randomly changes the spiral arm's direction. However, such added flexibility in the model increases the complexity of the parameter space.

The current algorithm is susceptible to the choice of initial conditions. As with other deterministic optimization methods, the current algorithm gets easily trapped in local minima. A simulated annealing strategy [48], for accepting or rejecting new parameter values, may be incorporated in the M-step, in order to avoid local minima.

In this chapter we discussed only an inference algorithm for a predefined set of knots. An optimization algorithm for an unknown number of knots may be derived using the Metropolis-Hastings (MH) [61] method, as introduced in Chapter 8.



(a)



(b)

Figure 9.5: 9.5(a) Galaxy NGC-895. 9.5(b) The initial and final states of the model for NGC-895. The final configuration does not fully capture the underlying spiral structure, as the initial arm angle is inaccurate.

Chapter 10

Modeling root development

10.1 Introduction

The flow of auxin across plant cells plays an extensive role in regulating patterns of cell growth, divisions and fates in development [125]. This is particularly apparent in the root apical meristem (RAM) where auxin flow is involved in the maintenance of an undifferentiated stem cell population and the continuous formation of regular cell layers on the meristem periphery. The RAM consists of meristematic zone with mitotically active cells and promeristem which contains stem cells. The promeristem is subdivided into a quiescent center (QC) surrounded by stem cells (initials) [36]. Upward and laterally located initials propagate the root radial pattern, a vascular cylinder surrounded by the concentric layers of pericycle, endodermis and cortex [125]. Directly underneath, columella initials produce columella cells. Monitoring of free auxin level in the root by the expression of auxin-responsive DR5 reporters has revealed an auxin maximum in the columella initial cells, with lower activity in the QC and mature columella [114]. *A. Thaliana* root tip structure with the auxin flow diagram is illustrated in Figure 10.1.

Auxin which is secreted from the shoot is transported acropetally in the root via the vascular cylinder toward the root tip [96, 47]. Auxin is transported between plant cells by a combination of diffusion and active transport mediated by influx and efflux carriers [69]. There is a variety of evidence that auxin maxima may be a consequence of actions of the PIN family efflux carriers. PIN1 and PIN4 proteins provide a continuous straight route for auxin along the central apical-basal root axis through the vascular cylinder toward the QC [46], [135].

10.2 Model for Auxin transport

Our proposed model [88] for auxin transport is as follows: at low intracellular concentrations, auxin increases the rate of its own efflux by enhancing the expression of PIN proteins. However, if its concentration exceeds a threshold, then auxin starts to inhibit its efflux by activating PIN protein degradation. There is a direct exchange of auxin between the cells (intercellular space is ignored).

Stochastic grammar reactions describe events such as cell division or death, whereas continuous rules depict cell growth and movement and the active or passive transportation of signaling molecules between cells as well as dissipation of these molecules. In root development simulation, cells are constantly moving along the central longitudinal axis due to growth of adjacent cells. This dynamic behavior is modeled by Dynamical Grammar rules that specify a weak (breakable) spring potential function between neighboring cells, as described in Section 6.3.

A summary of the root Grammar's rules is depicted below. A 'Cell' object is comprised of four parameters: x (location), r (size), m (mode - either growth or wait) and

a (auxin concentration).

(*cell enters idle mode*)

$\text{Cell}[x, r, m = 1, a] \to \text{Cell}[x, r, m = 2, a] \quad \textbf{with} \quad \rho_1(r)$

(*cell division when idle*)

$\text{Cell}[x, r, m = 1, a] \to \{\text{Cell}[x_1, \frac{r}{2}, m = 1, a], \text{Cell}[x_2, \frac{r}{2}, m = 1, a]\} \quad \textbf{with} \quad \rho_2(a)P(x_1, x_2|x)$

(*cell death*)

$\text{Cell}[x, r, m, a] \to \{\} \quad \textbf{with} \quad \rho_3(x)$

(*Auxin influx from shoot*)

$\{c1 = \text{Cell}[x_1, r_1, m_1, a_1], s1 = \text{shoot}[c1]\} \to \{c1, s1\} \quad \textbf{solving} \quad \{\frac{da_1}{dt} = \gamma(t)\}$

(*Auxin passive transport between cells*)

$\{c1 = \text{Cell}[x_1, r_1, m_1, a_1], c2 = \text{Cell}[x_2, r_2, m_2, a_2], s12 = \text{Spring}[c1, c2]\} \to \{c1, c2, s12\}$
$\quad \textbf{solving} \quad \{\frac{da_1}{dt} = P_t(a_2 - a_1), \frac{da_2}{dt} = P_t(a_1 - a_2)\}$

(*Auxin active transport between cells*)

$\{c1 = \text{Cell}[x_1, r_1, m_1, a_1], c2 = \text{Cell}[x_2, r_2, m_2, a_2], s12 = \text{Spring}[c1, c2]\} \to \{c1, c2, s12\}$
$\quad \textbf{solving} \quad \{\frac{da_1}{dt} = K_0 a_2 f(a_2), \frac{da_2}{dt} = -K_0 a_2 f(a_2)\}$

(*Auxin dissipation*)

$\{c1 = \text{Cell}[x, r, m, a]\} \to \{c1\} \quad \textbf{solving} \quad \{\frac{da}{dt} = -a(d + \frac{\upsilon(r)}{r})\}$

(*cell growth*)

$\{c1 = \text{Cell}[x, r, m = 1, a]\} \to \{c1\} \quad \textbf{solving} \quad \{\frac{dr}{dt} = \upsilon\}$

(*cell neighbors spring*)

$\{c1 = \text{Cell}[x_1, r_1, m_1, a_1], c2 = \text{Cell}[x_2, r_2, m_2, a_2], s12 = \text{Spring}[c1, c2]\} \to \{c1, c2, s12\}$

solving $\{\frac{dx_1}{dt} = \varphi(x_1, r_1, x_2, r_2)\}$

Auxin is secreted from the *shoot* object to the first *Cell* object in the root. Thereafter, auxin is transported between cells by diffusion (passively) and forward to the root tip by the active transport rule according to:

$$f(a) = \frac{\left(\frac{a}{q_{11}}\right)^{p_1}}{1 + \left(\frac{a}{q_{12}}\right)^{p_1}} \frac{1}{1 + \left(\frac{a}{q_2}\right)^{p_2}} \tag{10.1}$$

The active transport rule is derived by reduction from the comprehensive model which includes PIN1 effects on auxin concentration, see [88]. Auxin dissipation refers to any other processes leading to the decrease of auxin concentration in cells due to cell growth and other molecular processes, i.e. direct oxidation or auxin flow transversely to the longitudinal axis.

The first factor of Equation 10.1 describes the activation mechanism of auxin transport whereas the second factor describes the auxin inhibition mechanism. Furthermore, we introduce an auxin influx from shoot (parameter α) as time dependent linearly increasing function.

A cell object may be in either *growth* or *idle* mode where, initially, a cell is in growth mode. The rate function, $\rho_1(r)$, of mode change is a Sigmoid function. A cell may divide only when it is in idle mode. The waiting time for cell division, when the cell is in idle mode, is distributed exponentially. The idle mode signifies the cell cycle checkpoints and the stochastic waiting time during a real cell cycle.

The cell division rate is decomposed to a rate function over auxin concentration and a probability function of the location of new daughter cells. Auxin regulated division function is defined as the following Hill's function:

$$\rho_2(a) = \frac{\left(\frac{a}{k^1_{\text{cell\_div}}}\right)^{h_1}}{1 + \left(\frac{y}{k^2_{\text{cell\_div}}}\right)^{h_2}}$$

Auxin profiles of the simulated DG model are shown in Figure 10.2. The model is robust in some range of parameter values, see [88]. In case of slowly increasing auxin flow from the shoot, we observe persistent location of auxin maximum, in spite of root growth. This result supports the idea that regulated acropetal auxin transport is able to adjust the position of QC in the course of root development.

10.3 Regulation of cell division

Along the root, there are two distinguished zones with increased rates of cell division separated by the QC (Figure 10.3). The spatial profile of mitotic activity in the RAM, measured for cortical cells, is bell-shaped with the maximum located at 10-16 cells distance from the QC [12]. The dividing initial columella cells add an additional maximum to the spatial profile of mitotic activity in the root. Figure 10.3 (B) demonstrates the two distinct maxima: in the meristematic zone of RAM and in the columella initial.

Simulations of cell divisions in the root regulated by auxin failed to provide a profile of mitotic activity along the longitudinal axis that fits the experimental data. The model is extended with additional regulation of cell divisions by a hypothetic substance Y. The predicted substance has the following properties: Y is constitutively synthesized in the QC and its degradation exhibits dose response to the auxin concentration. Y is transported through the cell array by diffusion and regulates the mitotic activity of the cells. The rate of cell division is low (or zero) in the absence or abundance of Y and is high if its concentration is intermediate. The parameters of Y transport were adjusted in order to exhibit the desired behavior.

Y may be replaced by two factors: repressor and activator of cell division. Suppose that (1) the activator moves by diffusion from the QC, where it is synthesized; (2) the repressor's gradient is maximized in the root end. There are a number of hypotheses for the possible repressor and activator candidates [88].

The following grammar rules were augmented to the previous root-model. *Cell* objects have an additional parameter for Y concentration. The rules of Y diffusion (passive transport) and dissipation are similar to the auxin rules. Y is produced in the QC which is recognized by the high auxin concentration of its neighbor cell. Furthermore, Y diffuses (passive transport) between cells, in a higher rate than the diffusion rate of auxin.

(*Y passive transport between cells*)

$$\{c1 = \mathrm{Cell}[x_1, r_1, m_1, a_1], c2 = \mathrm{Cell}[x_2, r_2, m_2, a_2], s12 = \mathrm{Spring}[c1, c2]\} \to \{c1, c2, s12\}$$
$$\textbf{solving } \{\tfrac{dy_1}{dt} = P_{\mathrm{ty}}(y_2 - y_1), \tfrac{dy_2}{dt} = P_{\mathrm{ty}}(y_1 - y_2)\}$$

(*Y synthesis in the QC*)

$$\{c1 = \mathrm{Cell}[x_1, r_1, m_1, a_1, y_1], c2 = \mathrm{Cell}[x_2, r_2, m_2, a_2, y_2], s12 = \mathrm{Spring}[c1, c2]\}$$
$$\to \{c1, c2, s12\} \quad \textbf{solving } \{\tfrac{dy_1}{dt} = \beta \;\; \sigma(a_2 - a_1)\}$$

(*Y dissipation*)

$$\{c1 = \mathrm{Cell}[x, r, m, a, y]\} \to \{c1\} \quad \textbf{solving } \{\tfrac{dy}{dt} = -y(k_{\mathrm{degr\_y}}(a) + \tfrac{\upsilon(r)}{r})\}$$

$k_{\mathrm{degr\_y}}(a_i)$ defines the function of auxin regulated Y degradation, which has a Hill's function form:

$$k_{\text{degr\_y}}(a) = 0.05 + \frac{k^0_{\text{degr\_y}}(\frac{a}{k^1_{\text{degr\_y}}})^{p_1}}{1 + \left(\frac{a}{k^2_{\text{degr\_y}}}\right)^{p_2}}$$

The cell division rule is modified to include the effect of Y concentration:

(*cell division*)

$$\{\text{Cell}[x, r, m = 2, a, y]\} \rightarrow \{\text{Cell}[x_1, \tfrac{r}{2}, m = 1, a, y], \text{Cell}[x_2, \tfrac{r}{2}, m = 1, a, y]\}$$
$$\textbf{with} \;\; k_{\text{cell\_div}}(y)P(x_1, x_2|x)$$

$k_{\text{cell\_div}}(y)$ is defined as:

$$k_{\text{cell\_div}}(y) = 0.001 + \frac{k^0_{\text{degr\_y}}(\frac{y}{k^1_{\text{cell\_div}}})^{h_1}}{1 + \left(\frac{y}{k^2_{\text{cell\_div}}}\right)^{h_2}}$$

This model was able to recapitulate both the observed auxin gradient and mitotic profile as shown in Figure 10.4. In the calculated results, we were able to distinguish between different cell types (see Figure 10.1) by considering both their auxin concentration and mitotic activity.

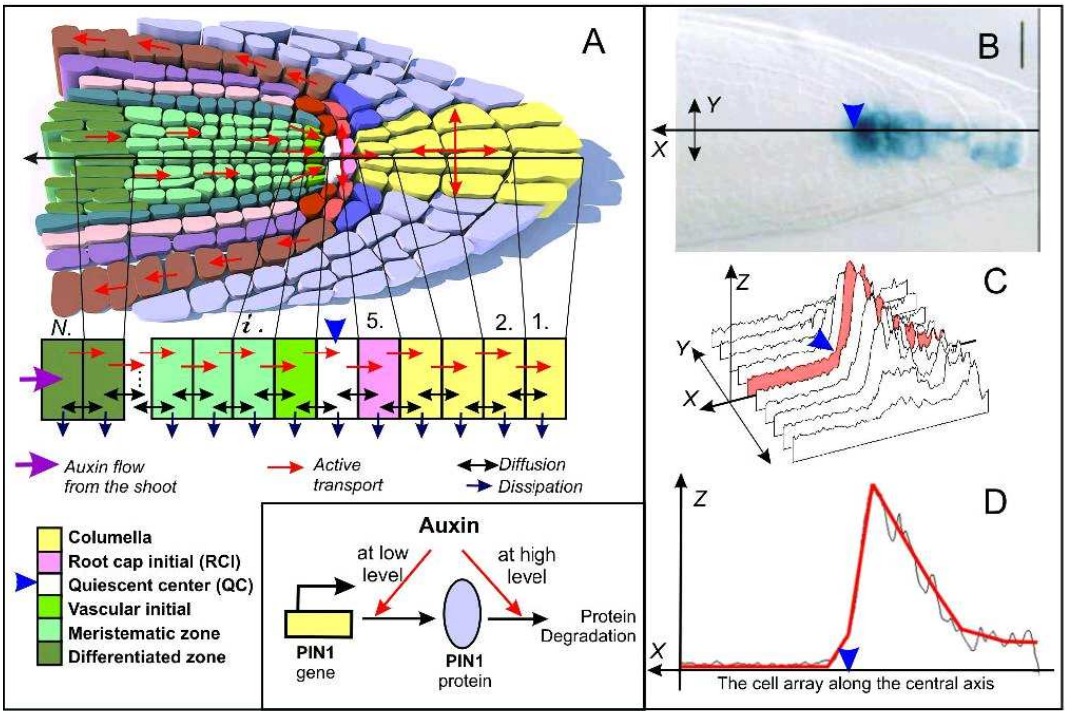


Figure 10.1: Root tip structure of A. thaliana and the 1-Dimensional representation in the DG model. A. Root tip structure with directions of auxin transport and its representation in the 1D DG model. The cell array along the central longitudinal axis was used to simulate auxin redistribution. Arrows mark the processes studied in the model that affect auxin distribution. The generalized mechanism of auxin regulated PIN1 expression includes both regulation of PIN1 synthesis and degradation by auxin on a concentration dependent manner. B. The DR5::GUS gene expression presents auxin distribution in the root tip of young seedlings (adapted from [125]). C. The surface plot of DR5 activity in the root tip according to the experimental data (B). D. The plot shows correspondence of the static model solution result (red line) to semi-quantitative data of auxin distribution (black line) scanned from (B) and marked by red in (C). The blue arrowhead marks the QC position.

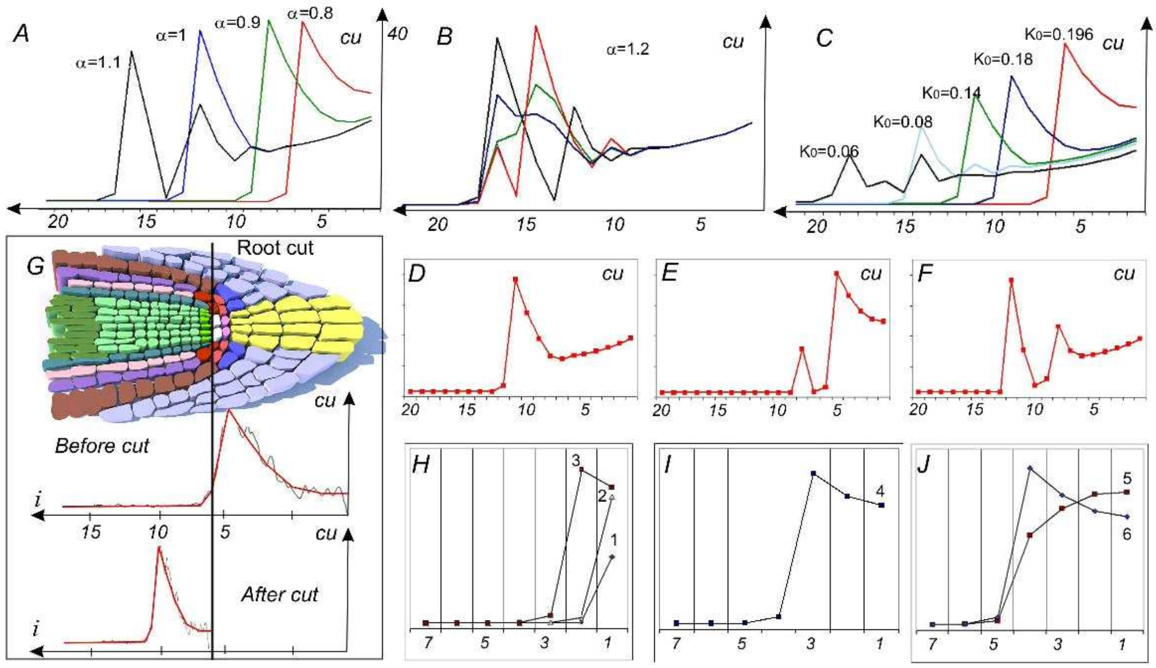


Figure 10.2: Changes in auxin distribution pattern in response to varying in parameter values. A-B, D-F. The model behavior under increase of auxin flow from the shoot. In the static model simulation (A, B) and while simulating root growth (D-F). C. The model behavior in response to varying values of parameter q2. D, H-J. Simulation of experiment on QC laser ablation or root cut. E. Shift of auxin maximum toward the middle of the root in response to increase auxin flow from the shoot. F-G. The different examples of the additional maximum formation in the model with cell divisions in response to increase auxin flow from the shoot. D. For simulation of QC laser ablation or root cut in silico, we removed from the array all cells located distally to vascular initial and started calculation with current auxin concentrations in cells. Qualitative correspondence of model solutions to experimentally observed auxin distribution before and after cut are shown. H-J. Computer simulation of changes in auxin distribution after cut is presented for the first 7 cells.

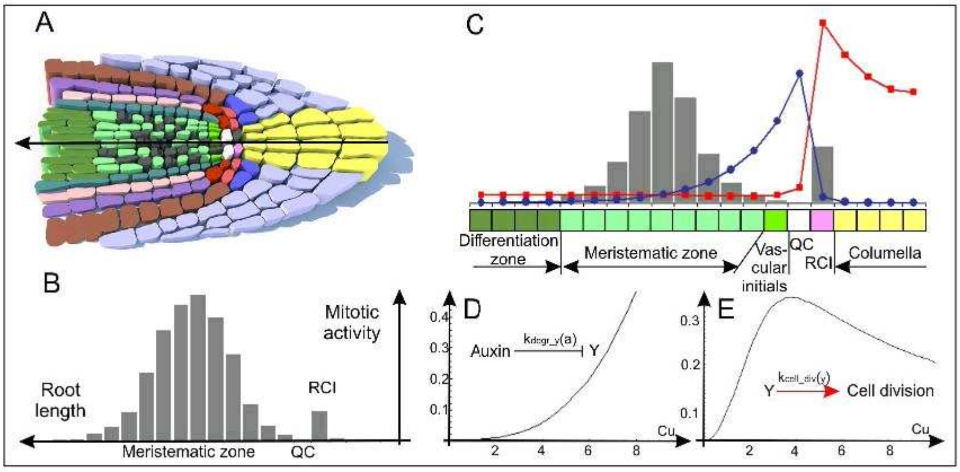


Figure 10.3: Mitotic activity in the root and its simulation

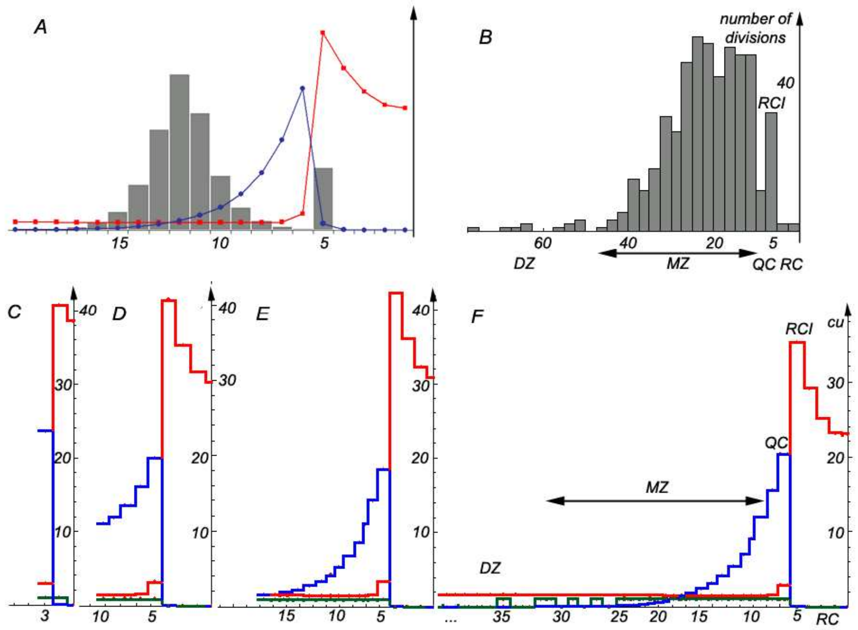


Figure 10.4: Simulation of root growth along the root longitudinal axis. A. Profiles of auxin (red) and Y (blue) distributions and rates of cell division (gray columns) in conventional units along the root longitudinal axis. B. The profile of mitotic activity along the longitudinal axis of root calculated in the model with cell divisions. The ordinate is the number of cell divisions, abscissa the serial number of the cell starting from the root end. C-F. Auxin (red) and Y (blue) distribution in conventional units during simulation of root growth. The green curve indicates the mode of cells (1- idle, 0- growth).C. the simulation was initialized with three cells; D. 10 cells; E. 20 cells; F. more than 100 cells. Cells of different types can be distinguished by considering both their auxin concentration and mitotic activity (see the main text for more details): QC – quiescent center; RCI – root cap initial; RC- root cap; MZ meristematic zone; DZ- differentiation zone.

Chapter 11

Modeling the olfactory epithelium

The olfactory epithelium (OE) is a self-renewing tissue in which multipotent, progenitor cells undergo a multistage process of successive cell type transitions and divisions regulated by intercellular signaling. We investigate the possible mechanisms for such a stem cell lineage to maintain robust size control and spatial distribution in the face of environmental and genetic perturbations, despite a number of complicating factors: (a) cell transitions may be stochastic, (b) OE cells at different lineage stages are distributed in a non-uniform manner in space, and (c) length scales associated with such spatial inhomogeneity may be substantial compared with the diffusion lengths of signaling molecules such as GDF11, Activin and Follistatin. The stochastic simulation strategy uses the Dynamical Grammar (DG) multiscale modeling framework, under which a dynamical system comprises both continuous differential dynamics and discrete stochastic processes. The resulting simulations are used to investigate conditions sufficient to produce the observed size control and spatial stratification of cell types in the OE.

11.1 Introduction

The olfactory epithelium (OE) of the mouse is a useful system for understanding the complex interactions that regulate tissue development and regeneration. The OE is one of a few regions of the vertebrate nervous system that undergoes neurogenesis, or *de novo* production of neurons, throughout life. The olfactory receptor neurons (ORNs) are in direct contact with the external environment through the nasal cavity. Due to their exposed location, ORNs are subject to variable damage by exogenous factors and must be constantly replenished. Remarkably, the thickness of the ORN cell layer is maintained almost constant throughout the OE.

Neurogenesis is maintained by a stem cell niche, which is a locus in the tissue where multipotent, self-renewing progenitor cells reside. Stem cells undergo a multistage process of successive cell type transitions and divisions regulated by intercellular signaling, as described in [21]. Stem cells may differentiate into progenitor cells which are also known as transit amplifying (TA) cells. TA cells give rise to terminally differentiated (TD) cells, the ORNs. Both stem and TA cells can also replicate, hence reproduce cells of the same cell type. The rate of cell division and the types of the resulting daughter cells vary across different cell types. The assignment of cell types to daughter cells is crucial for size control in the OE.

Recent studies show that the size of the OE tissue is controlled by two inhibitory feedback signals: Growth Differentiation Factor 11 (GDF11) [134], and ActivinβB (ActβB) [55]. Both GDF11 and ActβB are members of transforming growth factor-β (TGF-β) superfamily of ligands. GDF11, which is expressed by both ORNs and TAs, inhibits the proliferation of TAs, whereas ActβB, which is expressed by all neuronal lineage cell types, inhibits the proliferation of stem cells. A schematic view of a related generic cell lineage model is shown in Figure 11.1.

An additional diffusible molecule, Follistatin (FST), is produced in and around the OE, and acts as a competitive inhibitor of GDF11, ActβB and their homologs. The role of FST in the regulation of the cell lineage is established in [55].

A variety of different mathematical modeling approaches may be used to simulate such a complex and dynamic system. A common approach in systems biology is to use ordinary differential equation (ODE) models in which the system is assumed to be spatially "well-stirred", continuous, and deterministic. An ODE formulation for modeling the multistage cell lineage in the OE is shown in [77]. The cell lineage ODE model of [77] is used to establish stability enhancing conditions: autoregulation of TA proliferation and low TD death rate. A recent study [73] identified basic strategies by negative feedback factors in order to control the growth of multi-stage cell lineages. The feedback regulation that exists in the OE has important control objectives such as regeneration speedup, and decreased sensitivity to initial condition and parameter values [73].

However, stochastic events and an inhomogeneous spatial distribution, which are not modeled by ODEs, may have a substantial effect on the functionality of the OE system. Cells at different lineage stages are distributed in a non-uniform manner in space (inconsistent with the "well-stirred" assumption), and length scales associated with such spatial inhomogeneity may be comparable or large compared to the length scales over which signaling molecules such as GDF11 and Activin diffuse before being captured and destroyed. These observations suggest that cell behavior and cell decisions (such as whether to replicate or differentiate) depend only on conditions within a local region around the cell, and such a local region may be susceptible to stochastic fluctuations.

This chapter examines cell lineage models that represent the development of local regions in the OE. The simulations involve relatively small cell populations (around

50-100 cells) in which cells are modeled as discrete entities, and processes such as cell replication, differentiation or death are modeled as sequences of discrete stochastic events. The models studied include feedback molecules whose state is modeled either by global or locally spatially continuous concentrations.

Dynamical grammars (DGs) [93] provide a convenient mathematical and computational framework for modeling hybrid discrete/continuous systems such as OE cell lineages coupled with subcellular ODE dynamics. DGs describe a broad family of stochastic processes that include systems with time-varying structure, and both continuous and discrete processes. Grammatical transitions describe stochastic jump events such as cell differentiation and replication, whereas continuous-time rules describe cell growth, movement and diffusion of secreted molecules.

The simulations described in Section 11.3 of this chapter reveal key differences between the stochastic and deterministic models. In stochastic simulations, there is a strictly positive probability for extinction of the stem cell population. The probability for stem cell extinction is high because the effects of feedback signals are dominated by the neuronal population. We further show how stochastic variations may contribute to the dominance of a monoclonal cell population (all cells of which originate from single stem cell) in small sections of the tissue corresponding to the range of signal diffusion.

Another factor that is likely to play an important role in size control of the OE is the highly non-uniform spatial distribution of cells. Stem and progenitor cells form a layer near the basal lamina (BL). Such a layered stem cell distribution may be advantageous for maintaining uniform thickness across the tissue. Potential mechanisms to maintain the layered spatial distribution of cells are explored using two-dimensional stochastic models in Section 11.3.2. A model that includes strong affinity of stem and progenitor cells toward the BL is able to recapitulate the observed lamination in the OE. In this way, we show that DGs are capable of simulating a rich set of biologically known or plausible mechanisms by which OE homeostasis (both in cell number and layering) could be maintained, despite stochastic fluctuations, and can therefore serve as the engine for more detailed explorations of such hypotheses in the future.

The chapter is organized as follows. Section 11.2 describes the DG formulation of our models. Simulations results are shown in Section 11.3. Discussion of related modeling frameworks, a summary of our results, and a discussion of future directions are in Section 11.4.

11.2 Methods and model description

Following [134, 73], the cell lineage model is a simplified OE model that is comprised of just three cell types: stem, transit amplifying (TA), and terminally differentiated (TD) cells. Both stem and TA cells may replicate or differentiate after a random time interval (roughly the length of the cell cycle). There are two feedback signals denoted as ϕ_0 and ϕ_1. ϕ_0 is secreted by all cells and inhibits the replication probability of stem cells, whereas ϕ_1 is secreted by TA and TD, and inhibits the replication probability of TA cells. TD cells die according to a fixed stochastic rate. A schematic view of the model is shown in Figure 11.1. In this mapping, the feedback signals ϕ_0 and ϕ_1 characterize GDF11 and ActβB respectively.

The DG rules for the cell lineage model are defined using an object-oriented syntax where Cell represents an abstract object that can be specialized into one of three cell types: Stem :: Cell, TA :: Cell, and TD :: Cell. Any process that applies to Cells also applies to all three cell types. The following grammar rules describe mitosis events according to the dynamics in Figure 11.1:

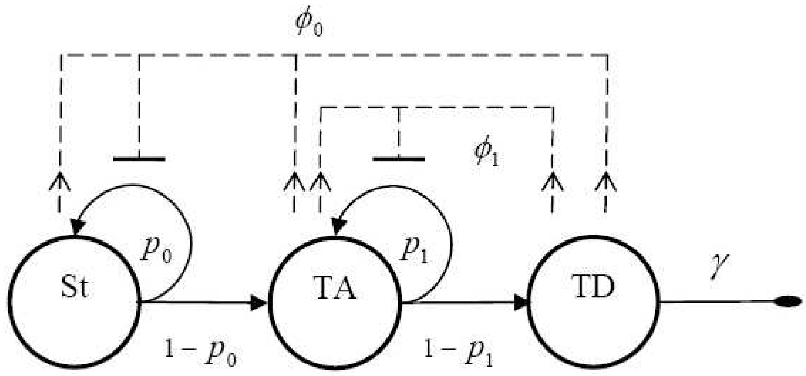


Figure 11.1: Schematic of cell lineage with negative feedback regulation on cell proliferation. Terminally differentiated (TD) cells derive from transit-amplifying (TA) cells which arise from stem cells (St). Feedback regulation (ϕ_0) on stem cell replication probability (ρ_0) is a function of total cell number. Feedback regulation (ϕ_1) on TA cell replication probability (ρ_1) is a function of TA and ORN cell number. γ is TD death rate.

(*stem cell mitosis: *)

$$
\begin{aligned}
\{\text{Stem} :: \text{Cell}[x, v], g = \text{Field}[0, \phi_0]\} \rightarrow & \{\text{Stem} :: \text{Cell}[x_1, v/2^{1/d}], \text{Stem} :: \text{Cell}[x_2, v/2^{1/d}], g\} \\
& \quad \textbf{with} \quad \rho_{0,1}(\phi_0(x), v) \cdot P(x_1, x_2|x) \\
\{\text{Stem} :: \text{Cell}[x, v], g = \text{Field}[0, \phi_0]\} \rightarrow & \{\text{Stem} :: \text{Cell}[x_1, v/2^{1/d}], \quad \text{TA} :: \text{Cell}[x_2, v/2^{1/d}], g\} \\
& \quad \textbf{with} \quad \rho_{0,2}(\phi_0(x), v) \cdot P(x_1, x_2|x) \\
\{\text{Stem} :: \text{Cell}[x, v], g = \text{Field}[0, \phi_0]\} \rightarrow & \{\ \ \text{TA} :: \text{Cell}[x_1, v/2^{1/d}], \quad \text{TA} :: \text{Cell}[x_2, v/2^{1/d}], g\} \\
& \quad \textbf{with} \quad \rho_{0,3}(\phi_0(x), v) \cdot P(x_1, x_2|x)
\end{aligned}
\tag{11.1}
$$

(*TA mitosis: *)

$$
\begin{aligned}
&\{\ \ \mathrm{TA} :: \mathrm{Cell}[x, v], g = \mathrm{Field}[1, \phi_1]\} \rightarrow \{\ \ \mathrm{TA} :: \mathrm{Cell}[x_1, v/2^{1/d}],\ \ \mathrm{TA} :: \mathrm{Cell}[x_2, v/2^{1/d}], g\} \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \textbf{with} \quad \rho_{1,1}(\phi_1(x), v) \cdot P(x_1, x_2|x) \\
&\{\ \ \mathrm{TA} :: \mathrm{Cell}[x, v], g = \mathrm{Field}[1, \phi_1]\} \rightarrow \{\ \ \mathrm{TA} :: \mathrm{Cell}[x_1, v/2^{1/d}],\ \ \mathrm{TD} :: \mathrm{Cell}[x_2, v/2^{1/d}], g\} \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \textbf{with} \quad \rho_{1,2}(\phi_1(x), v) \cdot P(x_1, x_2|x) \\
&\{\ \ \mathrm{TA} :: \mathrm{Cell}[x, v], g = \mathrm{Field}[1, \phi_1]\} \rightarrow \{\ \ \mathrm{TD} :: \mathrm{Cell}[x_1, v/2^{1/d}],\ \ \mathrm{TD} :: \mathrm{Cell}[x_2, v/2^{1/d}], g\} \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \textbf{with} \quad \rho_{1,3}(\phi_1(x), v) \cdot P(x_1, x_2|x)
\end{aligned}
\tag{11.2}
$$

This set of rules describes symmetric and asymmetric cell division. The probability of each option, $P_{i,j}$, is expressed in the rate functions $\rho_{i,j}$, as follows:

$$
\begin{aligned}
\rho_{0,j} &= P_{0,j}(\phi_0(x))\hat{\rho}(v) \\
\rho_{1,j} &= P_{1,j}(\phi_1(x))\hat{\rho}(v)
\end{aligned}
\tag{11.3}
$$

$\hat{\rho}(v)$ is the total cell division rate function (for both symmetric and asymmetric choices) which is dependent on the cell size. The function is defined as a step function: 0 whenever $v < v_{\max}$ and $0.5/h$ for $v \geq v_{\max}$ (h is hours). The probabilities for symmetric or asymmetric cell division are expressed by the functions $P_{0,j}$ and $P_{1,j}$. We assume that the probability of a cell becoming any given cell type is independent of its sibling's type, which mathematically is stated as : $P_{i,1} = {\psi_i}^2$, $P_{i,2} = 2(1-\psi_i)\psi_i$, and $P_{i,3} = (1-\psi_i)^2$. The feedback signals ϕ_0 and ϕ_1 are depicted, in the DG model, as parameters of the *Field* objects, whereas mitosis rate functions depend on the concentration of signal in the location of the cell ($\phi(x)$).

The probability functions of stem cell replication $P_{0,1}(\phi)$ and TA replication $P_{2,1}(\phi)$ are defined as follows:

$$P_{i,1}(\phi_i) = P_i^{max}\frac{S_i}{\phi_i + S_i} \qquad \text{for } i \in \{0,1\}\,, \text{where } 0 \leq P_i^{max} \leq 1 \tag{11.4}$$

The replication probabilities are Hill functions (with Hill coefficient 1) which are commonly used in modeling feedback functions [3]. The constants S_0 and S_1 determine the level of ϕ_0 and ϕ_1, respectively, where the replication probabilities are half of the maximum.

A daughter cell's linear size $v/2^{1/d}$ depends on the model's spatial dimension d which equals two in Section 11.3.2. In Section 11.3.1 a non-spatial variant of the grammar rules is used, in which the spatial parameters are omitted and signals concentrations are assumed to be uniform in space.

The final discrete rule in the model describes TD cell death:

$$\text{TD} :: \text{Cell}[x, v] \rightarrow \{\}, \quad \textbf{with} \quad \gamma/T \tag{11.5}$$

In addition to discrete stochastic events, the model incorporates continuous dynamics for the following processes: secretion, diffusion and decay of signaling molecules, and cell growth and movement. The DG formalism includes continuous-time rules which are rules that evolve the matching objects' parameters forwards in time by sets of differential equations. The following rules describe secretion of the signaling molecules by their source cells, and diffusion/decay of molecules in the two-dimensional field. In each case, the **solving** clause contains the set of applicable ODEs or PDEs.

$\{c = \text{Cell}[x, v], g = \text{Field}[0, \phi_0]\} \rightarrow \{c, g\},$ **solving** $\{\frac{\partial \phi_0}{\partial t} = k(\hat{x}; x, v)\}$

$\{c = \text{TA} :: \text{Cell}[x, v], g = \text{Field}[1, \phi_1]\} \rightarrow \{c, g\},$ **solving** $\{\frac{\partial \phi_1}{\partial t} = k(\hat{x}; x, v)\}$

$\{c = \text{TD} :: \text{Cell}[x, v], g = \text{Field}[1, \phi_1]\} \rightarrow \{c, g\},$ **solving** $\{\frac{\partial \phi_1}{\partial t} = k(\hat{x}; x, v)\}$

Here $\hat{x}$ denotes the spatial coordinates of the field variables (ϕ_0 and ϕ_1) and $k(\hat{x}; x, v)$ denotes the secretion function which is defined as:

$$k(\hat{x}; x, v) = \begin{cases} 1 & \text{if } ||\hat{x} - x||^2 < v/\pi \\ 0 & else \end{cases}$$

Diffusion and decay are defined in the following rule that applies to both ϕ_0 and ϕ_1:

$\text{Field}[i, \phi] \rightarrow \text{Field}[i, \phi],$ **solving** $\{\frac{\partial \phi}{\partial t} = D\frac{\partial^2 \phi}{\partial x^2} - d\phi\}$

Stem and TA cells grow in constant rate (until division occurs) whereas TD cell growth is diminished as the cell volume approaches $v_{\max}$. The TD cell growth function is defined as the following sigmoidal function:

$$\varsigma(v) = \frac{1}{1 + e^{\beta(v - v_{\max})}}$$

Cells constantly move due to their own growth and proximity to other cells and to the BL boundary. The initial assumption is that cells' movement due to collisions in space is unrelated to cell type, so we define these rules for the abstract Cell object. Grammar rules which are defined for the abstract Cell object apply to all concrete objects (eg. Stem, TA, and TD).

(*cell movement due to neighbor cell position: *)

$\{c1 = \mathrm{Cell}[x_1, v_1], c2 = \mathrm{Cell}[x_2, v_2]\} \to \{c1, c2\},$ **solving** $\{\frac{dx_1}{dt} = \varphi(x_1, v_1, x_2, v_2)\}$

(*cell movement due to boundary position: *)

$\{c1 = \mathrm{Cell}[x_1, v_1], b = \mathrm{Boundary}[x_2]\} \to \{c1, b\},$ **solving** $\{\frac{dx_1}{dt} = \varphi(x_1, v_1, x_2, 0)\}$

Cells move under the force (given by the function φ) exerted by weak spring connections to neighboring cells and to the boundaries. For more details on the weak spring model see [120] or [91]. Further discussion on the spatial rules and possible forces in the system is continued in Section 11.3.2. The computational aspects of DG simulations are discussed in [137].

A deterministic, non-spatial approximation of the foregoing stochastic model could be stated as the following set of ODEs (one may find equivalent models in [77] and [73]):

$$\frac{dc_0}{dt} = (2\rho_{0,1}(\phi_0) + \rho_{0,2}(\phi_0) - 1/(a_0 T))\, c_0 \tag{11.6}$$
$$\frac{dc_1}{dt} = (\rho_{0,2}(\phi_0) + 2\rho_{0,3}(\phi_0))\, c_0 + (2\rho_{1,1}(\phi_1) + \rho_{1,2}(\phi_1) - 1/T)\, c_1 \tag{11.7}$$
$$\frac{dc_2}{dt} = (\rho_{1,2}(\phi_1) + 2\rho_{1,3}(\phi_1))\, c_1 - \gamma c_2/T \tag{11.8}$$
$$\frac{\partial \phi_0}{\partial t} = \sum_{i=0}^{2} c_i - d\phi_0 \tag{11.9}$$
$$\frac{\partial \phi_1}{\partial t} = \sum_{i=1}^{2} c_i - d\phi_1 \tag{11.10}$$

a_0 is an additional parameter to adjust the average cell cycle time of stem cells.

11.3 Results

The results have three parts : 1) a non-spatial stochastic grammar is compared to the deterministic ODE model; 2) In order to incorporate the spatial information, possible mechanisms for tissue lamination are explored; and 3) A complete 2D spatial grammar reveals possible interplay between cell lineages.

11.3.1 Divergence between stochastic and deterministic solutions caused by random extinctions

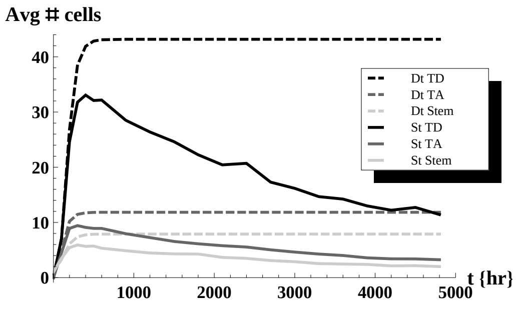
Before delving into a complex spatial model, a stochastic non-spatial cell lineage model is explored and compared to the deterministic equivalent. First, the set of free parameters (P_0^{max}, a_0, S_0, P_1^{max}, S_1, and γ) were optimized such that the dynamics of the deterministic model resemble the observed dynamics in the OE system. Taking into account results from genetic experiments [134][55], we assume that (1) in the absence of feedback from ϕ_0 (ϕ_0 is set to zero) twice as many stem cells develop, corresponding to results of ActβB mutations in [55]; (2) a ϕ_1 mutation develops twice as many TAs and 50% more TD cells (no change in stem population), corresponding to the results for GDF11 mutations in [134]; and (3) double mutants (both ϕ_0 and ϕ_1 set to zero) develop twice as many stem and TA cells and 50% more TD cells, corresponding to the results of double mutants in [55].

The set of free parameters were optimized according to the sum squared error (SSE) between the ODE model's (Equation 11.611.6- 11.8) output and the target cell numbers: one initial stem cell and 5 stem cells, 10 TA cells and 20 TD cells at $t = 7$ days for wildtype (after seven days of development the OE possesses a full cohort of stem, TA and TD cells), and the appropriate corresponding cell numbers for mutant animals. The optimization algorithm used was Powell's method [103] as implemented in the *Mathematica* computer program. Figure 11.2(a) shows the trajectories of the ODE model under the best-fit set of parameters. In addition Figure 11.2(a) plots the trajectories of the average cell quantities in the stochastic, non-spatial model given the same set of parameter values.

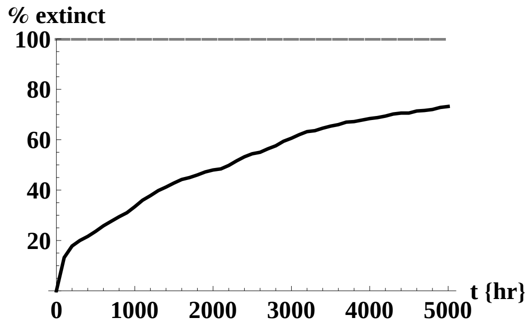
As depicted in Figure 11.2(a), the stochastic model diverges from its deterministic counterpart as time progresses. In fact, the stochastic model cannot converge to a strictly positive steady state. Over time, stochastic simulations have an increasing probability for differentiation of the entire stem cell population. Figure 11.2(b) plots the percentage of simulations that result in total elimination of stem cells over time. Since the feedback signal is dominated by the secretion by TAs and TDs, random reductions in the number of stem cells are not sensed by the remaining stem cells and result in their total extinction. When the cell lineage system is modified such that the signal on stem cells is secreted only or mostly by stem cells (which is not the case in the OE), then the rate of stochastic extinctions is dramatically reduced.

We examine the extent to which stem cell extinction is responsible for the divergence between stochastic and deterministic results by artificially maintaining a minimum of one stem cell in the system. The results are shown in Figure 11.2(c). In this case, the averaged stochastic simulation follows a trajectory close to the deterministic one.

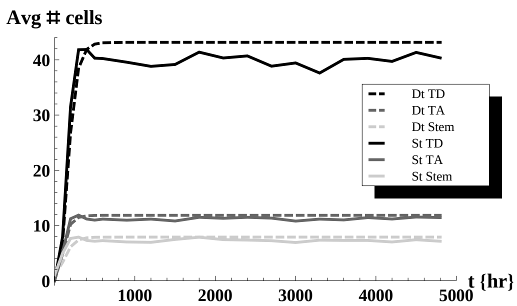
The continuous feedback functions ϕ_0 and ϕ_1 cannot be directly represented in a finite structure if considering a numerical solution. However, the concentration level can be approximated by the number of secreting cells. Another difficulty is in the representation of an infinite number of cell states. This difficulty can be ameliorated by representing a finite number of states and discarding states with high number of cells due to their negligible probability in a negative feedback system.



(a)



(b)



(c)

Figure 11.2: Divergence between stochastic and deterministic solutions results from random extinctions. 11.2(a) Stochastic simulations diverge from deterministic simulations over time. Plot shows the average number of cells (stem, TAs, and TDs) over simulated time. The straight lines depict the mean of the stochastic trajectories whereas the dashed lines depict the deterministic trajectories. The stochastic average is over 500 simulations. 11.2(b) Number of extinctions in stochastic simulations increases over time. Plot shows percentage of extinct simulations over time (out of 500 simulations). An extinct simulation is defined as a simulation that has no surviving stem cells. 11.2(c) A one-stem-cell minimum constraint on stochastic simulations eliminates most of the divergence from deterministic trajectories. A plot of the average number of cells over time. The stochastic model is modified by enforcing a zero probability for stem cell differentiation if there is only one stem cell left. Note that in this setting the stochastic average follows closely the deterministic solution. The stochastic average is over 500 simulations. Parameters values used were: $P_0^{max} = 0.7$, $a_0 = 2$, $S_0 = 84.5$, $P_1^{max} = 0.56$, $S_1 = 45.2$, and $\gamma = 0.36$, and the decay rate $d = 5 * 10^{-4}\ s^{-1}$.

Even for the modifications mentioned above, the three cell type model used in Section 11.3.1 could not be well approximated due to such computational limitations. Therefore, we resort to a numerical solution of a simplified model which has only two cell types: stem cells and differentiated cells. The differentiated cells are a combination of TA and TD cells (denoted as Diff cells). One portion of the differentiated cells, $\alpha\hat{c}_1$, can proliferate (as TAs) whereas cells in the second portion (representing the TDs) are subject to death with rate constant γ. The ratio parameter, α, was determined according to the steady state number of TA and TD cells in the deterministic three cell type model.

The complete grammar model of the simplified two cell types system is shown below. Note that these grammar rules apply for objects of 'Cells' type instead 'Cell' type. The $\hat{c}$ parameters represent the number of individual cells in 'Cells' objects. The rates of discrete events are multiplied by the appropriate $\hat{c}$ counter.

(*stem cell mitosis: *)

$$
\begin{aligned}
\{\mathrm{Stem} :: \mathrm{Cells}[\hat{c}_0], \mathrm{Diff} :: \mathrm{Cells}[\hat{c}_1]\} \rightarrow & \{\mathrm{Stem} :: \mathrm{Cells}[\hat{c}_0 + 1], \mathrm{Diff} :: \mathrm{Cells}[\hat{c}_1]\} \\
& \textbf{with} \quad \hat{c}_0 \ \rho_{0,1}(\hat{c}_0 + \hat{c}_1) \\
\{\mathrm{Stem} :: \mathrm{Cells}[\hat{c}_0], \mathrm{Diff} :: \mathrm{Cells}[\hat{c}_1]\} \rightarrow & \{\mathrm{Stem} :: \mathrm{Cells}[\hat{c}_0], \mathrm{Diff} :: \mathrm{Cells}[\hat{c}_1 + 1]\} \\
& \textbf{with} \quad \hat{c}_0 \ \rho_{0,2}(\hat{c}_0 + \hat{c}_1) \\
\{\mathrm{Stem} :: \mathrm{Cells}[\hat{c}_0], \mathrm{Diff} :: \mathrm{Cells}[\hat{c}_1]\} \rightarrow & \{\mathrm{Stem} :: \mathrm{Cells}[\hat{c}_0 - 1], \mathrm{Diff} :: \mathrm{Cells}[\hat{c}_1 + 2]\} \\
& \textbf{with} \quad \hat{c}_0 \ \rho_{0,3}(\hat{c}_0 + \hat{c}_1)
\end{aligned}
\tag{11.11}
$$

(*Diff cell mitosis: *)

$$\{\text{Stem} :: \text{Cells}[\hat{c}_0], \text{Diff} :: \text{Cells}[\hat{c}_1]\} \rightarrow \{\text{Stem} :: \text{Cells}[\hat{c}_1], \text{Diff} :: \text{Cells}[\hat{c}_1 + 1]\} \\ \textbf{with} \quad \alpha\ \hat{c}_1/T \tag{11.12}$$

(*Diff cell death: *)

$$\text{Diff} :: \text{Cell}[\hat{c}_1] \rightarrow \text{Diff} :: \text{Cell}[\hat{c}_1 - 1], \quad \textbf{with} \quad (1-\alpha)\ \hat{c}_1\ \gamma/T \tag{11.13}$$

The deterministic version of this simplified stochastic model is stated as follows:

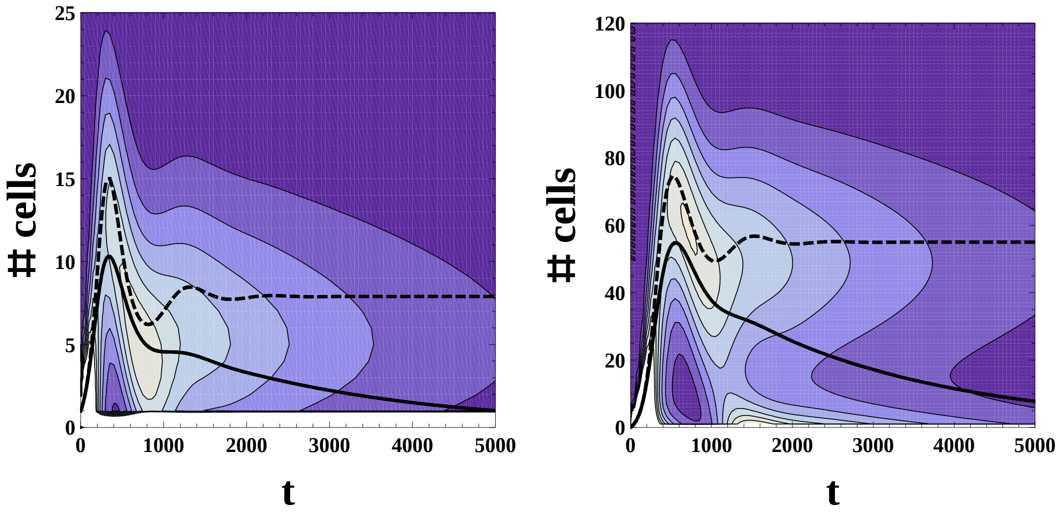
$$\frac{d\hat{c}_0}{dt} = \left(2\rho_{0,1}(\hat{c}) + \rho_{0,2}(\hat{c}) - 1/\left(a_0 T\right)\right)\hat{c}_0 \tag{11.14}$$

$$\frac{d\hat{c}_1}{dt} = \left(\rho_{0,2}(\hat{c}) + 2\rho_{0,3}(\hat{c})\right)\hat{c}_0 + \left(\alpha - (1-\alpha)\,\gamma\right)\hat{c}_1/T \tag{11.15}$$

$$\hat{c} = \hat{c}_0 + \hat{c}_1 \tag{11.16}$$

A solution was obtained by numerically solving the master equation with an artificial cap of 30 stem cells and 150 differentiated cells. This cap is greater than the number of cells typically present in simulations of the stochastic model. Furthermore, the contour plots in Figure 11.3 show that the probability of states outside of the grid is negligible.

The results in Figure 11.3 show the probability distribution of the system over time. There are two maxima for the probability of stem and differentiated cells. One maximum is in parallel to the deterministic solution, whereas the second maximum is fixed at the zero state. The probability of the zero state (0 stem and differentiated cells) gradually increases and converges to 1.



(a) Stem cells distribution over time (b) Differentiated cells distribution over time

Figure 11.3: Contour plots of the master equation solution show bimodal distribution of cell counts over time. The master solution is for a simplified model with two cell types: stem cells and differentiated cells. The deterministic version is defined in Equations 11.14-11.16. The ratio parameter α is set 0.21 according to the ratio of TA to TD cells in the steady state solution of the three cell type model. Straight lines depict the mean value whereas dashed lines show the deterministic solution. The contour plots show that the probability distribution over time has two maxima points: one is on the zero state and the second is in parallel to the deterministic trajectory. The probability of the zero state increases over time and converges to 1. The master equation was solved for states which have at most 30 stem cells and 150 differentiated cells. The equations for boundary states are adjusted accordingly. The other parameters values used here are: $P_0^{max} = 0.7$, $a_0 = 2$, $S_0 = 84.5$, and $\gamma = 0.36$, and the decay rate $d = 5 * 10^{-4}\ s^{-1}$.

11.3.2 Dynamics of lineage trees in spatial-stochastic models

The spatial distribution of cells and signaling molecules may have an impact on the functionality of the tissue. The spatial-stochastic DG model reveals that cell populations of different lineage trees in the same local region may have significant effect on each other. In order to replicate the spatial dynamics in the OE, we first discuss few possible mechanisms for cell lamination.

Possible mechanism for lamination in the OE

The OE is a highly laminated structure. To explore possible mechanisms that underlie OE lamination, we use the 2D model with spring potentials of Section 11.2. The spring potential is decomposed to two components: repulsion due to physical constraints and attraction or repulsion due to cell type adhesion. Both potentials decay exponentially as a function of the distance between the two cells. The springs are disconnected (hence weak springs) when the distance between the two cells is more than the average cell diameter (10μm).

The three boundaries (the bottom boundary that represents the BL and the two side boundaries) are impenetrable for cells. The tissue is assumed to grow uniformly in both directions. In the simulations, the side boundaries artificially maintain the horizontal width in accordance with the uniform growth assumption.

As lamination is observed even in double mutant mice ($\text{GDF11}^{-/-}/\text{Act}\beta\text{B}^{-/-}$) [55], we assume that the feedback signals play no essential role in the spatial orientation of cells in the OE. Thus, in the following simulations, feedback is suppressed.

Figure 11.4 presents the spatial distribution of different cell types across the tissue (from basal to apical). The first configuration is the simplest: every cell has the same affinity force toward the BL and no adhesion forces exist between cells. This basic configuration can not produce lamination as cells distribute uniformly across the tissue (Figures 11.4(a) and 11.4(b)).

Differential adhesion (DA) [123] is a well known cell sorting mechanism that postulates that sorting emerges because of random movement and different adhesion forces between cells of different types. In our system we can use a limited version of DA without random movement of cells. TD cells are assumed to have high adhesion force for each other. As in DA, cells with higher adhesion (TDs) form clusters in the center of the tissue. Such mechanism, however, does not produce directional layering of cells in the OE (Figures 11.4(c) and 11.4(d)).

Finally, we introduce a model configuration with differential affinity to the BL. Stem and TA cells are assumed to have stronger affinity to the BL than the TD cells. Figures 11.4(e) and 11.4(f) show that the desired spatial distribution with lamination does emerge in this model. The following section's results were produced with this differential affinity configuration.

Dominance of a single lineage tree over local region

Given the lamination mechanism of differential affinity, we investigated dynamics of the complete spatio-stochastic DG model. Following the analysis in [73], the apical and side boundaries are assumed to be impenetrable to signaling molecules whereas the basal side is a leaky boundary. Furthermore, in this model, signaling molecules are degraded at a faster rate in the region outside of the basal boundary (the stroma in the OE model). The increased degradation of signaling molecules is based on the observation that FST, which is known as an irreversible inhibitor of both GDF11 and ActβB [55] [4] [30], is produced in the stroma.

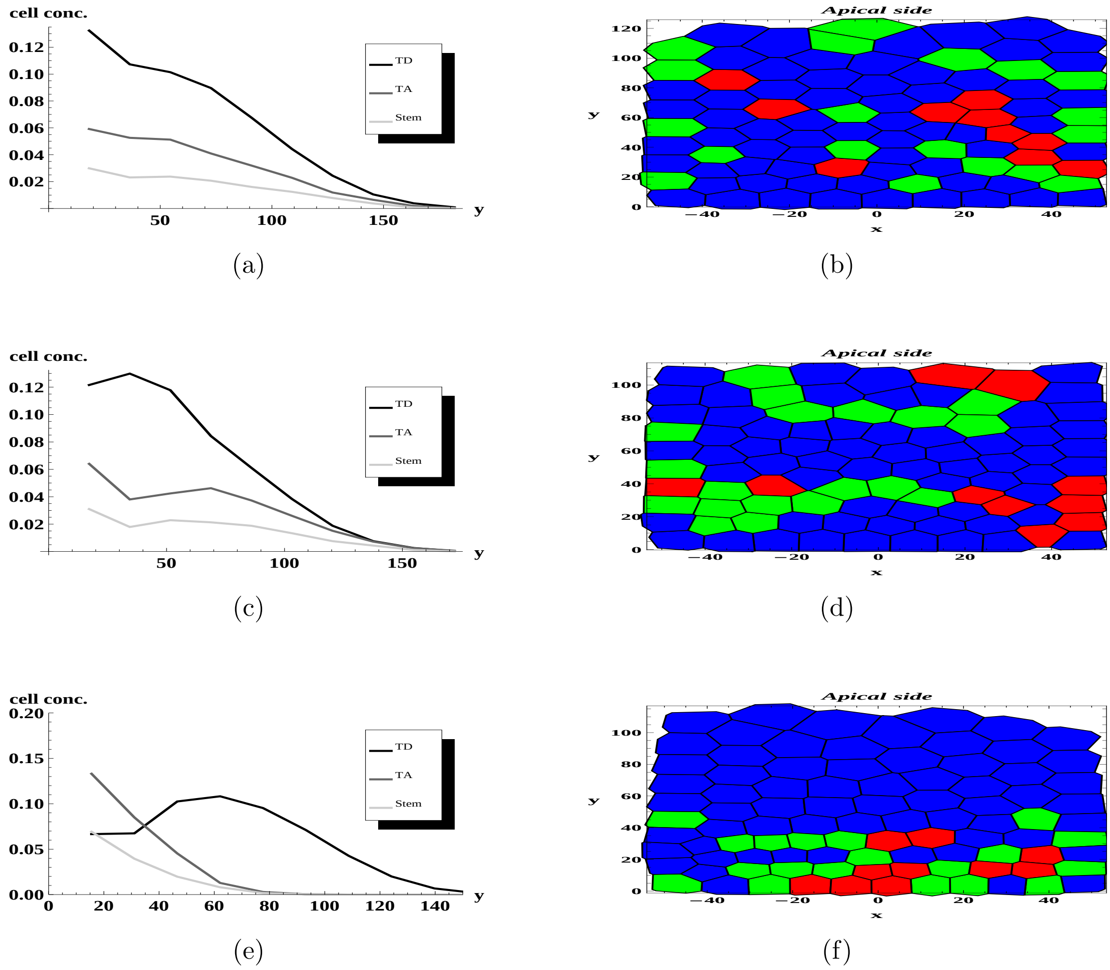


Figure 11.4: Increased affinity of stem cells and TAs to the basal lamina is sufficient to mimic OE lamination. 11.4(a),11.4(c),11.4(e): Plots of cell concentration over the basal-apical axis for different model configurations at time 200h. 11.4(b),11.4(d),11.4(f): Voronoi diagrams of the tissue state in one simulation (same random seed number) at time 200h for each of the corresponding model configurations. Red regions represent stem cells, green regions are TA cells, and blue regions are TD cells. 11.4(a),11.4(b): Equal affinity to the basal lamina is insufficient to mimic the OE lamination. Lamination is not reflected in the basal-apical distribution of cells. Cell's velocity toward the BL is $10\mu m/h$. The velocities are considered proportional to forces due to viscous drag. 11.4(c),11.4(d): Adhesion force between TD cells results in layers of stem and TA cells on both the apical and basal sides. The distribution plots show high concentration of TD cells around the center of the tissue. Stem and TA cells form clusters near the basal lamina and the apical side. Adhesion between TD cells results in velocity of $15\mu m/h$ (only between cells that are $10\mu m$ apart). Furthermore, all cell types have equal velocity of $10\mu m/h$ toward the BL. 11.4(e),11.4(f): Larger affinity force between stem/TA cells and the basal lamina, compared to TD cells and the BL, results in layered distribution of cells similar to the OE lamination. Cells' affinities toward the BL result in velocities of: $80\mu m/h$ for stem and TA cells, and $10\mu m/h$ for TD cells.

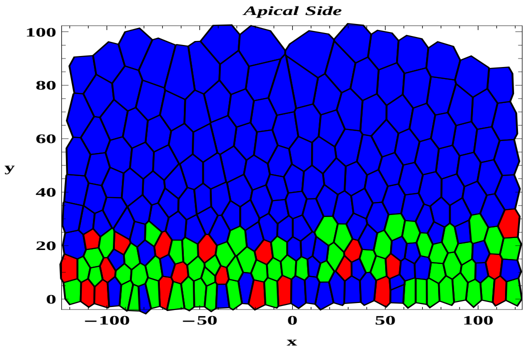
The parameter values of the non-spatial model are used here except for the constants S_0 and S_1. These constants prescribe the level of ϕ_0 and ϕ_1 in the deterministic model's steady state (and therefore the number of cells in steady state). Accordingly, S_0 and S_1 are reestimated in the spatial model so that the replication rate of both stem and TA cells match steady state levels. The concentration of ϕ_0 and ϕ_1 in steady state is approximated from a simulated constructed tissue with steady state number of cells which are arranged in layers (stem to TD layers).

The simulations shown here are initialized with two adjacent stem cells. Each stem cell is assumed to develop into an adjacent part of the tissue. Figure 11.5(a) and Figure 11.5(b) show the end result (at t=500h) of a simulation with two initial stem cells whereas Figure 11.5(c) shows the average number of cells over time.

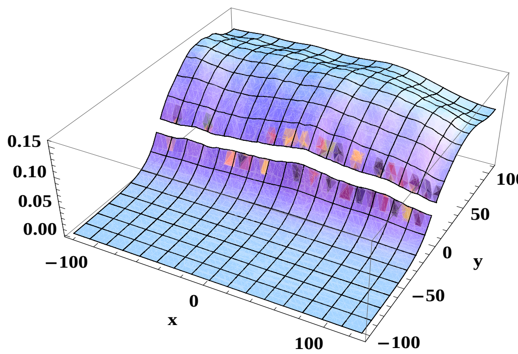
An initial stem cell proliferates into a monoclonal lineage tree (LT) of cells. We observe that in many simulations a single lineage tree survives while the other vanishes. A logical explanation is that the feedback strength is a function of the average size lineage tree. Therefore, a lineage tree which is below average size (due to random fluctuations) senses a signal for differentiation that is excessive for its size, while the opposite is true for above average size lineage trees. Figure 11.6 plots the average number of cells over time in each lineage tree according to their size. The plot shows that the largest LT gradually expands at the expense of the smaller LT that gets extinguished.

11.4 Conclusions

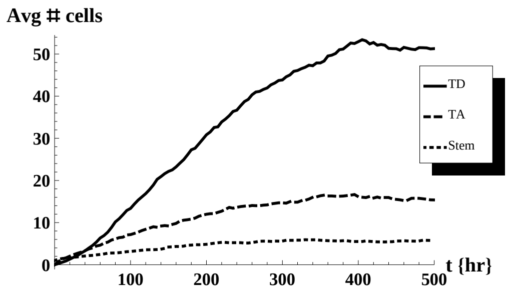
Here we discuss related work (Section 11.4.1 below), summarize our results, and discuss future directions (Section 11.4.2 below).



(a) Voronoi diagram of the simulated tissue



(b) Concentration of ϕ_1 over the 2D space



(c) Average number of cells per cell type

Figure 11.5: Spatial simulations recapitulate OE morphology. 11.5(a) and 11.5(b) are plots of the state of a single simulation at time 500hr. 11.5(a) A Voronoi diagram of the cellular structure. Red regions represent stem cells, green regions are TA cells, and blue regions are TD cells. 11.5(b) Concentration of ϕ_1 over the 2D space. The OE tissue is the $y > 0$ part of the grid whereas the $y < 0$ region is the stroma and the gap at $y = 0$ represents the BL. 11.5(c): The average number of cells of each type (stem, TA, and TD) over time. Simulations are initialized with two stem cells representing two lineage trees (LT). The concentration of ϕ_1 and ϕ_2 is initially zero. Diffusion constant (D) of ϕ_1 and ϕ_2 is 1 $\mu m^2 s^{-1}$. The decay constant (d) for both ϕ_0 and ϕ_1 is $5 * 10^{-4}\ s^{-1}$ in the OE, and $50 * 10^{-4}\ s^{-1}$ in the stroma region ($-100 \leq y < 0$). The average is taken over 60 simulations.

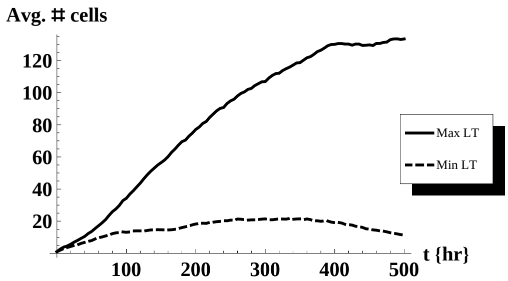


Figure 11.6: Stochastic simulations show that during tissue development one lineage tree (LT) dominates its region at the expense of neighboring LTs. The plot shows the average number of cells per LT ordered by size. The size of the largest LT increases whereas the other LT converges toward extinction state. The results are shown for the simulations of Figure 11.5.

11.4.1 Related work

Stochastic and deterministic non-spatial models of stem cell niches are described in [83]. A deterministic analysis in [83] shows the conditions for coexistence of stem cells from two different mutations (ie. different lineage trees). Our work shows how stochastic dynamics results in a competitive interplay between neighboring lineage trees as discussed in Section 11.3.2.

A directional cell sorting mechanism was proposed in [66] and [116] which uses the Cellular Potts Model (CPM) [54] for simulations. The advantage of the DG modeling framework and the current model is that no lattice discretization of space is required in order to represent cells and their dynamics, as there is in CPM models; this minimizes the number of degrees of freedom in the simulation and explicitly preserves translational and rotational invariance. Furthermore, the DG models represent a stochastic process independent of the simulation algorithm, while CPM models are defined in accordance with the CPM simulation algorithm of simulated annealing.

Models of the development and recovery of epidermis were presented in [95] and [57]. The models recreated the cell layering which is exhibited in the epidermis, similar to the lamination in the OE. However, both models are deterministic with no consideration of stochastic effects and the diffusion of extracellular molecules is approximated by direct cell-to-cell signaling.

11.4.2 Summary and future directions

In this chapter we present a computational framework for modeling stochastic biological systems which is based on the language of dynamical grammars (DG), and apply it to stem cell niches. A cell lineage system is modeled as a DG model such that the grammar rules for cell replication, differentiation, death and feedback signals follow observations of the development of the olfactory epithelium (OE). The full DG model includes a spatial representation of cells in 2D as cell growth and weak-spring forces between cell neighbors determine the spatial dynamics. In this DG model, a spatial grid is used for computing the numerical solution of secretion, diffusion and degradation of signaling molecules.

A simplified version of the spatial DG model was derived by eliminating the spatial coordinates. Such a non-spatial model is computationally faster to simulate and therefore important for the initial study of the behavior of the stochastic system and exploration of the parameter space. The importance of stochastic modeling for cell lineage systems is revealed in the non-spatial DG model as stochastic results diverge from those of the deterministic approximating models. Simulations of small cell populations have high probability of extinction and therefore the stochastic simulated system cannot maintain the positive steady state of the deterministic system.

The spatial simulations reveal the domination of a single lineage tree in each region as other lineage trees go extinct. The feedback signal diffuses in the tissue and therefore reduces the probability of proliferation in low cell density areas (whereas the opposite effect holds in high density areas). Larger than average lineage trees expand whereas their neighboring lineage trees contract and eventually go extinct. Of course mutations in stem cells would propagate within their lineage trees. So there may be evolutionary consequences of such stochastic lineage tree expansion within the tissue.

The current spatial DG is a simplified model of the tissue's expansion (or contraction) that assumes equal growth in both dimensions. Further study of the forces of expansion and contraction, consistent with the model's side boundary conditions, should consider the lineage trees' expansion and feedback signals' range. If the range of signaling molecules is larger than the range of cell mobility then the tissue may develop in an inhomogeneous way. In this case cells that belong to larger lineage trees will not be able to migrate to low cell density regions and therefore will accumulate as clusters, contradicting the homogeneity observed in the OE.

The spatial DG model was also used to explore the possible mechanisms for lamination in the OE. A possible mechanism is the increased adhesion between neuron cells, but such adhesion generates a central layer of TD (neurons) whereas the stem and TA form layers on both sides (basal and apical). This observation is in accord with differential adhesion theory [123] although no random movement of cells was postulated here. In order to recapitulate the structure in the OE, the stem and TA cells were assumed to have stronger (four times or more) affinity to the basal lamina than the TD cells. A further investigation using this model may unravel the importance of cell layers in reducing the impact of stochastic extinctions of lineage trees on the tissue's uniform thickness.

Chapter 12

Conclusions

12.1 Contributions of this thesis

This thesis has implemented a novel computational framework, the Stochastic Parameterized Grammar (SPG) for modeling stochastic processes. SPGs (defined in [93]) form a generalization of the probabilistic formulation of chemical reactions. SPG Grammar rules are reaction schemas where the reactants and products are parameterized terms. This formulation is shown to have wide expressive power which includes modeling complex chemical reactions and biological systems, and can be used for various machine learning applications.

The thesis defined and implemented some useful extensions of the basic SPG framework: constraints over objects' parameters, rules that depend on the absence of some objects and the incorporation of recursive calls to subgrammars. These extensions are useful for modeling complex operations on graph systems. For example, the nonexistence constraint is convenient when modeling an event that includes connecting two nodes in the graph that are not directly connected. The subgrammar calls can describe a dynamic number of operations in a single reaction. For example, nodes are connected to a variable number of nodes in a graph. Therefore, a node removal event encompasses a dynamic number of operations that can be encoded in a single subgrammar call. An additional application for subgrammar calls is multiscale modeling. Subgrammars may describe low level entities and events that occur in high frequency, such as chemical reactions, whereas calling grammars describe higher level entities, such as cells.

The Dynamical Grammars (DGs [93]) were implemented as a generalization of SPGs that includes continuous dynamics as may wise for example in spatial simulations. Ordinary and partial differential equations are declared as part of continuous-time grammar rules. Thus, DGs can model complex deterministic and continuous movement in space that results from interaction between objects. Examples of DG applications include: cells growth, cells movement due to spatial interactions, direct signaling between cells, and indirect signaling due to secretion and diffusion of molecules. The thesis presented a comprehensive DG model of the stem cell niche and neuronal population in the olfactory epithelium (OE). In the OE grammar, cells are represented as objects in a two dimensional space and signaling molecules are spatial concentrations.

SPGs and their generalization, DGs, are defined as stochastic processes which are decoupled from any specific simulation algorithm. The semantics (master equation) allows the derivation of multiple simulation and sampling algorithms. The thesis implemented a simulation algorithm for SPGs which was generalized to DGs, as described in [93]. Furthermore, exact and approximate inference algorithms were derived according to the master equation. The inference algorithms can be used to infer the most likely hidden realization of an SPG process and the most probable values of the hidden variables in an SPG model, given a set of observations. The approximate inference algorithm is a Markov Chain Monte Carlo (MCMC) method where new samples are generated by modifying the current multi-reactions path over a random time interval. The time interval size is a tunable parameter that is correlated to the samples acceptance rate and, therefore, may be optimized in order to maximize the MCMC convergence rate.

The thesis introduced an SPG model of spiral galaxies' structures. A learning algorithm, which is based on an approximation of the galaxy model, was presented and used for inferring spiral galaxy structures from images. This is the first probabilistic model that is used to automatically extract low-level features of spiral galaxies from images.

The OE model was developed in order to study the spatial interactions and possible mechanisms of spatial distribution of cells in the tissue. The two dimensional spatial model shows that the arrangement of stem and progenitor cells near the basal lamina can be maintained with differential affinity to the basal lamina. Furthermore, simulations of the OE model expose an interesting behavior of cell lineage trees that arises due to random fluctuations of cell populations. In every local region, a single cell lineage that originated from one stem cell dominates the other cell lineages that go extinct.

12.2 Future directions

The SPG and DG simulation algorithms are analogous to Gillespie's stochastic simulation algorithm (SSA) [52]. Substantial speedups over the straightforward SSA method have been gained by the approximate methods, *τ-leap* [53] and *R-leap* [10], and the recent exact method, *Exact R-leap* [94]. The *τ-leap* method approximates the number of reactions in a discrete time interval whereas the *R-leap* methods approximate the number reactions of each type out of a total predefined number of reactions in every simulation step. Such leaping methods can be incorporated in the SPG framework. However, such accelerated simulations (by leaping methods) of SPGs and DGs require handling objects' parameters (especially output parameters), and incorporating approximate integration techniques for the differential equations of DGs.

Simulations of the OE model, shown in Chapter 11, require extensive computing time and resources. A typical simulation of the OE model, that was shown, has approximately 10,000 reactions over hundreds of cells. For each reaction, the simulator integrates a set of two dimensional diffusion equations (PDEs) and ODEs. Some simulations require hours or even days of computation (an order of one minute per simulation step). *Plenum* uses multiple processors to compute multiple simulations in parallel. However, parallel computing can be used to speedup the computation of each simulation separately. General methods of parallel computation for numerical methods are discussed in [20, 37]. In case the DG model has spatial coordinates as the OE model, the differential equations and stochastic rate functions can be distributed between processors according to their spatial location. Movement of spatial objects may trigger some modifications of the objects distribution among processors during a simulation.

A possible direction for improving the accuracy of the SPG approximate sampling algorithm is to combine the Monte-Carlo method with an exact marginalization over some parameters. Such method of combining sampling with exact inference methods is known as Rao-Blackwellisation [22] which was used successfully for some Dynamic Bayesian Networks models [38]. In a Rao-Blackwellised inference algorithm for SPGs, the sampling part generates a multi-reaction path over internal objects, whereas exact marginalization calculates the probability of the observations given the set of generated internal objects. Exact marginalization is feasible only for some context-free SPGs, as was discussed in Chapter 7. Hence, the marginalization part should be confined to a context-free decomposition of the remaining possible reactions given the set of generated internal objects.

The EM algorithm, used for inference in spiral galaxies' images, is susceptible to local minima and initial conditions. A simulated annealing approach, as in Chapter 8, can avoid convergence to undesirable local minima. Moreover, the EM method optimizes a static model (as opposed to the dynamic grammar model), meaning that the model has a predefined number of objects and clusters (knots). The approximate inference algorithm of Chapter 8 samples from the SPG probability distribution and could be extended to handle unknown number of objects. An improved approximate inference algorithm (with the extensions discussed above) may provide a more flexible and accurate spiral galaxy inference scheme.

Plenum provides a basic interface for developing SPG models that is suitable for models with small scale number of rules and terms. However, even the models that were presented in this thesis can become increasingly complex and difficult to encode and manipulate as more constraints and spatial interactions are introduced. To increase the clarity of the model, the SPG syntax can be enhanced with object-oriented constructs, as the rule syntax that is part of the OE model in chapter 11. For future direction, a graphical user interface (GUI) can further improve the usability of the SPG framework. A related GUI design is implemented in Sigmoid [64], which provides a graphical representation of chemical reactions that are translated and simulated in Cellerator [119]. For SPGs and DGs, a GUI can provide a graphical representation of the interactions between different objects that will be translated to low level SPG and DG rules and simulated in *Plenum*.

Appendices

A Derivation of the Dynamical Grammar's simulation algorithm

This section derives Equation 4.3 which is used in the simulation scheme for grammars that include continuous rules. The simulation waiting time distribution (Equation 4.2) is an exponential over a sum of operators (i.e. e^{A+B}). Such exponential may be decomposed to a multiplication of exponentials (i.e. $e^{A}e^{B}$) only if the operators commute (i.e. $AB = BA$). In general, continuous rules operators and diagonal operators do not commute. In this case, the Trotter product formula [128, 118] can be used:

$$\exp\left(t(D + \sum_r O_r^c)\right) = \lim_{n\to\infty} \left(e^{tD/n} e^{tO_1^c/n} e^{tO_2^c/n} \cdots\right)^n \tag{A.1}$$

Let x denote a state space and $\rho(x)$ denote the diagonal element in D that corresponds to state x, i.e. the rate outflow in state x. The simulation waiting time distribution (Equation 4.2 can be expressed according to the Trotter product formula as:

$$\prod^n \left(\exp(-dt\rho(x)) \exp(-dt\frac{\partial}{\partial x}v(x))\right) \delta(x) \quad \text{, where} dt = \lim_{n\to\infty} t/n \tag{A.2}$$

The Taylor series expansion of the exponential is:

$$\exp(-dt\frac{\partial}{\partial x}v(x))\delta(x) = \left(\delta(x) - dt\frac{\partial v(x)\delta(x)}{\partial x} + (-dt)^2\frac{\partial^2 v(x)^2\delta(x)}{(\partial x)^2}/2! + ...\right) \quad \text{(A.3)}$$

The first derivative term may be reduced according to the fundamental property of the delta function:

$$\frac{\partial}{\partial x}v(x)\delta(x) = \int_{-\infty}^{\infty}\frac{\partial v(y)\delta(y)}{\partial y}\ \delta(y{-}x)dy = -\int_{-\infty}^{\infty}v(y)\delta(y)\frac{\partial}{\partial y}\delta(y{-}x)dy = v(0)\frac{\partial}{\partial x}\delta(x)$$

The second derivative term may be reduced similarly:

$$\begin{aligned}\frac{\partial^2 v(x)^2\delta(x)}{(\partial x)^2} &= \int_{-\infty}^{\infty}\frac{\partial^2 v(y)^2\delta(y)}{(\partial y)^2}\ \delta(y-x)dy = -\int_{-\infty}^{\infty}\frac{\partial v(y)^2\delta(y)}{\partial y}\ \frac{\partial}{\partial y}\delta(y-x)dy = \\ &= \int_{-\infty}^{\infty}v(y)^2\delta(y)\ \frac{\partial^2}{(\partial y)^2}\delta(y-x)dy = v(0)^2\ \frac{\partial^2}{(\partial x)^2}\delta(x) \qquad \text{(A.4)}\end{aligned}$$

We can derive equivalent expressions for the rest of the terms, therefore Equation A.3 results in:

$$\left(\delta(x) - dtv(0)\frac{\partial\delta(x)}{\partial x} + (dtv(0))^2\frac{\partial^2\delta(x)}{(\partial x)^2}/2! + ...\right) = \delta(x - dtv(0))$$

Equation A.2 after applying one factor of the multiplication:

$$\prod^{n-1}\left(\exp(-dt\rho(x))\exp(-dt\frac{\partial}{\partial x}v(x))\right)\exp(-\mathrm{dt}\rho(x))\delta(x - \mathrm{dt}v(0))$$

After applying all the factors, the result is:

$$\exp(-\int_0^t \rho(\hat{x}(\tau))d\tau)\delta(x - \hat{x}(t))$$

where $\hat{x}$ is the solution of the differential equation:

$$\frac{d\hat{x}}{dt} = v(\hat{x}) \quad , \hat{x}(0) = x_0$$

Bibliography

[1] Neural Information Processing Systems (NIPS) 2006 workshop on Dynamical Systems, Stochastic Processes and Bayesian Inference, December 2006. Whistler, BC, Canada. http://www.cs.ucl.ac.uk/staff/C.Archambeau/dsb.htm.

[2] Parameter Estimation in Systems Biology (PESB), March 2007. Manchester, UK. http://www.cs.manchester.ac.uk/ai/pesb07/.

[3] Uri Alon. *An Introduction to Systems Biology: Design Principles of Biological Circuits*. Chapman Hall, 2006.

[4] H. Amthor, G. Nicholas, I. McKinnell, CF Kemp, and M. Sharma. Follistatin complexes myostatin and antagonises myostatinmediated inhibition of myogenesis. *Dev Biol*, 270:19–30, 2004.

[5] Brigham Anderson, Andrew Moore, Andrew J. Connolly, and Robert Nichol. Fast nonlinear regression via eigenimages applied to galactic morphology. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 40–48. ACM Press, 2004.

[6] C. Andrieu, J. P. G. de Freitas, and A. Doucet. Reversible jump MCMC simulated annealing for neural networks. In *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence*, pages 11–18. Morgan Kaufmann, 2000.

[7] C. Andrieu, N. de Freitas, A. Doucet, and M. I. Jordan. An introduction to mcmc for machine learning. *Machine Learning*, 50:5–43, Jan 2003.

[8] Ana Valladares Enrique Flores Antonia Herrero, Alicia M. Muro-Pastor. Cellular differentiation and the ntca transcription factor in filamentous cyanobacteria. *FEMS Microbiology Reviews*, 28(4):469–487, 2004.

[9] K. B. Athreyea and P. E. Ney. *Branching Processes*. Springer-Verlag; Dover, 1972.

[10] A. Auger, P. Chatelain, and P. Koumoutsakos. R-leaping: Accelerating the stochastic simulation algorithm by reaction leaps. *Journal of Chemical Physics*, (125), 2006.

[11] J. Baker. Trainable grammars for speech recognition. *Speech communication papers presented at the 97th meeting of the Acoustical Society of America*, pages 547–550, 1979.

[12] GT Beemster and TI Baskin. Stunted plant 1 mediates effects of cytokinin, but not of auxin, on cell division and expansion in the root of arabidopsis. *Plant Physiol.*, 124:1718–27, Dec 2000.

[13] J.L. Bentley. Multidimensional binary search trees used for associative searching. *Communication of the ACM*, 18:1718–27, September 1975.

[14] Ashish Bhan and Eric Mjolsness. Static and dynamic models of biological networks. *Complexity*, 11:11–13, 2006.

[15] Yang J. Faeder-J.R. Blinov, M.L. and W.S. Hlavacek. Graph theory for rule-based modeling of biochemical networks. *Lect. Notes Comput. Sci.*, 4230:89–106, 2006.

[16] R. J. Boys, D. J. Wilkinson, and T. B.L. Kirkwood. Bayesian inference for a discretely observed stochastic kinetic model. *Statistics and Computing*, 18, June 2008.

[17] R. Bracewell. The impulse symbol. In *The Fourier Transform and Its Applications, 3rd ed*, pages 74–104. McGraw-Hill, 2000.

[18] Ola Bratteli and Derek W. Robinson. *Operator Algebras and Quantum Statistical Mechanics 2: Equilibrium States. Models in Quantum Statistical Mechanics*. Springer, 2003.

[19] Lee Brownston, Robert Farrell, Elaine Kant, and Nancy Martin. *Programming expert systems in OPS5: an introduction to rule-based programming*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1985.

[20] Are Magnus Bruaset and Aslak Tveito, editors. *Numerical Solution of Partial Differential Equations on Parallel Computers*. Springer, 2006.

[21] Anne L. Calof, Alexandre Bonnin, Candice Crocker, Shimako Kawauchi, Richard C. Murray, Jianyong Shou, and Hsiao-Huei Wu. Progenitor cells of the olfactory receptor neuron lineage. *Microscopy Research and Technique*, 58:176–188, 2002.

[22] G. Casella and C. P. Robert. Rao-blackwellisation of sampling schemes. *Biometrika*, 83(1):81–94, 1996.

[23] Tuncer Cebeci. *Convective Heat Transfer*. Springer, 2002.

[24] N. Chomsky. Three models for the description of language. *IRE Transactions Information Theory*, 2:113–124, 1956.

[25] N. Chomsky. On certain formal properties of grammars. *Information and Control*, 2:137–167, 1959.

[26] King-Wai Chu, Yuefan Deng, and John Reinitz. Parallel simulated annealing by mixing of states. *Journal of Computational Phys*, 148:646–662, 1999.

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

[28] J. Crank and P. Nicolson. A practical method for numerical evaluation of solutions of partial differential equations of the heat conduction type. *Proceedings of the Cambridge Philosophical Society*, 43:50–64, 1947.

[29] Mark de Berg, Marc van Kreveld, Mark Overmars, and Otfried Schwarzkopf. *Computational Geometry*. Springer-Verlag, 2000.

[30] JP de Winter, P. Dijke, CJ de Vries, TA van Achterberg, and H. Sugino. Follistatins neutralize activin bioactivity by inhibition of activin binding to its type II receptors. *Mol Cell Endocrinol*, 116:105–114, 1996.

[31] T. Dean and K. Kanazawa. A model for reasoning about persistence and causation. *Comp. Intelligence*, 5:142–150., 1989.

[32] R. Dechter. Bucket elimination: A unifying framework for processing hard and soft constraints. *Constraints: An International Journal*, (2):51–55, 1997.

[33] Rina Dechter. *Constraint Processing*. Morgan Kaufmann, 2003.

[34] Arthur Dempster, Nan Laird, and Donald Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society*, Series B:1–38, 1977.

[35] Luc Devroye. *Non-Uniform Random Variate Generation (see chapter 2)*. New York: Springer-Verlag, 1986.

[36] L. Dolan, K. Janmaat, V. Willemsen, P. Linstead, and S. Poethig. Cellular organisation of the arabidopsis thaliana root. *Development*, 119:71–84, 1993.

[37] J. Dongarra, G. Fox, K. Kennedy, L. Torczon, and W. Gropp. *The Sourcebook of Parallel Computing*. Morgan Kaufmann, 2002.

[38] Arnaud Doucet, Narido de Freitas, Kevin Murphy, and Stuart Russell. Raoblackwellised particle filtering for dynamic bayesian networks. In *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence*, pages 176–183, Stanford,CA, 2000.

[39] R. Durbin, R. Eddy, A. Krogh, and G. Mitchison. *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge University Press, 1999.

[40] F. Dyson. *Phys. Rev.*, 75:107–136, 1949.

[41] S. R. Eddy and R. Durbin. Rna sequence analysis using covariance models. *Nucleic Acids Research*, 22:2079–2088, 1994.

[42] Greg Eyink, Juan Restrepo, and Francis Alexander. A mean field approximation in data assimilation for nonlinear dynamics. *Physica*, D:347–368, 2004.

[43] P. Federl and P. Prusinkiewicz. Solving differential equations in developmental models of multicellular structures expressed using l 2212systems. *Proceedings of Computational Science*, pages 1–5, 2004.

[44] Kurt W. Fleischer. *A Multiple-Mechanism Developmental Model for Defining Self-Organizing Geometric Structures.* PhD thesis, California Institute of Technology, 1995.

[45] Charles Forgy. Rete: A fast algorithm for the many pattern/many object pattern match problem. *Artificial Intelligence*, (19):17–37, 1982.

[46] J. Friml, E. Benkov, I. Blilou, J. Wisniewska, and T. Hamann. Atpin4 mediates sinkdriven auxin gradients and root patterning in arabidopsis. *Cell*, 108:661–673, 2002.

[47] J. Friml, A. Vieten, M. Sauer, D. Weijers, H. Schwarz, T. Hamann, R. Offringa, and G. Jurgens. Efflux-dependent auxin gradients establish the apical basal axis of arabidopsis. *Nature*, 426:147–153, 2003.

[48] S. Geman and D. Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6:721–741, 1984.

[49] H.J. Genrich. Predicate/transition nets. *Advances in Petri nets : APN*, 1:208–247, 1986.

[50] Jean-Louis Giavitto and Olivier Michel. Mgs: a programming language for the transformations of topological collections. Technical Report 61-2001, May 2001.

[51] Gavin J. Gibson and Eric Renshaw. Estimating parameters in stochastic compartmental models using markov chain methods. *Math Med Biol*, 15:19–40, 1998.

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

[53] D.T. Gillespie. Approximate accelerated stochastic simulation of chemically reacting systems. *J. Phys.Chem.*, 115:1716–1733, 2001.

[54] J. A. Glazier and F. Graner. Simulation of the differential adhesion driven rearrangement of biological cells. *Phys. Rev.*, 47:2128–2154, 1993.

[55] Kimberly K. Gokoffski, Hsiao-Huei Wu, Crestina L. Beites, Joon Kim, Martin Matzuk, Arthur D. Lander, and Anne L. Calof. Cell stage-specific feedback regulation of neurogenesis and gliogenesis by activin and GDF11. *Submitted to Genes and Development*, Feb 2009.

[56] A. Golightly and D.J. Wilkinson. Bayesian inference for stochastic kinetic models using a diffusion approximation. *Biometrics*, 61:781–788, 2005.

[57] Niels Grabe and Karsten Neuber. A multicellular systems biology model predicts epidermal morphology, kinetics and Ca2+ flow. *Bioinformatics*, 21:3541–3547, 2005.

[58] Franois Graner and James A. Glazier. Simulation of biological cell sorting using a two-dimensional extended potts model. *Physical Review Letters*, 69:2013–2016, 1992.

[59] Peter J. Green. Reversible jump markov chain monte carlo computation and bayesian model determination. *Biometrica*, 82:711–732, 1995.

[60] Peter J. Haas. *Stochastic Petri Nets*. Springer, 2006.

[61] W. K. Hastings. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57:97–109, 1970.

[62] John E. Hopcroft and Jeffrey D. Ullman. *Introduction to automata theory, languages, and computation*. Reading, MA: Addison-Wesley, 1979.

[63] David V. Hutton. *Fundamentals Of Finite Element Analysis*. McGraw Hill, 2004.

[64] P. Baldi J. Cheng, L. Scharenbroich and E. Mjolsness. Sigmoid: Towards a generative, scalable, software infrastructure for pathway bioinformatics and systems biology. *IEEE Intelligent Systems*, 20(3):68–75, May/June 2005.

[65] Kurt Jensen. *Coloured Petri Nets, vols I, II, III*. Springer-Verlag, 1997.

[66] J. Kafer, P. Hogeweg, and F. M. Maree. Moving forward moving backward: directional sorting of chemotactic cells due to size and adhesion differences. *PLoS Computational Biology*, 2:429–454, 2006.

[67] S. Kirshner, I. Cadez, P. Smyth, and C. Kamath. Learning to classify galaxy shapes using em algorithm. *Neural Information Processing Systems*, pages 4658–4659, 2002.

[68] K.N. Kozlov and A.M. Samsonov. New migration scheme for parallel differential evolution. *5th International Conference on the Bioinformatics of Genome Regulation and Function (BGRS-2006)*, 2:141–144, 2006.

[69] EM Kramer and MJ Bennett. Auxin transport: a field in flux. *Trends in plant science*, 11:382–386, 2006.

[70] G. Soules L. E. Baum, T. Petrie and N. Weiss. A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains. *Ann. Math. Statist.*, 41(1):164–171, 1970.

[71] John E. Laird, Allen Newell, and Paul S. Rosenbloom. Soar: an architecture for general intelligence. *Artif. Intell.*, 33(1):1–64, 1987.

[72] J. Lam and J.-M. Delosme. An efficient simulated annealing schedule: Derivation. Technical Report 8816, 1988.

[73] Arthur D. Lander, Kimberly K. Gokoffski, Frederic Y.M. Wan, Qing Nie, and Anne L. Calof. Cell lineages and the logic of proliferative control. *PLoS Biology*, 7(1), 2008.

[74] Kenneth Lange. *Applied Probability. Section 9.6.* Springer-Verlag, 2004.

[75] K. Lari and S. J. Young. The estimation of stochastic context-free grammars using the inside-outside algorithm. *Computer Speech and Language*, 4:35–56, 1990.

[76] J. E. Lennard-Jones. Cohesion. *Proceedings of the Physical Society*, 43:461–482, 1931.

[77] Win-Cheong Lo, Ching-Shan Chou, Kimberly K. Gokoffski, Frederic Y.M. Wan, Arthur D. Lander, Anee L. Calof, and Qing Nie. Feedback regulation in multistage cell lineages. *Mathematical biosciences and engineering*, pages 407–417, 2008.

[78] Cash S. S. Poo M. Lowen, S. B. and M. C. Teich. Quantal neurotransmitter secretion rate exhibits fractal behavior. *Journal of Neuroscience*, (17):5666–5677, 1997.

[79] S. B. Lowen and M. C. Teich. Fractal renewal processes generate 1/f noise. *Phys. Rev.*, 47:992–1001, 1993.

[80] Steven Bradley Lowen and Malvin Carl Teich. *Fractal-Based Point Processes.* Wiley-Interscience, 2005.

[81] David MacKay. *Information Theory, Inference, and Learning Algorithms.* Cambridge University Press, 2003.

[82] J. B. MacQueen. Some methods for classification and analysis of multivariate observations. *Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability*, pages 407–417, 1967.

[83] M. Mangel and M. B. Bonsall. Phenotypic evolutionary models in stem cell biology: Replacement, quiescence, and variability. *PLoS ONE*, 3:407–417, 2008.

[84] Christopher D. Manning and Hinrich Schuetze. *Foundations of Statistical Natural Language Processing.* The MIT Press, 1999.

[85] Georgios Marnellos. *Gene Network Models Applied to Questions in Development and Evolution*. PhD thesis, Yale University, 1997.

[86] P. Mendes and D. Kell. Non-linear optimization of biochemical pathways: applications to metabolic engineering and parameter estimation. *Bioinformatics*, 14:869–883, 1998.

[87] B. Milch, B. Marthi, S. Russell, and D. Sontag. Blog: Probabilistic models with unknown objects. *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1352–1359, 2005.

[88] V. Mironova, V. A. Likhoshvai, N. A Omelyanchuk, S. I. Fadeev, G. Yosiphon, E. Mjolsness, and N. A. Kolchanov. Acroptal auxin transport mediates patterning along the root longitudinal axis. *In submission.*

[89] G. J. Mitchison and M. Wilcox. Rules governing cell division in anabaena. *Nature*, 239:110–111, 1972.

[90] E. Mjolsness, D. H. Sharp, and J. Reinitz. A connectionist model of development. *Journal of Theoretical Biology*, 152:429–454, 1991.

[91] Eric Mjolsness. The growth and development of some recent plant models: A viewpoint. *Journal of Plant Growth Regulation*, 25:270–277, December 2006.

[92] Eric Mjolsness, Charles D. Garrett, John Reinitz, and David H. Sharp. Modeling the connection between development and evolution: Preliminary report. *Evolution and Biocomputation: Computational Models of Evolution*, 899:103–122, 1995.

[93] Eric Mjolsness and Guy Yosiphon. Stochastic process semantics for dynamical grammars. *Annals of Mathematics and Artificial Intelligence*, 47:329–395, August 2006.

[94] Chatelain P Koumoutsakos P. Mjolsness E, Orendorff D. An exact accelerated stochastic simulation algorithm. *Technical Report UCI-ICS*, (08-09), 2009.

[95] D. Morel, R. Marcelpoil, and G. Brugal. A proliferation control network model: The simulation of two-dimensional epithelial homeostasis. *Acta Biotheoretica*, 49:219–234, 2001.

[96] GK. Muday. Auxins and tropisms. *J Plant Growth Regul.*, 20:226–243, 2001.

[97] Yves Fomekong Nanfack, Jaap A. Kaandorp, and Joke Blom. Efficient parameter estimation for spatio-temporal models of pattern formation: Case study of drosophila melanogaster. *Bioinformatics*, 23:3356–3363, September 2007.

[98] U. Nodelman, C.R. Shelton, and D. Koller. Continuous time bayesian networks. *Proceedings of the Eighteenth Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 378–387, 2002.

[99] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufmann Publishers, 1988.

[100] Chien Y. Peng, Luis C. Ho, Chris D. Impey, and Hans-Walter Rix. Detailed structural decomposition of galaxy images. *The Astronomical Journal*, 124:266–293, 7 2002.

[101] Carl A. Petri. *Kommunikation mit Automaten.* PhD thesis, University of Bonn, 1962.

[102] A. Phillips and L. Cardelli. A correct abstract machine for the stochastic pi-calculus. In *Concurrent Models in Molecular Biology*, August 2004.

[103] William H. Press, Brian P. Flannery, Saul A. Teukolsky, and William T. Vetterling. *Numerical Recipes in C: The Art of Scientific Computing.* Cambridge University Press, 1992.

[104] C. J. Preston. Spatial birth-and-death processes. *Bull. Int. Statist. Inst.*, 46:371–391, 1977.

[105] Przemyslaw Prusinkiewicz, Mark S. Hammel, and Eric Mjolsness. Animation of plant development. *SIGGRAPH '93 Conference Proceedings*, pages 3508–3515, 1993. Association for Computing Machinery.

[106] Przemyslaw Prusinkiewicz and Aristid Lindenmeyer. *The Algorithmic Beauty of Plants.* Springer-Verlag, 1990.

[107] N. Przulj, D. G. Corneil, and I. Jurisica. Modeling interactome: Scale-free or geometric? *Bioinformatics*, 20:3508–3515, 2004.

[108] R. Raffard, K. Amonlirdviman, J.D. Axelrod, and C. Tomlin. An adjoint-based parameter identifcation algorithm applied to planar cell polarity signaling. *Automatic Control, IEEE Transactions on*, 53:109–121, Jan 2008.

[109] Oliver Rbenknig. Imtek mathematica supplement IMS. http://www.imtek.uni-freiburg.de/simulation/mathematica/IMSweb/. Last accessed May 12, 2009.

[110] John Reinitz, Eric Mjolsness, and David H. Sharp. Model for cooperative control of positional information in drosophila by bcd and maternal hb. *Journal of Experimental Zoology*, 271:47–56, 1995.

[111] S. Reinker, R.M. Altman, and J. Timmer. Parameter estimation in stochastic biochemical reactions. *Systems Biology, IEE Proceedings*, 153:168–178, 2006.

[112] B. D. Ripley and A. I. Sutherland. Finding spiral structures in images of galaxies. *Philosophical Transactions: Physical Sciences and Engineering*, 332:477–485, 1990.

[113] H. Risken. *The Fokker-Planck Equation.* Springer, 1984.

[114] S. Sabatini, D. Beis, H. Wolkenfelt, J. Murfett, and T. Guilfoyle. An auxin-dependent distal organizer of pattern and polarity in the arabidopsis root. *Cell*, 99:463–472, 1999.

[115] S. Sanghai, P. Domingos, and D. Weld. Relational dynamic bayesian networks. *Journal of Artificial Intelligence Research*, 2005.

[116] Nicholas J. Savill and Paulien Hogeweg. Modelling morphogenesis: From single cells to crawling slugs. *Journal of Theoretical Biology*, 184:229–235, 1997.

[117] J. Schlecht, K. Barnard, E. Spriggs, and B. Pryor. Inferring grammar-based structure models from 3d microscopy data. *Computer Vision and Pattern Recognition. CVPR '07. IEEE Conference on*, 1:1–8, June 2007.

[118] L. S. Schulman. *Techniques and Applications of Path Integration.* Wiley, 1981.

[119] Bruce E. Shapiro, Andre Levchenko, Elliot M. Meyerowitz, Barbara J. Wold, and Eric D. Mjolsness. Cellerator: extending a computer algebra system to include biochemical arrows for signal transduction simulations. *Bioinformatics*, 19:677–678, 2003.

[120] Bruce E. Shapiro and Eric D. Mjolsness. Developmental simulations in Cellerator. In *Second International Conference on Systems Biology*, Pasadena, CA, 2001.

[121] D. L. Snyder and M. I. Miller. *Random Point Processes in Time and Space.* Wiley, 1991.

[122] Robert St-Aubin, Joel Friedman, and Alan K. Mackworth. A formal mathematical framework for modeling probabilistic hybrid systems. In *Proceedings of the Ninth Annual Conference on Artificial Intelligence and Mathematics*, pages 407–417, January 2006.

[123] MS Steinberg. Reconstruction of tissues by dissociated cells. some morphogenetic tissue movements and the sorting out of embryonic cells may have a common explanation. *Science*, 141:401–408, 1963.

[124] Heneghan C Lowen S. B. Ozaki T. Teich, M. C and E. Kaplan. Fractal character of the neural spike train in the visual system of the cat. *Journal of the Optical Society of America*, A(14):529–546, 1997.

[125] C. van den Berg, P. Weisbeek, and B. Scheres. Cell fate and cell differentiation status in the arabidopsis root. *Planta.*, 205:483–491, 1998.

[126] N. G. van Kampen. *Stochastic Processes in Physics and Chemistry.* North-Holland, 1981.

[127] Andrew J. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13(2):260–269, 1967.

[128] Eric W. Weisstein. Trotter product formula. *From MathWorld–A Wolfram Web Resource. http://mathworld.wolfram.com/TrotterProductFormula.html.*

[129] A.V. Werhli and D. Husmeier. Reconstructing gene regulatory networks with bayesian networks by combining expression data with multiple sources of prior knowledge. *Statistical Applications in Genetics and Molecular Biology*, 6, 2007.

[130] Pieter Wesseling. *An Introduction to Multigrid Methods*. Wiley, 1992.

[131] Darren J. Wilkinson. *Stochastic Modelling for Systems Biology*. Chapman Hall/CRC, 2006.

[132] William T. Vetterling and Brian P. Flannery William H. Press, Saul A. Teukolsky. *Numerical Recipes : The Art of Scientific Computing*. Cambridge University Press, 2007.

[133] Stephen Wolfram. *The Mathematica Book, Fourth Edition*. Cambridge University Press, 1999.

[134] H. Wu, S. Ivkovic, R. C. Murray, S. Jaramillo, K. M. Lyons, J. E. Johnson, and A. L. Calof. Autoregulation of neurogenesis by GDF11. *Neuron*, 37:197–207, 2003.

[135] J. Xu, H. Hofhuis, R. Heidstra, M. Sauer, and J. Friml. A molecular framework for plant regeneration. *Science*, 311:385–388, 2006.

[136] Guy Yosiphon and Eric Mjolsness. Plenum - a dynamical grammar interpreter/simulator, 2008. Software package written for Mathematica computer algebra system. Available at http://computableplant.ics.uci.edu/. Last accessed April 10, 2009.

[137] Guy Yosiphon and Eric Mjolsness. Towards the inference of stochastic biochemical network and parameterized grammar models. In N. D. Lawrence, M. Girolami, M. Rattray, and G. Sanguinetti, editors, *Learning and Inference in Computational Systems Biology*. MIT Press, 2009.

[138] Daniel H. Younger. Recognition and parsing of context-free languages in time n^3. *Information and Control*, 10(2):189–208, 1967.

[139] L. Zhang. *Dynamic Biological Signal Pathway Modeling and Parameter Estimation through Optimization*. PhD thesis, University of California, Irvine, 2008.

[140] S.C. Zhu and D. Mumford. A stochastic grammar of images. *Foundations and Trends in Computer Graphics and Vision*, 2(4):259–362, 2006.