Table of Contents

Preface .................................................................................................................................................. 1
    Ion Petre

Invited Presentation: Computational Methods in Systems Biology: Case Studies and Biological Insights ......................................................................................................................... 3
    Daniela Besozzi

Invited Presentation: Computational Methods for Metabolic Networks ........................................... 11
    Juho Rousu

Canonical Labelling of Site Graphs ....................................................................................................... 13
    Nicolas Oury, Michael Pedersen and Rasmus Petersen

Informal Presentations

Rule-Based Modeling of Polymerization Reactions ............................................................................. 29
    Thilo Krüger and Verena Wolf

Robustness Analysis of Stochastic Systems ........................................................................................ 33
    Luboš Brim, Milan Češka, Sven Dražan and David Šafránek

Mathematical modelling of the Platelet-Derived Growth Factor signalling pathway ..................... 35
    Andrzej Mizera, Jun Pang, Thomas Sauter and Panuwat Trairatphisan

Rule-based modelling as a platform for the analysis of synthetic self-assembled nano-systems ....... 37
    Abdulmelik Mohammed and Eugen Czeizler

An Algebraic Approach to Gene Assembly in Ciliates ...................................................................... 41
    Robert Brijder

Large-Scale Text Mining of Biomedical Literature ............................................................................. 43
    Filip Ginter
Preface

Ion Petre
˚Abo Akademi University and Turku Centre for Computer Science, Turku 20520, Finland
ipetre@abo.fi

The fourth international workshop on Computational Models for Cell Processes (CompMod 2013) took place on June 11, 2013 at the Åbo Akademi University, Turku, Finland, in conjunction with iFM 2013. The first edition of the workshop (2008) took place in Turku, Finland, in conjunction with Formal Methods 2008, the second edition (2009) took place in Eindhoven, the Netherlands, in conjunction with Formal Methods 2009, and the third one took place in Aachen, Germany, in conjunction with CONCUR 2013. This volume contains the final versions of all contributions accepted for presentation at the workshop.

The goal of the CompMod workshop series is to bring together researchers in Computer Science and Mathematics (both discrete and continuous), interested in the opportunities and the challenges of Systems Biology. The Program Committee of CompMod 2013 selected 3 papers for presentation at the workshop (one was withdrawn). In addition, we had two invited talks and five other informal presentations. We thank the PC members for their excellent work in making this selection. The CompMod 2013 Program Committee consisted of:

- Oana Andrei (University of Glasgow, UK)
- Luca Bortolussi (University of Trieste, Italy)
- Lubos Brim (Masaryk University, Czech Republic)
- Eugen Czeizler (Aalto University, Finland)
- Jerome Feret (ENS, France)
- Vashti Galpin (University of Edinburgh, UK)
- Russ Harmer (University of Paris 7, France)
- Monika Heiner (Brandenburg University of Technology Cottbus, Germany)
- Ina Koch (Johann Wolfgang Goethe-University Frankfurt am Main, Germany)
- Andrzej Mizera (University of Luxembourg)
- Ion Petre (Åbo Akademi University, Finland) chair
- Corrado Priami (Microsoft Research – University of Trento, Centre for Computational and Systems Biology, Italy)
- David Safranek (Masaryk University, Czech Republic)
- Angelo Troina (Universita di Torino)
- Erik de Vink (Eindhoven University of Technology, the Netherlands)
- Claudio Zandron (Universitá degli Studi di Milano-Bicocca, Italy)
The scientific program of the workshop spans an interesting mix of approaches to systems and even synthetic biology, encompassing several different modeling approaches, ranging from quantitative to qualitative techniques, from continuous to discrete mathematics, and from deterministic to stochastic methods. We thank our invited speakers

- Daniela Besozzi (Università degli Studi di Milano, Milano, Italy)
- Juho Rousu (Aalto University, Finland)

for accepting our invitation and for presenting some of their recent results at CompMod 2013. The technical contributions address the mathematical modeling of the PDGF signalling pathway, the canonical labelling of site graphs, rule-based modeling of polymerization reactions, rule-based modeling as a platform for the analysis of synthetic self-assembled nano-systems, robustness analysis of stochastic systems, an algebraic approach to gene assembly in ciliates, and large-scale text mining of biomedical literature.

We would also like to thank the editorial board of the *Electronic Proceedings in Theoretical Computer Science* (EPTCS) for accepting to publish these proceedings in their series.

*Ion Petre*

*Turku, Finland, May 2013*

*Workshop organizer and PC chair*
Computational Methods in Systems Biology: Case Studies and Biological Insights

Daniela Besozzi
Università degli Studi di Milano, Dipartimento di Informatica
Via Comelico 39, 20135 Milano, Italy
besozzi@di.unimi.it

Extended abstract of the invited contribution presented at the 4th International Workshop on Computational Models for Cell Processes (CompMod 2013), June 11, 2013, Turku, Finland.

Does Cell Biology Need Modeling?

In the last decades, mathematical modeling and computational methods have proven to be valuable approaches to achieve a better comprehension of the functioning of biological systems, which are complex (non-linear) dynamical systems evolving in time and in space [6, 15, 27]. Computational investigations can aid in bridging the gap between what is already known of a biological system of interest, and what is hardly comprehensible through the interpretation of laboratory work outcomes (that sometimes can be inconsistent with previous results) or by merely grounding on data-driven hypotheses (that often require the further design of complicated and onerous verification experiments). In some cases, mathematical models can simply represent a thought-provoking standpoint for further extended analysis [18], but most of the times they allow to derive useful predictions on the system functioning.

The choice of the proper modeling and computational approach is constrained by the nature of the biological system under examination and by the experimental data in hand. Before the mathematical model of a biological system can be defined, one needs to solve some preliminary issues.

- The first point that must be necessarily established right from the beginning is the purpose of the model, and what modeling will add to the experimental study of the biological system. This must follows from the identification of a complicated (biological) problem that requires, or might benefit from, a complementary analysis based on modeling and computational methods. To this aim, experimental data are necessary to define a plausible model, since they suggest possible assumptions about the mechanisms governing the system, but they are not sufficient: it is the scientific question that motivates the development of a model to represent the essential key for any meaningful modeling and computational workflow. In this context, a sentence taken from [6] is illustrative: “In fact, all modelers should be prepared to answer the question: What do you know now that you did not know before? If the answer is “that I was correct”, it is best to look elsewhere”.

- The second point concerns the scale of the model or, stated otherwise, the level of abstraction that has to be chosen to describe the components of the system and their mutual interactions, according to the biological question that one wishes to solve. This matter requires to identify the components of the system that bring about the most relevant functions, to make specific assumptions about
their interactions, and then to develop a corresponding fine-grained or coarse-grained model. In this context, if the emergent properties of the system need to be reconstructed – in the physiological state as well as in response to genetic, chemical or environmental perturbations – then the model should take into account all available knowledge on the component properties. Indeed, an accurate description of the fundamental molecular mechanisms governing the functioning of a system allows not only to derive its emergent properties, but also to connect such properties to the basic interactions among the system components, which altogether contribute to the behavior of the system as a whole [16]. As a matter of fact, over-simplification in modeling does not always represent a good choice, especially when fundamental multi-factorial relationships between the state of the system and its response to different perturbations might be erroneously neglected (e.g., in the modeling of complex diseases) [18].

• Once the first and the second point are settled, then the modeler has to select the most appropriate mathematical formalism. In some cases, this might be straightforward, but usually one has to argue the suitability of each of numerous dichotomic approaches that are found in the field of Computational Systems Biology (e.g., top-down vs. bottom-up; interaction-based vs. mechanism-based; quantitative vs. qualitative; static vs. dynamic; parameterized vs. non parameterized) [25]. In general, the mechanism-based, parameterized, quantitative and dynamic modeling approach is considered as the most likely candidate to achieve a system-level comprehension of biological systems, since it can lead to quantitative predictions of cellular dynamics (on the basis of the reconstructed temporal evolution of all molecular species occurring in the model) and to the formulation of experimentally verifiable hypotheses. Nonetheless, the high complexity intrinsic in biological systems and in their experimental analysis, together with the usual lack of quantitative parameters, represent a limit to the wide applicability of this modeling approach.

A thorough description of the above questions, and of the related applicable formal and computational methods, is clearly outside the scope of this extended abstract; general discussions and practical applications can be found in [1, 6, 9, 12, 15, 25, 26, 27, 29, 30].

In what follows, I briefly discuss the application of the mechanism-based modeling approach and the related computational methods, chosen according to the biological questions established for the study of two signal transduction pathways. In particular, I consider here state-dependent reaction-based models, where reactions describe molecular interactions based on the mass-action law; these models are fully parameterized from both the kinetic and the stoichiometric points of views, that is, (i) a kinetic constant is associated to each reaction, (ii) precise values of reactant and product molecules are specified for every species occurring in each reaction, and (iii) the initial state of the system is defined in terms of molecular amounts of all species.

This type of modeling approach presents some remarkable advantages with respect to other classical formalisms. First of all, reaction-based models are immediately understandable to experimental biologists, who are usually not familiar with mathematical formalisms, and therefore they can easily intervene in the discussion and in the development of the model itself. Secondly, they can be easily modified during further model refinements (e.g., after the model has been validated with laboratory experiments), in order to introduce new species or new reactions that were not included in the first formalization. Last but not least, the same model can be exploited to carry out both stochastic and deterministic simulations of the system dynamics, by simply transforming the reaction-based into a “generalized mass-action based”
model (formalized by means of ordinary differential equations), and by converting stochastic constants and molecule copy numbers into kinetic rates and concentrations, respectively [13, 31].

Dynamics simulations then allow to globally explore the behavior of the system, in both quantitative and qualitative ways, so that by modulating the model components (species, reactions, and their related parameters) it is possible to better understand complex biological phenomena whenever simple intuition might fail to solve the established biological questions, or where contradictory experimental data prevent to find lifelike and unambiguous answers.

A Bird’s Eye View on Two Case Studies

I discuss the investigation of two signal transduction pathways in yeast that have been a topic of my research in the last years, considered here as case studies to highlight the relevance of the crosstalk between computational scientists and biologists, and to show that in silico computational analysis can be reasonably considered as a complementary and effective integration tool to traditional wet approaches in Cell Biology.

The Ras/cAMP/PKA pathway

In *S. cerevisiae*, the Ras/cAMP/PKA signalling pathway regulates cell metabolism and cell cycle progression in response to nutritional sensing and stress conditions [28]. These processes are controlled through multiple feedback loops exerted by the protein kinase A (PKA) on a few pivotal components of the pathway, i.e., proteins Cdc25 and Ira (which regulate the GTPase Ras protein in a positive and negative manner, respectively) and two phosphodiesterases (which control the signal transduction termination through the degradation of cAMP). Both experimental and computational studies of this pathway are complicated by the complex interplay between these regulation mechanisms, making it hard to predict the behavior of the pathway in different growth conditions or in response to stress signals.

Previous experimental studies reported the observation of oscillations related to the Ras/cAMP/PKA pathway, though this was stated only in indirect ways, that is, by analyzing downstream processes as the nucleocytoplasmic shuttling of Msn2 [11, 17], a transcription factor that is under the control of PKA; other analysis then showed that the oscillations frequency of Msn2 between the nucleus and the cytoplasm can be influenced by PKA as well as by the two phosphodiesterases [5]. This periodicity can be ascribed to an oscillatory behavior of cAMP concentration and of PKA activity, though no direct measurements of the dynamics of these components were executed in vivo to verify their effective role.

In order to understand the role of these feedback controls, and to make predictions on the cellular conditions that are able to regulate the transition between stationary and oscillatory regimes, a computational analysis of the Ras/cAMP/PKA pathway was carried out, primarily focusing on the mechanisms that allow the emergence of oscillatory regimes in cAMP dynamics [4, 23]. The model of the Ras/cAMP/PKA pathway was initially developed according to the stochastic formulation of chemical kinetics [13], and then translated into a generalized mass-action based model [31], in order to compare the outcomes of stochastic and deterministic simulations. To this aim, specific computational tools had to be developed, in order to quantify the amplitude and frequency of cAMP oscillations in stochastic simulations. The choice of the stochastic approach was motivated by the low molecular amounts of some pivotal components of the pathway (i.e., Cdc25 and Ira proteins). The comparison between stochastic and deterministic analysis or of the kinetic values, which can span over several orders of magnitude), a feature that might induce stiffness and longer computational time to execute the simulations.
evidenced different quantitative and qualitative dynamics of cAMP under various conditions (see, e.g., Figure 6 and 13 in [4]), suggesting a functional role of the intrinsic noise occurring in this pathway.

In particular, the computational analysis indicated that stable oscillatory regimes of cAMP can be established when the negative feedback on Ira proteins is activated, and that this dynamics is regulated by the balance between the activities of Cdc25 and Ira proteins. In addition, the analysis highlighted that GTP and GDP pools, which concurrently regulate Ras proteins, are able to control the transition between steady states and oscillations of cAMP, and they also suggested a regulative role for the phosphodiesterases [4]. Overall, the model allowed to find out, in a detailed quantitative way, that the coupling between the feedback mechanisms and the molecular level of Ras modulators (Cdc25/Ira, GTP/GDP) is able to influence the oscillatory regimes of cAMP and PKA.

One of the biological insights achieved with the modeling of the Ras/cAMP/PKA pathway concerns the multi-level regulation carried out through different feedback mechanisms, which might represent a way to extend the regulatory span of the system in yeast cells, and might therefore act as a tuning mechanism for the control of the numerous downstream targets of PKA. For instance, the modeling suggested that a frequency modulation can be achieved through the perturbation of the ratio between the cellular amounts of Cdc25 and Ira proteins.

Despite the Ras/cAMP/PKA pathway has been extensively investigated in *S. cerevisiae*, time-course data on cAMP variations in yeast cells are still lacking, due to the difficulty of developing an accurate experimental protocol to carry out these measurements. Laboratory work is currently in progress to set up a FRET-sensor able to respond to cAMP levels in *S. cerevisiae*, which should allow to measure in vivo the changes in the level of cAMP in single cells [8]. This very promising protocol still needs to be optimized, in order to increase its stability and to reduce the noise intrinsic in the measurements, so that long term oscillations of cAMP in single yeast cells will finally become measurable. As a consequence, the novel experimental setup will allow to carry out a full validation of the model, and to conduct an in depth analysis of the response of the pathway to different nutritional and stress conditions.

### The Post-Replication Repair pathway

Post-Replication Repair (PRR) is a DNA-damage tolerance pathway involved in the recognition and bypass of the genome lesions induced by sunlight exposure and UV radiation [19, 24]. PRR acts through two different sub-pathways, that are activated by a post-translational modification of PCNA, a protein acting as a scaffold for the binding of several proteins involved in DNA replication, repair and cell cycle regulation. In *S. cerevisiae*, the experimental investigation of this complicated and not well characterized pathway determined that PCNA mono-ubiquitylation drives the activation of an error-prone DNA lesion-bypass mechanism (Translesion DNA Synthesis, TLS), while PCNA poly-ubiquitylation activates an error-free mechanism (Template Switching, TS) [32]. A major issue in the study of PRR consists in understanding how the dynamics of PCNA ubiquitylation might influence the choice between TLS and TS, or whether there exists a damage-related threshold able to regulate the crosstalk between these sub-pathways.

In order to dissect these aspects, we defined a reaction-based stochastic model of PRR [2]. The choice of the stochastic framework was motivated by the low molecular amounts of most species involved in PRR, and by the quite large noise that we evidenced in the experimental data when measuring the ratio between mono- and poly-ubiquitylated PCNA. The definition of the mathematical model of PRR benefited from a preliminary bioinformatic analysis based on 3D protein structure modeling, which served to confirm the actual spatio-temporal cascade of interactions between the proteins involved in PRR, which were not completely known; in addition, a successive computational phase based on parameter sweep
analysis and sensitivity analysis was used to test the reliability of the model parameterization.

The laboratory measurements of mono- vs. poly-ubiquitylated PCNA ratio were used as the wet read-out of the cellular response to acute UV irradiation, carried out at low and high doses in both wild-type and mutant yeast cells. To this aim, a specific experimental protocol had to be developed so that, differently from other previously devised methods, both mono- and poly-ubiquitylated PCNA isoforms could be detected on a single western blot; these measurements were then processed in order to compare the different kinds of experimental and computational results (that is, the ratios of modified PCNA derived in laboratory, and the molecular amounts of modified PCNA obtained from stochastic simulations).

The results of the continuous crosstalk between experimental and computational analysis suggested the existence of a UV dose threshold for the proper functioning of the PRR model, while at higher UV doses we came across an unexpected discrepancy between wet data and simulation outcomes. This inconsistency first led to several steps of model verification, and then to the formulation of novel biological hypotheses on the functioning of PRR. As a result, further laboratory experiments especially designed to validate the model predictions evidenced an overlapping effect of Nucleotide Excision Repair (the cellular pathway effectively responsible to clean the genome from UV lesions) on the dynamics of PCNA ubiquitylation in different phases of the cell cycle [2]. Altogether, these findings highlighted an intricate functional crosstalk between PRR and other processes controlling genome stability, establishing that PRR is less far characterized than previously thought, and opening new research perspectives on the study of DNA repair and tolerance systems.

**Conclusive Remarks**

In the investigation of biological systems, a negative result – either obtained from unmatching simulation outcomes, or due to unexpected experimental data – can be as important as a positive result, since it highlights a deficiency in the comprehension of the underlying molecular mechanisms. As a matter of fact, sometimes the most interesting part of the modeling game is not what the model allows to understand, but exactly what it cannot explain. Obviously, the model’s incapability necessarily requires a careful check to determine the source of the weaknesses, which can be related to some defects or incompleteness of the model, as well as to unforeseen bugs in the experiment setup or in the interpretation of the corresponding measurements.

An important problem, related to these aspects, is that modelers typically have insufficient knowledge of laboratory methodologies, and are usually not able to interpret the results of an experiment, to determine if literature data are fully reliable, or to settle the right questions that can drive the development of a good model. To overcome such limitations, only a continuous side-by-side work between modelers and biologists, as well as the iterative crosstalk between the computational and experimental outcomes they generate, can lead to a proper Systems Biology interdisciplinary workflow consisting in data-driven modeling and model-driven experiments, that can eventually turn into a better understanding of the functioning of the system under investigation.

Indeed, in order for a model to be useful to experimental biologists, it must be strictly linked to biological data and, possibly, rely to dedicated *ad hoc* measurements. To this aim, experimental biologists should be open to design new laboratory experiments and specific measurement methodologies that allow to identify the qualitative and, most important, the quantitative features of the system that are still lacking, while modelers should be prepared to develop novel mathematical and computational tools, that are able to deal with the established biological questions and the actual data in hand, instead of “forcing” the system to be described and analyzed with some favorite and handily applicable framework. The
Computational Methods in Systems Biology

awareness and fulfillment of these requisites is surely likely to increase our ability in comprehending the complexity of biological systems.

Concerning the computational analysis of biological systems, an important aspect related to the computational costs is worth to be mentioned here. In order to gain novel insights into the functioning of a biological system, under physiological or perturbed conditions, the application of many computational methods requires the execution of a large number of simulations, especially when one wants to derive statistically meaningful properties or to exploit tools based on parameter sweep analysis, sensitivity analysis, parameter estimation and reverse engineering of model topologies (see [1, 7] and references therein). These analyses usually result into very high computational costs when the simulations are executed on standard central processing units (CPUs), therefore demanding a shift to the use of parallel architectures (e.g., computer clusters) or high-performance processors. To this aim, the higher memory bandwidth and computational capability of graphics processing units (GPUs) have been increasingly exploited in the last years in the field of Bioinformatics and Computational Biology [10, 14, 22]. Besides the relevant reduction of computational costs and the possibility to run multiple analysis in parallel, general-purpose GPU computing benefits from cheap, diffused and highly efficient multi-core devices, and it provides an effective mean to sustainable computing since it drastically reduces the power consumption. Despite these advantages, scientific applications of GPUs risk to remain a niche for few specialists, since GPU-based programming substantially differs from CPU-based computing and therefore requires specific programming skill. In order to provide user-friendly computational methods to analyze the dynamics of biological systems, we started to develop GPU-powered tools to carry out stochastic [21], deterministic [20] and hybrid simulations [3] for mass-action based models. These methods have been successfully tested to execute sensitivity analysis, parameter sweep analysis and parameter estimation of the Ras/cAMP/PKA and PRR models, showing that the GPU implementation\(^2\) allows to gain a speedup ranging from 25× to 50× with respect to the corresponding CPU implementation\(^3\).

Finally, a sentence that perfectly resumes the concepts discussed above and epitomizes the hurdles and the beauty inherent to the computational analysis of biological systems, can serve here as my last remark: “Building a good model depends too much on intuition, on the rare abilities to ask the right question and to sense mathematical order behind messy facts, on tricky timing (not too early, when absence of data leaves a model unsubstantiated, not too late, when everything is clear without a model), and on hard, long thinking that makes modeling so painful, but also so much fun” [18].

References


\(^2\) Nvidia GeForce GTX 590, equipped with 1024 cores and a theoretical limit of 2.48 Tflops.

\(^3\) Note that 100× to 1500× speedup were also achieved with other reaction-based models of biological systems, according to the model dimension and to the given parameterization [20].


Computational Methods for Metabolic Networks

Juho Rousu
Department of Information and Computer Science, Aalto University, Finland

juho.rousu@aalto.fi

Metabolic networks provide a rich source of computational problems and applications including metabolic reconstruction, pathway search and flux analysis. The main thrust in the research field has so far been on single species models with metabolites appearing as atomic entities. In this talk I will discuss two less explored viewpoints on metabolic network models, namely atom-level representations, and simultaneous reconstruction of networks for multiple species. I will look at the computational challenges, algorithms, and modelling benefits of these viewpoints and present a case study using our new CoReCo framework that implements these viewpoints.
Canonical Labelling of Site Graphs

Nicolas Oury¹
School of Informatics¹
Edinburgh University
Edinburgh, Scotland

Michael Pedersen²,³
Department of Plant Sciences²
Cambridge University
Cambridge, UK

Rasmus Petersen³
Microsoft Research³
Cambridge, UK

We investigate algorithms for canonical labelling of site graphs, i.e. graphs in which edges bind vertices on sites with locally unique names. We first show that the problem of canonical labelling of site graphs reduces to the problem of canonical labelling of graphs with edge colourings. We then present two canonical labelling algorithms based on edge enumeration, and a third based on an extension of Hopcroft’s partition refinement algorithm. All run in quadratic worst case time individually. However, one of the edge enumeration algorithms runs in sub-quadratic time for graphs with “many” automorphisms, and the partition refinement algorithm runs in sub-quadratic time for graphs with “few” bisimulation equivalences. This suite of algorithms was chosen based on the expectation that graphs fall in one of those two categories. If that is the case, a combined algorithm runs in sub-quadratic worst case time. Whether this expectation is reasonable remains an interesting open problem.

1 Introduction

Graphs are widely used for modelling in biology. This paper focuses on graphs for modelling protein complexes: vertices correspond to proteins, and edges correspond to bindings. Moreover, vertices are labelled by protein names, and edges connect vertices on labelled sites, giving rise to a notion of site graphs. Importantly, site labels can be assumed to be unique within vertices, i.e. a protein can have at most one site of a given name. This uniqueness assumption introduces a level of rigidity which can be exploited in algorithms on site graphs. Rigidity is for example crucial for containing the computational complexity of one algorithm for stochastic simulation of rule-based models of biochemical signalling pathways [3].

In this paper we investigate how rigidity can be exploited in the design of efficient algorithms for canonical labelling of site graphs. Informally, a canonical labelling procedure must satisfy that the canonical labellings of two graphs are identical if and only if the two graphs are isomorphic. A graph isomorphism is understood in the usual sense of being an edge-preserving bijection, with the additional requirement that vertex and site labellings are also preserved. Hence two site graphs are isomorphic exactly when they represent protein complexes belonging to the same species. The graph isomorphism problem on general graphs is hard: it is not known to be solvable in polynomial time, but curiously is not known to be NP-complete either even though it is in NP [10]. The canonical labelling problem clearly reduces to the graph isomorphism problem, but there is no clear reduction the other way. In this sense the canonical labelling problem is harder than the graph isomorphism problem.

Efficient canonical labelling of site graphs has an application in a second algorithm for stochastic simulation of rule-based languages [7]. This algorithm at frequent intervals determines isomorphism of a site graph, representing a newly created species, with a potentially large number of site graphs representing all other species in the system at a given point in time. The algorithm can hence compute a canonical labelling when a new species is first created, and subsequently determine isomorphism quickly by checking equality between this and existing canonical labellings.

Ion Petre (Ed.): Fourth International Workshop on
Computational Models for Cell Processes (CompMod 2013)
We assume in this paper that the number of site labels and the degree of vertices in site graphs are bounded, i.e. they do not grow asymptotically with the number of vertices, which indeed appears to be the case in the biological setting. An algorithm for site graph isomorphism is presented in [9] and has a worst-case time complexity of $O(|V|^2)$, where $V$ is the set of vertices. In this paper we exploit similar ideas, based on site uniqueness, in the design of new algorithms for canonical labelling, which also have a worst-case time complexity of $O(|V|^2)$. We furthermore characterise the graphs for which lower complexity bounds are possible: if the bisimulation equivalence classes are “small”, meaning $O(1)$, or if the automorphism classes are “large”, meaning $O(|V|)$, then an $O(|V| \cdot \log|V|)$ time complexity can be achieved in worst and average case, respectively.

Site graphs can be encoded in a simpler, more standard notion of graphs with edge-colourings. We formally define these graphs in Section 2, along with the notion of canonical labelling and other preliminaries. In Section 3 we specify two canonical labelling algorithms based on ordered edge enumerations, and in Section 4 we show how these algorithms can be improved using partition refinement. We conclude in Section 5.

2 Preliminaries

2.1 Notation

We write $X \rightarrow Y$ (respectively $X \rightarrow_{bij} Y$) for the set of total functions (respectively total bijective functions) from $X$ to $Y$. We use the standard notation $\prod x \in X. T(x)$ for dependent products, i.e. the set of total functions which map an $x \in X$ to some $y \in T(x)$. We write $\text{Dom}(f)$ and $\text{Im}(f)$ for the domain of definition and image of $f$, respectively, and $f \downarrow Z$ for the restriction of $f$ to the domain $Z$. We view functions as sets of pairs $(x \mapsto y)$ when notationally convenient. We use standard notation for finite multisets, and use the brackets $\{ \cdot \}$ for multiset comprehension. We write $\mathcal{T}(X)$ for the set of total orders on $X$. We write $X^*$ for the Kleene closure of $X$, i.e. the set of finite strings over the symbols in $X$. A linearly ordered set $(X, \prec)$ we assume the lexicographic extension of $\prec$ to pairs and lists over $X$. Given a partially ordered set $(X, \preceq)$ we write $\text{min}_\preceq(X)$ for the set of minimal elements of $X$ under $\preceq$, and if the ordering is total we identify $\text{min}_\prec(X)$ with its unique least element. Given an equivalence relation $\rho$ on a set $X$ and $x \in X$, we write $[x]_\rho$ for the equivalence class of $x$ under $\rho$, and we write $X'/\rho$ for the partition of $X' \subseteq X$ under $\rho$. Finally, given a list $x$, we write $x.i$ for the $i$th element of $x$ starting from 1.

2.2 Site Graphs and Coloured Graphs

A site graph is a multi-graph with vertices labelled by protein names and edges which connect vertices on sites labelled by site names. An example is shown in Figure 2.1a where protein names are indicated by colours, and a formal definition is given in Appendix A, Definition 16. From an algorithmic perspective, the key property of site graphs is their rigidity: a vertex can have at most one site of a given name, so a site name uniquely identifies an adjacent edge. This rigidity property can be captured by the simpler, more standard notion of directed graphs with edge-colourings. An example is shown in Figure 2.1b, and the formal definition follows below.

**Definition 1.** Let $(\Sigma, \prec)$ be a given linearly ordered set of edge colours. Then $\mathcal{CG}$ is the set of all *edge coloured graphs* $G = (V, E, \phi)$ where:

- $V = \{1, \ldots, k\}$ is a set of vertices.
- $E \subseteq V \times V$ is a set of directed edges.
Figure 2.1: An example of a site graph (a), its encoding as a coloured graph (b), and the formal definition of colours (c) following the encoding in Appendix A with the site name ordering \( a \leq_s b \leq_s c \leq_s \alpha \leq_s \beta \). The direction of e.g. the edge \((2, 3)\) is determined by \( a \leq_s b \) and \((a, b)\) being the minimum pair of connected sites between vertices 2 and 3.

- \( \phi : E \to \Sigma \) is an edge colouring satisfying for all \( v \in V \) that \( \phi \downarrow \{ (v, v') \in E \} \) and \( \phi \downarrow \{ (v', v) \in E \} \) are injective.

Given a coloured graph \( G \), we write \( V_G, E_G \) and \( \phi_G \) for the vertices, edges and edge colouring of \( G \), respectively. We write \( \text{Adj}(v) \), respectively \( \text{Adj}^{-1}(v) \), for the set of all outgoing, respectively incoming, edges adjacent to \( v \). We measure the size of graphs as the sum of the number of edges and vertices, i.e. \( |G| \overset{\Delta}{=} |V_G| + |E_G| \). Since we assume the degree of vertices to be bounded, we have that \( O(|G|) = O(|V_G|) + O(|E_G|) \).

The condition on edge colours states that a vertex can have at most one incident edge of a given colour, thus capturing the rigidity property.

\[ \delta_v = \{ (x_1, x_2) \} \quad \text{and} \quad \delta_{(a, b)} = \{(\alpha, \beta)\} \]

### 2.3 Isomorphism and Canonical Labelling

Our notion of coloured graph isomorphism is standard. In addition to preserving edges, isomorphisms must preserve edge colours.

**Definition 2.** Let \( G \) and \( G' \) be coloured graphs. An isomorphism is a bijective function \( \sigma : V_G \to V_{G'} \) satisfying that \( \sigma(G) = G' \), i.e.:

- \( \forall v_1, v_2 \in V_G. \{ (v, v_2) \in E_G \iff (\sigma(v_1), \sigma(v_2)) \in E_{G'} \} \).
- \( \forall e \in E_G. \phi_G(e) = \phi_{G'}(\sigma(e)) \).

Furthermore define \( G \simeq G' \) (\( G \) and \( G' \) are isomorphic) iff there exists an \( \sigma \) relating \( G \) and \( G' \), and define \( (G, v) \simeq (G', v') \) (or simply \( v \simeq v' \)) iff there exists an \( \sigma \) s.t. \( \sigma(v) = v' \). We denote by \( I(G, G') \) the set of isomorphisms from \( G \) to \( G' \). An isomorphism from \( G \) to itself is called an automorphism.

Since all graphs of the same size have the same vertices, all isomorphisms are in fact automorphisms. We next define our notion of canonical labelling. Having vertices given by integers allows the canonical labelling of a graph to be a graph itself.
Definition 3. A canonical labeller is a function \( L : \prod G \in CG . [G] \cong \) satisfying for all \( G, G' \in CG \) that \( L(G) = L(G') \iff G \cong G' \). We say that \( L(G) \) is a canonical labelling of \( G \), and that \( G \) is canonical if \( L(G) = G \).

3 Edge Enumeration Algorithms

This section introduces two algorithms based on enumeration of edges and the comparison of these enumerations.

3.1 Edge Enumerators

The rigidity property of coloured graphs, together with the linear ordering of colours, means that any given initial node uniquely identifies an enumeration of all the edges in a graph. Such an enumeration can be obtained by traversing the graph from the initial node while always following edges according to the given linear ordering on colours. There are many possible enumeration procedures, so we generalise the notion of an edge enumerator as follows.

Definition 4. An edge enumerator is a function of the form \( \eta : \prod G \in CG . V_G \rightarrow \mathcal{T}(E_G) \times (V_G \rightarrow_{bij} V_G) \) satisfying for all \( G, G' \) and \( v \in V_G, v' \in V_{G'} \) with \( v \cong v' \), \( \eta(G,v) = (<\cdot , \alpha) \) and \( \eta(G',v') = (<\cdot , \alpha') \) that \( \alpha(G, <\cdot) = \alpha'(G', <\cdot') \).

An edge enumerator must hence produce a linear ordering of edges and an alpha-conversion, with the key property that the alpha-converted graph is invariant under isomorphism. A more direct definition is possible, but we require the alpha-conversion to be given explicitly by the enumerator for use in subsequent algorithms. We get the following directly from the definition of edge enumerators.

Lemma 5. Let \( \eta \) be an edge enumerator and let \( \eta(G,v) = (<\cdot, \alpha) \) and \( \eta(G',v') = (<\cdot, \alpha') \) for some \( G, G' \) and \( v, v' \). If \( \alpha(G) = \alpha(G') \) then \( (G,v) \cong (G',v') \).

Algorithm 1 implements an edge enumerator. It essentially carries out a breadth first search (BFS) on the underlying undirected graph: edges are explored in order of colour, but with out-edges arbitrarily explored before in-edges (lines 7-9). The inner loop checks whether an edge has been encountered previously (line 12) in order to avoid an edge being enumerated twice. The generated alpha-conversion renames vertices by their order of discovery by the algorithm (line 17).

Proposition 6. Algorithm 1 computes an edge enumerator.

The intuition of the proof is that the control flow of the algorithm does not depend on vertex identity, and hence the output should indeed be invariant under automorphism. The full proof is by induction in the number of inner loop iterations. For the complexity analysis observe that the algorithm is essentially a BFS which hence runs in \( O(|V_G| + |E_G|) \) time, which by assumption is \( O(|V_G|) \) in our case. The extensions do not affect this complexity bound, assuming that the checks for containment of elements in Visited and Enum are \( O(1) \); this is possible using hash map implementations.

3.2 A Pair-Wise Canonical Labelling Algorithm

Alpha-converted edge enumerations can be ordered lexicographically based on edge identity firstly, and on edge colours secondly. This ordering is defined formally as follows.

Definition 7. Define a linear order \( \sqsubseteq \) on coloured edges as \( (e, c) \sqsubseteq (e', c') \) iff \( e < e' \lor (e = e' \land c < c') \). We extend the order to pairs \( (G, e) \sqsubseteq (G', e') \) iff \( (e, \phi_G(e)) \sqsubseteq (e', \phi_{G'}(e')) \). Finally, we assume the lexicographic extension of \( \sqsubseteq \) to pairs \( (G, <\cdot), (G', <\cdot') \) of edge-ordered graphs.
Algorithm 1: An edge enumeration algorithm. We assume given an operator $\text{Sort}_<(X)$ which sorts the set $X$ according to a linear ordering $\prec$. The operators :: and @ are list cons and append, respectively.

```
input : a graph $G$ and a start vertex $v \in V_G$
output: an enumeration of the edges in $G$ from $v$

Enum ← [] /* a list of enumerated edges */
$\alpha$ ← $\{(v \rightarrow 1)\}$ /* a vertex renaming identifying order of encounter */
$Q$ ← Queue($v$) /* a queue initialised with $v$ */
Visited ← $\{v\}$ /* a set of visited vertices */

while $Q$ is not empty do
    $v$ ← Dequeue($Q$)
    outEdges ← $\text{Sort}_<(\text{Adj}(v))$
    inEdges ← $\text{Sort}_<(\text{Adj}^{-1}(v))$
    edges ← outEdges@inEdges
    for $i = 1$ to $|\text{edges}|$ do
        $(v_1, v_2) \leftarrow \text{edges}.i$
        if $(v_1, v_2)$ does not occur in Enum then
            Enum ← $(v_1, v_2) :: \text{Enum}$
            $v_{\text{new}}$ ← $v_1$ if $v = v_2$ and $v_2$ if $v = v_1$
            if $v_{\text{new}} \notin \text{Visited}$ then
                Visited ← Visited $\cup \{v_{\text{new}}\}$
                $\alpha$ ← $\alpha \cup \{(v_{\text{new}} \mapsto |\text{Visited}|)\}$
                Enqueue($Q$, $v_{\text{new}}$)
        end
    end
end
return (Enum, $\alpha$)
```

The ordering is used for canonical labelling by Algorithm 2, which essentially finds the edge enumeration from each vertex in the input graph, picks the smallest after alpha-conversion, and applies the associated alpha-conversion to the graph. Whenever two alpha-converted enumerations are identical (line 11), the source vertices are isomorphic by definition of edge enumerators. The isomorphism is then computed (line 12) and the set of pending vertices is filtered so that it contains at most one element from each pair of isomorphic vertices (lines 13-20). If both elements of a pair of isomorphic vertices are in the set of pending vertices, one element is removed from the pending set (lines 16-17); the least element under $\prec$ is chosen arbitrarily, which ensures that the other element does not get removed at a later iteration. If just one element of the pair is in the pending set (lines 18-19), the other element must have been visited earlier, and so the present pending element can be safely removed.

**Proposition 8.** Algorithm 2 is a canonical labeller.

The worst-case time complexity of Algorithm 2 is $O(|V_G|^2)$, namely when no vertices are isomorphic. If all nodes in the input graph are isomorphic, the number of pending vertices is halved at every iteration of the outer loop and hence the complexity is $O(|V_G| \cdot \log(|V_G|))$. More generally, this bound also holds in the average case if the largest automorphism equivalence class has size $O(|V_G|)$. There are several ways of improving the algorithm. One way is to further exploit automorphisms. When new automorphisms are found, these may generate additional automorphisms through composition with existing ones. In addition
Algorithm 2: A pair-wise canonical labelling algorithm.

**input**: a graph \( G \) and an edge enumerator \( \eta \)

**output**: a canonical labelling of \( G \)

1. \( V_{\text{pending}} \leftarrow V_G \) /* vertices yet to be enumerated */
2. \( \langle \min, \alpha \rangle \leftarrow \eta(G, v) \)
3. \( \alpha_{\min} \leftarrow \alpha \) /* the least alpha-conversion */
4. while \( V_{\text{pending}} \neq \emptyset \) do
5. \( v \leftarrow \text{any } v \in V_{\text{pending}} \)
6. \( V_{\text{pending}} \leftarrow V_{\text{pending}} \setminus v \)
7. if \( \langle \min = \alpha_{\min} = \text{null} \rangle \) or \( \alpha(G, \langle \rangle) \sqsubseteq \alpha_{\min}(G, \langle \min \rangle) \) then
8. \( \langle \min \leftarrow \langle \rangle \)
9. \( \alpha_{\min} \leftarrow \alpha \)
10. else if \( \alpha(G, \langle \rangle) = \alpha_{\min}(G, \langle \min \rangle) \) then
11. \( \alpha' \leftarrow \alpha^{-1} \circ \alpha \) /* find the automorphism */
12. \( V'_{\text{pending}} \leftarrow V_{\text{pending}} \) /* keep a copy of pending vertices */
13. for \( v' \in \text{Dom}(\alpha) \) do
14. /* discard isomorphic vertices from pending */
15. if \( \{v, \alpha(v)\} \subseteq V_{\text{pending}} \) then
16. \( V'_{\text{pending}} \leftarrow V_{\text{pending}} \setminus \text{Min}_{\langle \rangle} \{v, \alpha(v)\} \)
17. else
18. \( V'_{\text{pending}} \leftarrow V_{\text{pending}} \setminus \{v, \alpha(v)\} \)
19. \( V_{\text{pending}} \leftarrow V'_{\text{pending}} \)
20. return \( \alpha_{\min}(G) \)

To removing elements of the pending set, automorphisms could also be exploited by only considering the automorphism quotient graph in subsequent iterations. Another way of improving the algorithm is to compute edge enumerations lazily, up until the point where they can be distinguished from the current minimum. Neither of the above improvements, however, change the worst case complexity characteristics of the algorithm.

### 3.3 A Parallel Canonical Labelling Algorithm

The idea of lazily enumerating edges can be taken a step further with a second algorithm which enumerates edges from all nodes in parallel: at each iteration, a single edge is emitted by each enumeration. Following standard notions of lazy functions, we formally define the notion of a lazy edge enumerator below, where the singleton set \( \{ \ast \} \) corresponds to the unit type.

**Definition 9.** For each coloured graph \( G \) and \( i \in \{0 \ldots |E_G| - 1\} \), define the set of functions \( T_i(G) \overset{\Delta}{=} \{ \ast \} \rightarrow E_G \times V_G \times V_G \times T_{i+1} \) with \( T_{[E_G]} \overset{\Delta}{=} \mathcal{G} \). A lazy edge enumerator implementing a given enumerator \( \eta \) is then a function \( \eta_L : \prod G \in \mathcal{G}.V_G \rightarrow T_0(G) \) satisfying for any \( G \) and \( v \in V_G \) with \( \eta(G, v) = (\langle \rangle, \alpha) \) that \( (E_G, \langle \rangle).i = e_i, \alpha(e_i) = (v_i, v'_i) \) and \( \eta_L|_{E_G} \overset{\Delta}{=} \alpha(G) \) where \( (e_i, v_i, v'_i, \eta_{L_i}) \overset{\Delta}{=} \eta_{L_{i-1}}(\ast) \) for \( i \in \{1 \ldots |E_G|\} \).
and \( \eta_{L_0} \triangleq \eta_{L}(G,v) \).

Hence a lazy edge enumerator produces, given a graph and a start vertex, a function which can be applied to yield an edge, an alpha-conversion of the edge (i.e. two vertices) and a continuation which in turn can be applied in a similar fashion; the final function thus applied yields an alpha-converted graph for notational convenience below. A lazy version of the BFS edge enumerator in Algorithm 1 can be implemented in a straightforward manner in a functional language. This does not affect the complexity characteristics of the algorithm, i.e. it still runs in \( O(\lvert V_G \rvert) \) time.

Algorithm 3 computes canonical labellings using lazy edge enumerators. It first initialises a set of enumerators from each vertex in the input graph (line 1). It then enters a loop in which enumerators are gradually filtered out, terminating when the remaining enumerators complete with an alpha-converted graph. At termination, all remaining enumerators will have started from isomorphic vertices, and hence they all evaluate in their last step to the same alpha-converted graph. At each iteration, each enumerator takes a step, yielding the next version of itself and an alpha-converted coloured edge; the result is stored as a mapping from the former to the latter (line 3). A multiset of alpha-converted, coloured edges is then constructed for the purpose of counting the number of copies of each alpha-converted coloured edge (line 4). The alpha-converted coloured edges with the smallest multiplicity are selected, and of these the least under \( \subseteq \) is selected (lines 5-6). Only the enumerators which yielded this selected edge are retained in the new set of pending edge enumerators (line 7). Note that further discrimination according to connectivity with other enumerators would be possible; for example, the vertices which are sources of enumerators eliminated in step \( n \) could be distinguished from those which are sources of enumerators eliminated in step \( m \neq n \). We have omitted this for simplicity.

**Algorithm 3**: A parallel canonical labelling algorithm.

```
input : a graph \( G \) and a lazy edge enumerator \( \eta_L \)
output: a canonical labelling of \( G \)
1 Pending \( \leftarrow \{ \eta_L(G,v) \mid v \in V_G \} \)
2 while Pending \( \not\subseteq \emptyset \) do
3     StepMap \( \leftarrow \{ (\eta_L', (v,v'), \phi_G(e)) \mid (e,v,v', \eta_L') = \eta_L(*) \land \eta_L \in \text{Pending} \} \)
4     Steps \( \leftarrow \{ \text{StepMap}(\eta_L') \mid \eta_L' \in \text{Dom}(\text{StepMap}) \} \)
5     SmallestMult \( \leftarrow \min \{ \text{Steps}(e,c) \mid (e,c) \in \text{Steps} \} \)
6     LeastEdge \( \leftarrow \min \{ (e,c) \in \text{Steps} \mid \text{Steps}(e,c) = \text{SmallestMult} \} \)
7     Pending \( \leftarrow \{ \eta_L' \in \text{Dom}(\text{StepMap}) \mid \text{StepMap}(\eta_L') = \text{LeastEdge} \} \)
8 return the one member of Pending
```

**Proposition 10.** Algorithm 3 is a canonical labeller.

The initialisation in line 1 runs in \( O(\lvert V_G \rvert) \) time. The loop always requires \( |E_G| = O(\lvert V_G \rvert) \) iterations. In the worst case where all vertices are isomorphic, each line within the loop requires \( O(\lvert V_G \rvert) \) time, so the worst case complexity is \( O(\lvert V_G \rvert^2) \). Hence in the worst case there is no asymptotic improvement over Algorithm 2. In practice, however, Algorithm 3 is likely to perform significantly better.

The key question is how Algorithm 3 behaves in cases where there are few automorphisms, i.e. when the number of automorphisms is sub-linear in the number of vertices. One can hypothesise that asymmetry is then discovered sufficiently early to yield an \( O(\lvert V_G \rvert \cdot \log(\lvert V_G \rvert)) \) time complexity. If so, an overall \( O(\lvert V_G \rvert \cdot \log(\lvert V_G \rvert)) \) algorithm is obtained by running the pairwise and the parallel algorithms simultaneously, terminating when the first of the two algorithms terminates. However, this question
remains open. Therefore also the question of whether a sub-quadratic time complexity bound exists in
the general case remains open.

4 A Partition Refinement Algorithm

The vertex set of a coloured graph can be partitioned based on “local views”: vertices with the same
colours of incident edges are considered equivalent and are hence included in the same equivalence class
of the partition. If two vertices are in different classes, they are clearly not isomorphic. The partition can
then be refined iteratively: if some vertices in a class \( P \) have \( c \)-coloured edges to vertices in a class \( Q \)
while others do not, \( P \) is split into two subclasses accordingly. This partition refinement process can be
repeated until no classes have any remaining such diverging edges. An efficient algorithm for partition
refinement was given in 1971 by Hopcroft [5]. The original work was in the context of deterministic finite
automata (DFA), where partition refinement of DFA states gives rise to a notion of language equivalence,
thus facilitating minimisation of the DFA. Hopcroft’s algorithm runs in \( O(n \cdot \log(n)) \) where \( n \) is the
number of states of the input DFA.

We show in the next subsection how our coloured graphs can be viewed as DFA, thus enabling the
application of Hopcroft’s partition refinement algorithm. The literature does provide generalisations of
Hopcroft’s algorithm to other structures, including general labelled graphs [2] where a vertex can have
multiple incident edges with the same colour. However, we adopt Hopcroft’s DFA algorithm as this
remains the simplest for our purposes. Furthermore, the algorithm has been thoroughly described and
analysed in [6], which we use as the basis for our presentation. In the second subsection we extend
Hopcroft’s algorithm for use in canonical labelling. We finally discuss how partition refinement relates
to the edge enumeration algorithms presented in the previous section.

4.1 A Deterministic Finite Automata View

We refer to standard text books such as [11] for details on automata, but recall briefly that a DFA is a
tuple \( (A, \Sigma, \delta, a_0, A') \) where \( A \) is a set of states, \( \Sigma \) is the input alphabet, \( \delta : A \times \Sigma \to A \) is a total transition
function, \( a_0 \in A \) is an initial state and \( A' \subseteq A \) is a set of final states. A coloured graph \( G \) is almost a DFA: \( V_G \)
can be taken both as the set of states and as the set of final states, the edge colours in \( G \) can be taken
as the input alphabet, and \( E_G \) determines a transition function albeit a partial one, meaning that the DFA
is incomplete. There is no dedicated initial state, but initial states play no role in partition refinement
[6]. A total transition function is traditionally obtained by adding an additional non-accepting sink state
with incoming transitions from all states for which such transitions are not defined in the original DFA.
However, it is convenient for our purposes to add a distinct such sink state for each state in the original
DFA. This allows us one further convenience, namely to ensure that each transition on a colour \( c \) has a
reverse transition on a distinct “reverse” colour, \( c^{-} \notin \Sigma \). These ideas are formalised as follows.

**Definition 11.** Let \( G \in \mathcal{G} \). Define the states \( A_G \overset{\Delta}{=} V_G \cup V^m_G \) where \( V^m_G \overset{\Delta}{=} \{ \text{Undef}, v \mid v \in V_G \} \). Define the
final states \( F_G = V_G \). Define the input alphabet \( \Sigma_G \overset{\Delta}{=} \text{Im}(\phi_G) \cup \{ c^{-} \mid c \in \text{Im}(\phi_G) \} \). Define the reversible
colour transition function \( \delta_G : A_G \times \Sigma_G \to A_G \) and the colour path function \( \hat{\delta}_G : A_G \times \Sigma^*_G \to A_G \) as follows:
Hence the tuple \((A_G, \Sigma_G, \delta_G, F_G)\) is a DFA with no initial state. It follows from rigidity of coloured graphs and from the choice of \textbf{Undef} states that \(\delta_G\) is injective. The colour path function \(\hat{\delta}_G\) gives the end state of a path specified by a colour word \(w\) from an initial state \(a\). Hence \(\hat{\delta}_G(a, w) \in V_G = F_G\) exactly when the word \(w\) is accepted from the state \(a\), or equivalently when the colour path \(w\) exists in the graph \(G\). With this in mind, the following notion of \textit{vertex bisimulation} corresponds exactly to the notion of DFA state equivalence given in [6] and proven to be the relation computed by Hopcroft’s algorithm (Corollary 15 in [6]).

**Definition 12.** Let \(G \in \mathcal{G}\). Define the \textbf{vertex bisimulation} relation \(\rho_G \subseteq A_G \times A_G\) as \(a \rho_G a’\) iff \(\forall w \in \Sigma_G^+ (\hat{\delta}_G(a, w) \in F_G \Leftrightarrow \hat{\delta}_G(a', w) \in F_G)\).

### 4.2 Adapting Hopcroft’s Algorithm

The strategy of using partition refinement for canonical labelling is to limit the number of vertices under consideration to those of a single equivalence class in the partition \(V_G / \rho_G\). If the selected class has size 1, the unique vertex in this class can be chosen as the source of canonical labelling via edge enumeration. If the selected class is larger, one of the two edge enumeration algorithms from Section 3 can be employed, but starting from only the vertices of this class.

The challenge then is how, exactly, to select an equivalence class from the unordered partition \(A_G / \rho_G\) resulting from Hopcroft’s algorithm. The selection must clearly be invariant under automorphism in order to be useful for canonical labelling. One approach could be to employ edge enumeration algorithms on the quotient graph \(G / \rho_G\), but in the worst case this yields quadratic time and hence defeats the purpose of partition refinement. Instead we give an extension of Hopcroft’s algorithm which explicitly selects an appropriate class. The following definition introduces the relevant notation, adapted from [6], required by the algorithm.

**Definition 13.** Let \(G \in \mathcal{G}\) and let \(\theta\) be an equivalence relation on \(V_G\). Let \(P, Q \in A_G / \theta\) and let \(x \in \Sigma_G\). Then define \(P_{Q, x} \overset{A}{=} P \cap \{\delta_G^{-1}(a, x) \mid a \in Q\}\) and \(P_{Q, x}^\theta \overset{A}{=} P \setminus P_{Q, x}\). Also define the refiners \(\text{ref}(P, \theta) \overset{A}{=} \{(Q, x) \in (A_G / \theta) \times \Sigma_G \mid P_{Q, x}^\theta \neq \emptyset \land |P_{Q, x}| < |P|\}\) and the objects \(\text{obj}(Q, x, \theta) \overset{A}{=} \{P \in A_G / \theta \mid (Q, x) \in \text{ref}(P, \theta)\}\).

Informally, the set \(\text{obj}(Q, x, \theta)\) specifies the \(\theta\)-classes which can be refined based on \(x\)-labelled transitions from states in the class \(Q\). The original version of Hopcroft’s algorithm from [6] is listed in Algorithm 5 in Appendix A for the sake of completeness. The algorithm maintains two sets: one is the partition at a given stage (line 1), and the second is the set \(L\) of pending refiners (line 2), i.e. pairs of classes and transitions which will be used for refining at a later stage. The algorithm then loops until there are no more refiners in \(L\). At each iteration a refiner \((Q, x)\) is selected arbitrarily from \(L\) and used to refine all the classes in \(\text{obj}(Q, x, \theta)\). The key insight of Hopcroft is to selectively add new refiners to
Algorithm 4: An extensions of Hopcroft’s partition refinement algorithm. The AddBetter routine
is defined as for Algorithm 5 in Appendix A.

\[
\text{input} \quad G \quad \text{output} \quad \rho_G \quad \text{and a selected } P \in (\mathcal{V}_G^u / \rho_G)
\]

1. \(M \leftarrow V_G\)
2. \(A_G / \theta \leftarrow \{V_G, V_G^u\}\)
3. \(L \leftarrow []\)

4. foreach \(x \in \Sigma_G\) in increasing order of \(<\) do
   - add \((V_G^u, x)\) to beginning of \(L\)

5. while \(L \neq \emptyset\) do
   - remove the first pair \((Q, x)\) from \(L\)
   - foreach \(P \in \text{obj}(Q, x, \theta)\) do
     - replace \(P\) with \(P_{Q,x}\) and \(P_{Q,x}^\theta\) in \(A_G / \theta\)
     - if \(P = M\) then
       - \(M \leftarrow P_{Q,x}\)
   - foreach \(P\) just refined do
     - foreach \(x' \in \Sigma_G\) in increasing order of \(<\) do
       - if \((P, x') \in L\) then
         - replace \((P, x')\) with \((P_{Q,x}, x')\) first and \((P_{Q,x}^\theta, x')\) second in \(L\)
       - else
         - AddBetter((\(P_{Q,x}, x')\), \((P_{Q,x}^\theta, x'), x', L\))
   - return \((\theta, M)\)

the set \(L\), namely only the better half of any classes which are split and not already included as a refiner (line 14). We choose the smallest half to be the better one, although other choices are possible.

The problem with the original algorithm for canonical labelling purposes is essentially that \(A_G / \theta\) and \(L\) are maintained as sets, hence introducing non-determinism at several points. One solution could be to maintain both \(A_G / \theta\) and \(L\) as lists instead, and adapt the algorithm to process their content in a consistent order. But this would complicate the analysis and implementation details meticulously described in [6]. Our adapted version, Algorithm 4, instead maintains only \(L\) as a list, which is not at odds with the implementation in [6]. There, \(L\) is a list of integer pairs with the first element identifying a partition and the second element identifying an edge colour. We then change the control flow governing \(L\) to become deterministic by exploiting the linear ordering of edge colours (lines 4 and 13), and otherwise consistently operate on elements of the list (lines 7, 15 and 17). The linear ordering on edge colours is assumed extended to states, e.g. by ordering the “reverse” colours after the standard colours. Finally, we explicitly maintain a “least” class (line 1): whenever the least class is refined, we consistently choose one sub-class to become the new least class (lines 10 and 11). Note how the rigidity property is key to this extension of Hopcroft’s algorithm.

The extensions do not affect the correctness of the algorithm because the particular non-deterministic choices made in Hopcroft’s algorithm do not affect its final output (Corollary 10 in [6]). The extensions do not affect the complexity analysis either, so the algorithm still runs in \(O(|A_G| \cdot \log |A_G|)\) time, which is the same as \(O(|V_G| \cdot \log |V_G|)\). In particular, the ordering of colours can be computed up front in
Figure 4.1: (a) A graph with one bisimulation class and two non-trivial automorphism classes (obtained from reflection on the two diagonals). (b) A graph with one bisimulation class and just one non-trivial automorphism class (obtained from vertical reflection); all vertices participate in single-coloured cycles of the same type.

\( O(|\Sigma_G| \cdot \log |\Sigma_G|) \) by standard sorting algorithms; the list operations on \( L \) can be implemented in \( O(1) \); and the comparison in line 10 can likewise be implemented in \( O(1) \) given that the classes in \( L \) can be represented by integers.

The key property needed for canonical labelling is that the “least” class \( M \) returned by the algorithm is invariant under automorphism. This is indeed the case; as for the edge enumeration algorithms, the intuition is that the control flow does not depend on vertex identity.

**Proposition 14.** Let \( G, G' \in \mathcal{G} \) and let \( \sigma \in I(G, G') \). Let \((\rho, M)\) and \((\rho', M')\) be the results of running Algorithm 4 on \( G \) and \( G' \), respectively. Then \( \sigma(M) = M' \).

It follows that the composite algorithm which first runs partition refinement via Algorithm 4, and then runs one of the edge enumeration algorithms 2 or 3 on the returned least class, is in fact a canonical labeller. In the cases where the selected \( \rho_G \)-class has size 1, the composite algorithm runs in worst case time \( O(|\Sigma_G| \cdot \log |\Sigma_G|) \). More generally, this bound also holds if there are “few” bisimulations, i.e. if \( V_G/\rho_G \) has size \( O(|\Sigma_G|) \). This is due to all bisimulation equivalence classes having the same size as the following proposition shows, and hence the particular choice of equivalence class does not affect time complexity.

**Proposition 15.** For any coloured graph \( G \) and any \( P, Q \in (V_G/\rho_G) \), \(|P| = |Q|\).

### 4.3 Bisimulation Versus Isomorphism

One could hope that the bisimulation and isomorphism relations for coloured graphs were identical, for then the choice of vertex from a selected \( \rho_G \)-class would not matter, giving a worst-case \( O(|V_G| \cdot \log |V_G|) \)
canonical labelling algorithm. But, unsurprisingly, this is not the case. Figure 4.1a shows a graph which has a single bisimulation equivalence class but two automorphism equivalence classes. Hence graphs of this kind, where all vertices have the same “local view”, capture the difficult instances of the graph isomorphism and canonical labelling problem for site graphs. The difficulty is essentially that isomorphisms are bijective functions and hence must account for vertex identity at some level. This is not the case for bisimulation equivalence.

One possible solution could be to annotate vertices with additional, “semi-local” information such as edge enumeration up to a constant length, as also suggested in [9], and then take this into account during partition refinement. However, this is unlikely to improve the asymptotic running time. Another approach could be to analyse the cycles of the input graph after partition refinement, and then take this analysis into account during a second partition refinement run on the quotient graph from the first run. The observation here is that all same-coloured paths in the quotient graph are cycles, and these cycles can be detected in linear time. However, vertices with the same local view and single-coloured cycles are not necessarily isomorphic, as demonstrated by Figure 4.1b. It seems that simple cycles of arbitrary colour combinations must be taken into account, e.g. to obtain cycle bases of the graph, but there is no clear means of doing so in linear time. Further attempts in this direction have been unsuccessful.

However, we observe that there are classes of coloured graphs for which bisimulation and isomorphism do coincide. This holds for example for coloured graphs with out and in-degree at most 1, and more generally for acyclic coloured graphs (trees). For these classes, the partition refinement approach does yield an $O(|V_G|\cdot \log|V_G|)$ worst case canonical labelling algorithm. Furthermore, linear time may be possible for these classes using e.g. the partition refinement in [4], assuming an extension similar to that of Algorithm 4 can be realised.

Finally, we note that there is an open question of how the parallel edge enumeration algorithm behaves on graphs with “few” bisimulations. We conjecture that the algorithm may in these cases run in $O(|V_G|\cdot \log|V_G|)$ time. If so, partition refinement would be unnecessary. However, for the time being, the partition refinement approach does serve the purpose of providing the upper bound of $O(|V_G|\cdot \log|V_G|)$ for graphs with few bisimulations.

5 Conclusions

We have considered the problem of canonical labelling of site graphs, which we have shown reduces to canonical labelling of standard digraphs with edge colourings. We have presented an algorithm based on edge enumeration which runs in $O(|V_G|^2)$ worst-case time, and in $O(|V_G|\cdot \log|V_G|)$ average-case time for graphs with many automorphisms. A variant of this algorithm, based on parallel enumeration of edges, is likely to perform well in praxis, but in general does not improve on the worst case complexity bounds. However, the question of how the parallel algorithm performs in cases with few automorphisms remains open. If it is found to run in $O(|V_G|\cdot \log|V_G|)$ time, this yields an overall $O(|V_G|\cdot \log|V_G|)$ average-case algorithm and hence resolves the open question of whether such an algorithm exists. We have also introduced an algorithm based on partition refinement which can be used as a preprocessing step, yielding $O(|V_G|\cdot \log|V_G|)$ worst case time for graphs with few bisimulation equivalences.

A different line of attack is taken in [9] which introduces a notion of a graph’s “gravity centre”, namely a subgraph on which it is sufficient to detect automorphisms. Hence this approach is efficient when the gravity centre is small, and could also be a useful pre-processing step for canonical labelling. However, many of the difficult graphs that we have considered, including the one in Figure 4.1b, do not appear to have small gravity centres.
Other related work includes that on the general graph isomorphism problem which has been extensively studied in the literature. Hence highly optimised algorithms, such as the one by McKay [8], exist, although none run in sub-exponential time on “difficult” graphs. Many other special cases have been studied, including notably graphs with bounded degree for which worst-case polynomial time algorithms do exist [1]. Site graphs can indeed be encoded into standard graphs with bounded degree. However, polynomial time algorithms on such encodings do not appear to be sub-quadratic or even quadratic. Perhaps surprisingly, isomorphism of site graphs, or equivalently digraphs with edge colourings, has to the best of our knowledge not been treated in the general literature. Although the difficult cases are perhaps rare and of limited practical relevance, the question is theoretically interesting. It certainly appears to be conducive to infection with the graph isomorphism disease [10].

Acknowledgements

We thank the anonymous reviewers for useful comments. The work of the second author was supported by an EPSRC Postdoctoral Fellowship (EP/H027955/1).

References


A Site Graphs and Hopcroft’s Original Algorithm

In the literature site graphs are typically defined as expressions in a language, which is natural when considering simulation and analysis of rule-based models. We give a more direct definition suitable for our purposes. Site graphs in the literature often include internal states of sites, representing e.g. post-translational modification. We omit internal states but they are straightforward to encode.

Definition 16. Let \((\Sigma_p, \preceq_p)\) and \((\Sigma_s, \preceq_s)\) be given, disjoint linearly ordered sets of protein and site names, respectively. Then \(\mathcal{IG}\) is the set of all site graphs \(S = (V, E, \phi_p)\) satisfying:

1. \(V = \{1, \ldots, k\}\) is a set of vertices.
2. \(E \subseteq (V \times \Sigma)_2\) is a set of site-labelled, undirected edges satisfying that \(\forall e, e' \in E, e \neq e' \Rightarrow e \cap e' = \emptyset\).
3. \(\phi_p : V \rightarrow \Sigma_p\) is a vertex (protein) naming.

Note the key condition on edges that a given site can occur at most once within a vertex.

Definition 17. Let \(S, S' \in \mathcal{IG}\). A site graph isomorphism is a bijective function \(\sigma : V_S \rightarrow V_{S'}\) satisfying:

1. \(\forall v_1, v_2 \in V_S. [((v_1, s_1), (v_2, s_2)) \in E_S] \Leftrightarrow [((\sigma(v_1), s_1), (\sigma(v_2), s_2)) \in E_{S'}]\).
2. \(\forall v \in V_S. \phi_{pS}(v) = \phi_{pS'}(\sigma(v)).\)

The first condition states that edges and edge site names are preserved by the isomorphism, and the second condition states that protein names are preserved. We next show how to encode site graphs into coloured graphs. Let \((\Sigma_p, \preceq_p)\) and \((\Sigma_s, \preceq_s)\) be given. The aim is to construct a linearly ordered edge colour set \((\Sigma, \prec)\) and a total function \(\rho : \mathcal{IG}(\Sigma_p, \Sigma_s) \rightarrow \mathcal{CG}(\Sigma)\) which preserves isomorphism. We define the colour set as \(\Sigma \overset{A}{=} \mathcal{P}(\Sigma_p \times \Sigma_s) \cup \Sigma_p\), and the linear order on \(\Sigma\) as \(c \prec c'\) iff one of the following conditions hold:

- \(c \in \Sigma_p \land c' \notin \Sigma_p\) or
- \(c, c' \in \Sigma_p \land c \preceq_p c'\)
- \(c, c' \in \mathcal{P}(\Sigma_p \times \Sigma_s) \land \text{Sort}_{\preceq_s}(c) \preceq_{\preceq_s} \text{Sort}_{\preceq_s} c'\)

In the latter case we assume the \(\preceq_s\) relation extended lexicographically to pairs and lists as usual. Let \(S \overset{A}{=} (V, E, \phi_p)\) be a given site graph. We then define \(\rho(S) \overset{A}{=} (V', E', \phi')\) where:

1. \(V' \overset{A}{=} V\).
2. \(E' \overset{A}{=} E_1' \cup E_2'\) where:
   - \(E_1' \overset{A}{=} \{(v, v') \mid \exists s, s', s \preceq_s s' \land (s, s') = \min_{\preceq_s} \{(s, s') \mid \{(v, s), (v', s')\} \in E\}\}
   - \(E_2' \overset{A}{=} \{(v, v) \mid v \in V\}\)
3. \(\phi'(v, v') \overset{A}{=} C_1' \cup C_2'\) where
   - \(C_1' \overset{A}{=} \{(s, s') \mid \{(v, s), (v', s')\} \in E\}\)
   - \(C_2' \overset{A}{=} \begin{cases} \{\phi_p(v)\} & \text{if } v = v' \\ \emptyset & \text{otherwise} \end{cases}\)
The encoding does not affect vertices. Note that site graphs can have multiple unordered edges between nodes while coloured graphs have at most one, ordered edge. The direction of this one edge is determined in Step 2a from the site ordering of the least pair of sites, where the least pair of sites is determined from the extension of the site ordering to pairs. All vertices have self-loops (step 2b) which are used to encode vertex colour as edge colour in step 3b. Step 3a assigns a colour to an edge as the union colours of each edge between the edge in the site graph; the ordering of colours follows that assigned to the edge.

**Proposition 18.** The coding function $\rho$ satisfies the following for all $S, S' \in \text{Dom}(\rho)$:

1. **Injective:** $\rho(S) = \rho(S') \Rightarrow S = S'$.
2. **Respects isomorphism:** $S \simeq S' \iff \rho(S) \simeq \rho(S')$.
3. **Linear size:** $|\rho(S)| = O(|S|)$.
4. **Linear time computable:** $\rho$ is computable in $O(|S|)$.

It follows immediately that the coding function together with a canonical labeller for coloured graphs can be used to define a canonical labeller for site graphs as follows.

**Corollary 19.** Given a canonical labeller $L : \prod G \in \mathcal{G} . [G] \simeq$ on coloured graphs running in $O(f(|G|)) \geq O(|G|)$ time, the function $L^*(S) \triangleq \rho^{-1}(L(\rho(S)))$ is an $O(f(|G|))$ time canonical labeller on site graphs.
Algorithm 5: A version of Hopcroft’s original algorithm adapted from Algorithm 4 in [6].

\begin{verbatim}
input : a graph $G$
output: the relation $\rho_G$
1 $A_G/\theta \leftarrow \{V_G, V_G^u\}$
2 $L \leftarrow \emptyset$
3 foreach $x \in \Sigma_G$ do
4   add $(V_G^u, x)$ to $L$
5 while $L \neq \emptyset$ do
6   remove a pair $(Q, x)$ from $L$
7   foreach $P \in \text{obj}(Q, x, \theta)$ do
8     replace $P$ with $P_{Q,x}$ and $P^{Q,x}$ in $A_G/\theta$
9   foreach $P$ just refined do
10      foreach $x' \in \Sigma_G$ do
11         if $(P, x') \in L$ then
12            replace $(P, x')$ with $(P_{Q,x}, x')$ and $(P^{Q,x}, x')$ in $L$
13         else
14            AddBetter($(P_{Q,x}, x'), (P^{Q,x}, x'), x', L)$
15 return $\theta$
\end{verbatim}
Rule-Based Modeling of Polymerization Reactions

Thilo Krüger  
Verena Wolf

Department of Computer Science  
Saarland University  
Saarbrücken, Germany

thilo@dennis.cs.uni-saarland.de  
wolf@cs.uni-saarland.de

The modeling of polymerization reactions is a field of great interest. Since polymer chemistry deals with synthetic macromolecules, which are the raw materials of plastics, polymerizations are important for the area of materials science. By using models of polymerizations it is possible to optimize industrial processes in two ways, optimization of the process itself (utilization/yield) and therefore the reduction of production cost and optimization of the properties of the technical products, such as the stiffness of plastics, the porosity of foams or the adhesive power of adhesives. The most common approach to model polymerization reactions is based on ordinary differential equations (see [6] and the references therein).

Certain aspects of polymerization reactions, however remain hidden in ODE models. Most of these aspects concern the microstructure of the products of the reactions. For example, it is difficult to model the exact monomer sequence in copolymer products. Another example for such a microstructural issue is the distribution of branching points and other network parameters in polymerizations with branching reactions. Polymerizations can be branching, if some monomers have more than two possible reaction sites. In the extreme case, the complete product of such a reaction can be a single molecule, representing a complex molecular network. Recent work simulates branching reactions using the Gillespie method [5]. The major problem these approaches are dealing with is the large number of possible successor states for most of the states. The presented solutions for this problem are based on grouping of the reactions [2, 8]. These solutions are specialized for the presented models and constitute no general approach to deal with the problem of large numbers of transitions. Nevertheless, more detailed network characteristics can be obtained in some of the stochastic models. A concrete example for data which could be obtained by such models but which were difficult to get via ODE models is the bivariate MW-LCB distribution [8].

In general there are many technical products, which contain branched polymers. A well-known example are epoxy resins [9], which we use here to illustrate the principle mechanisms of a possible branching polymer reaction. The first reaction is the formation of linear prepolymer (cf. Figure 1 in [9] for the structure of the prepolymer DGEBA). The reactive groups (sites) of the products of this reaction are epoxy-groups, which are highly reactive cyclic ethers with three ring atoms. The prepolymer undergo a curing process afterwards (cf. Figure 1 in [9] for examples of hardeners). Therefore, they might for example react with a diamine. Since each amine group can react twice, a diamine reacts like a molecule with four sites. This reaction leads to a polymer network, therefore obtaining the detailed network parameters by modeling via ODEs is difficult as mentioned above.

In order to combat the problems related to the huge number of possible transitions of a state, we consider stochastic simulation methods for complex biochemical reaction networks. Rule-based modeling approaches solve the complexity problem. A detailed analysis of this approach can be found in [1, 3, 10]. In the case of the example above the epoxy groups are reacting with the hydrogens of the amine groups, not depending on the chemical environment of the epoxy group. Therefore applying rule-based modeling will gain a huge reduction of the complexity.
In this work, our goal is to apply rule-based modeling to polymerization reactions. Since the existing tools for running simulations on rule-based models are specialized for applications from systems biology, several adaptions are necessary to use them for the simulation of polymerization reactions. The largest difference is that the modeled processes are macroscopic. Polymerizations happen in large reactors instead of small cells, where processes in systems biology occur. Therefore the number of participating molecules is much higher than in models of microscopic processes. To get representative results for microstructural parameters of the products, the simulations should run with large numbers of monomers and therefore the product should be represented in a compressed way. One possibility for this is to represent longer chains of monomers without active reaction sites via a single symbol $M_x$, with $x$ as the number of the monomers $M$ in the chain. Other differences are due to model details of polymerization reactions. Some rate constants for example were found to be dependent on the chain length of the reacting polymers [7]. In certain polymerization reactions the formation of molecular cycles is not possible [4]. Forbidding cycles and combining chains of the same monomer type can be modeled with existing tools, while chain-dependent transition rates cannot. One important goal of modeling polymerization reactions is the computation of distributions of quantities like numbers of molecules with a certain chain length or the number of branches dependent on the weight like in [8]. The existing tools do not directly offer such a functionality. These examples show that specialized software tools for the rule-based simulation of polymerization reactions are necessary.

References


Robustness Analysis of Stochastic Systems

Luboš Brim, Milan Češka, Sven Dražan*, David Šafránek
Systems biology laboratory at Faculty of Informatics, Masaryk University, Brno, Czech Republic
sven@mail.muni.cz

Models of biological phenomena occurring in low populations (molecules or cells) may be more precisely modeled stochastically rather than by ordinary differential equations (ODEs), this enables to capture effects of random noise or fluctuations leading to multiple stable states. Even under conditions of stochastic fluctuations and changing parameters some properties of models may be robustly satisfied, quantification and precise computation of such robustness is the main goal of our approach.

Properties of models can be expressed in temporal logics such as Continuous Stochastic Logic (CSL) [1, 2] and automatically verified using statistical model checking with extensive use of simulations or numerical methods based on uniformization and transient probability analysis [5]. Even though the former is sometimes considered more computationally feasible since the later makes use of large state space structures, uniformization can outperform statistical model-checking in cases where phenomena with very low probabilities are involved [6] or results are required with high precision.

Robustness, as generally defined by Kitano [4], is the ability of a system $S$ to maintain its functionality $a$ over a set of perturbations $P$ of its internal or external parameters. $R^S_{a,P} = \int_P \psi(p) D^S_a(p) dp$. Formally it is the integral of the evaluation function $D^S_a$ over all $p \in P$ weighted by the probability of occurrence of each such perturbation $\psi(p)$. $D^S_a(p)$ can be computed as the quantitative validity of the temporal property $a$ in $p$ by above mentioned formal methods.

In the case of a stochastic system we can define functionality by means of a CSL formulae and understand perturbations as changing values of its numerical constants such as initial population numbers, kinetic rates or stoichiometric coefficients leading to different system dynamics. These numerical constants may not be known and are only guessed within several orders of magnitude, may be uncertain due to limited precision of measurements or do not have a single value because of inherent heterogeneity.

While for deterministic ODE models robustness computation can be carried out using trajectory simulation and property monitoring in different perturbation points, a stochastic model has instead of a single trajectory a time evolving state probability distribution which substantially increases computational complexity.

To be able to compute the robustness of a stochastic system $S$ with respect to a CSL property $a$ and a perturbation space $P$ we must be able to estimate the quantitative validity/probability of $a$ over $P$ or subspaces of $P$. For discrete parameters this is straight forward although possibly demanding, however for continuous parameters such as kinetic rates only approximative sampling techniques were feasible up to now.

By using a modified version of uniformization [3] we have proposed a method to compute the upper and lower bounds of property validity over continuous parameter spaces. The main idea is to compute for each model state its probability as an upper and lower bound instead of a single number. This is possible by computing the minimum resp. maximum stochastic rate of each reaction with parameters being perturbed over the examined value interval. In case of more complex nested CSL properties minimum and maximum sets of states satisfying nested formulae have to be computed. Depending on the form of reaction kinetics min/max rates can be more or less precisely computed (mass action kinetics precisely,
Robustness Analysis of Stochastic Systems

Figure 1: Schloegl's model: state probabilities show bistability; size of each stable mode is analyzed with respect to the perturbed kinetic parameter $k_1$; computation with precision 1% (left) and 0.1% (right); upper bound on probability of each state in red, lower in green.

sigmoid hill kinetics approximately), together with local greedy computation the resulting under/over approximation is more or less exact and refinement may be needed to achieve requested precision.

This modified min/max uniformization together with a refinement procedure on the parameter space leads to an over/under approximation of the robustness with arbitrary precision accompanied by correspondingly increased computation complexity. For even small models, computation with a sensible precision can be very demanding in the order of hours, fortunately the algorithm can be straight-forwardly distributed.

A prototype implementation based upon the tool PRISM is used for dynamics analysis of the simple one-dimensional Schloegl's model [7] with pure mass action kinetics (Figure 1, 1000 states). A more elaborate four-dimensional model of a virtual signal transduction mechanism in cells is analyzed to show how two different variants of the model cope with increasing levels of intrinsic noise induced by gene transcription (116 000 states).

References

Mathematical modelling of the platelet-derived growth factor (PDGF) signalling pathway

Andrzej Mizera, Jun Pang, Thomas Sauter and Panuwat Trairatphisan
University of Luxembourg, Luxembourg

andrzej.mizera@uni.lu

Constructing a mathematical/computational model for a biological system consists of two main steps: first, of specifying the model structure and, second, of determining the numerical values for the parameters of the model. Usually, the structure of the model is represented in the form of a biochemical reaction network and the parameters are the reaction rate constants. The values of the reaction rates can either be measured directly in experiments or determined by fitting the model to experimental data by performing parameter estimation. Since the former approach is often impeded by technical limitations or high costs of experimental practice, parameter fitting is a fundamental problem in systems biology research.

In this study we consider issues related to parameter estimation for a model of the platelet-derived growth factor (PDGF) signalling pathway with use of steady-state data and so called knock-down mutant models, i.e., variants of the original model obtained by suppressing one or more interactions in the model. Since the knock-down mutants of the real biological system can be obtained and investigated in experimental practice as well as the physical/chemical properties are usually common for all variants, the measurements of the mutants can enrich the set of data available for parameter estimation. We consider parameter estimation both in the deterministic (system of ordinary differential equations) and stochastic (continuous-time Markov chain) modelling frameworks. We discuss certain difficulties related to parameter estimation we encountered while modelling the PDGF signalling pathway and present our solutions.
Rule-based modelling as a platform for the analysis of synthetic self-assembled nano-systems
Extended Abstract

Abdulmelik Mohammed Eugen Czeizler
Aalto University, Espoo, Finland
abdulmelik.mohammed@aalto.fi eugen.czeizler@aalto.fi

Recent advances in DNA-based nano-technology have opened the way towards the systematic engineering of inexpensive, nucleic-acid based nano-scale devices for a multitude of purposes[4]. The field itself evolved dramatically in the past 10-15 years from a state where careful manual design and intimate knowledge of DNAs atomic structure were needed for the design of simple structures [6], to the current algorithmic approaches employing the use of universally engineered elementary building blocks that are further functionalized and driven to self-assemble into the desired shapes, see [2, 8]. The Tile Assembly Model (TAM), which is the theoretical design platform for the above modular assembly scheme, employs the use of so-called DNA-tiles, which can be seen as unit square blocks with active glues on their four (North, East, South and West) edges. These active glues, implemented using single-stranded DNA sticky-ends, are driving the self-assembly process and determine the controlled aggregation of the tiles into the desired structures.

Since its introduction [5], the TAM formalism has been used successfully both in designing complex assembly nano-structures and in providing predictions regarding the possible experimental outcomes of certain designs. To this extent, the Tile Assembly Model has been expanded to a kinetic counterpart, kTAM [9]. The kTAM incorporates two types of reactions: association of tiles to an assembly (forward reaction) and dissociation (reverse reaction), see e.g. Figure 1 a). In a forward reaction, any tile can attach to the assembly at any position, even if only a weak bond is formed; the rate of this reaction, $r_f$, is proportional to the concentration of free tiles in the solution. In the second reaction, any tile can detach from the assembly with rate $r_{r,b}$, $b \in \{0...5\}$, which depends exponentially on the total strength of the
bonds between the tile and the assembly. Thus, tiles which are connected to the assembly by fewer or weaker bonds are more prone to dissociation than those which are strongly connected. Previous computational modeling of kTAM has been performed almost exclusively using a special tailored version of the Gillespie’s stochastic simulation algorithm. The assembly starts at \( t = 0 \) from a seed structure. Then, in discrete time steps, tiles are added or detached from the assembly, according to the corresponding association and dissociation rates.

In our work, we propose a rule-based modelling approach for predicting the time evolution of kTAM systems and their variants. Rule-based modelling is a discrete modeling technique [1] in which molecules are represented as agents with a finite number of free sites. The sites, which may have several internal states, allow for agent-agent binding, thus generating molecular complexes. Rules are defined based on local patterns rather than by the full specification of the reactants, and thus provide a compact representation on how agents interact. In this way, rather than handling explicitly a large number of model variables, we often only have a small number of local interaction rules. This makes the rule-based paradigm well suited in taming the problem of the combinatorial explosion of the state space, as is the case of modelling self-assembly or polymerization systems, see e.g., [7].

In the modelling of DNA tile systems, the primary agent, Tile, has a number of sites, including Nedge, Eedge, Sedge, and Wedge, each with internal states corresponding to possible glues of these tiles. Then, both tile-association and tile-dissociation reactions can be implemented by appropriate local rules. For example, the addition of a tile to a free North site, as depicted in Figure 1 b), can be implemented by the local rule,

\[
\text{Tile(Nedge, in} \sim 1) + \text{Tile(Sedge, in} \sim 0) \rightarrow \text{Tile(Nedge}!1,\text{in} \sim 1).\text{Tile(Sedge}!1,\text{in} \sim 1),
\]

where \( \text{in} \) is a specific site which is in state 1 if the tile is inside an assembly and 0 otherwise. The decoding of the above rule is as follows: a Tile with an unbounded Nedge site placed inside the assembly (i.e., \( \text{in} \sim 1 \)) interacts with a free Tile (i.e., \( \text{in} \sim 0 \)) with an unbounded Sedge site, and the two become bonded on the sites Nedge and Sedge; the reaction has a kinetic rate constant \( k_{on} \). Tile dissociation reactions, which are dependent on the total bond strength of the tiles, can be implemented in a similar manner.

There are a number of advantages in using a rule-based modelling approach for kTAM. Since this is a coarse-grain modelling framework, it allows the examination of a very diverse family of observables. Thus, the system can be analyzed extensively and both final and intermediate states can be inquired with detailed precision. Also, due to the current availability of appropriate software frameworks, numerical simulations of rule-based models are simple to run. Such simulations can be written in pseudocode, using e.g. BNGL [1] or \( \kappa \) [3]. Thus, the emphasis is placed on describing the system’s reaction rules, and not in dealing with the numerical simulation algorithm. Hence, custom simulation are easy to create, update, and modify.

A major drawback of previous modelling approaches was that they could express only those systems where there exists a single growing assembly, and all the reactions (addition and dissociation) were between this unique assembly and the free floating tiles. This situation however, does not cover the case where two partial assemblies, each consisting of more than one tile, are interacting. However, using the rule-based modelling framework one can implement such reactions too. This opens the possibility of modelling various variants of TAM (which are closer to experimental implementations), accounting for both assembly–tile interactions and assembly–assembly interactions, e.g., staged or hierarchical tile assembly models.

In our study, we first created a rule-based model of kTAM. Using the same kinetic parameters as in previous numerical implementations of kTAM, we compared the prediction of those models (regarding time-growths and error-fraction of the assembly) with that of our model, see e.g. Figure 2. Then, we
introduced some small variations into the model and studied how does this affect the overall behaviour of the system. Also, we implemented a rule-based model for staged TAM. Using numerical simulations, we analyzed the possible shortcomings of the combinatorial design in the case of possible experimental implementations.

References

An Algebraic Approach to Gene Assembly in Ciliates

Robert Brijder
Hasselt University and Transnational University of Limburg, Belgium
robert.brijder@uhasselt.be

Gene assembly is an intricate process occurring in unicellular organisms called ciliates. During this process a nucleus, called the micronucleus, is transformed into a functionally and structurally different nucleus called the macronucleus. It is accomplished using involved DNA splicing and recombination operations [10, 11]. Gene assembly has been formally studied on the level of individual genes, see, e.g., the monograph [8] and a recent survey [4].

The theory of Euler circuits in 4-regular graphs was initiated in a seminal paper by Kotzig [9]. Bouchet further developed the theory by relating it to delta-matroids [2] and isotropic systems [1, 3]. In [2], Bouchet uses a matrix transformation that turns out to be “almost” principal pivot transform [12]. Principal pivot transform, delta-matroids, and isotropic systems enjoy many interesting properties which have direct consequences for the theory of Euler circuits in 4-regular graphs.

Although, at first glance, the formal theory of gene assembly seems to be related to the theory of Euler circuits in 4-regular graphs, there have been little attempts to fit the former theory into the latter. In this talk we do exactly this. We show that the formal model of gene assembly can be defined quite efficiently in terms of 4-regular graphs. We discuss consequences of known results in the theory of 4-regular graphs (including, e.g., results related to principal pivot transform and delta-matroids) for the theory of gene assembly. This talk is based on the articles [6, 5, 7].

References


Large-Scale Text Mining of Biomedical Literature

Filip Ginter
Department of Information Technology, University of Turku
Turku, Finland
ginter@cs.utu.fi

Natural Language Processing in the Biomedical Domain (BioNLP) deals with text mining of scientific biomedical literature, most typically represented by the PubMed\(^1\) abstracts, and less frequently also the Open Access subset of PubMed Central\(^2\) full article texts. The methods of BioNLP are many, and address a wide variety of tasks relevant to information search and indexing, as well as hypothesis generation.

Recently, the particular problem of event extraction has become the focus of a community-wide effort, due to the series of BioNLP Shared Tasks on Event Extraction \[2\]. Events, as defined within the Shared Tasks, are an expressive, formal representation of statements in the underlying text. An event is a structure with a type from a pre-defined type vocabulary (such as Regulation and Protein catabolism), and arguments with a role, typically Cause or Theme. An important property of events is that their arguments are not only physical entities such as genes and proteins, but also other events, thus forming recursively nested structures. An example event is shown in Figure 1 (upper half).

![Event example](image)

Figure 1: An event extracted from a single sentence. Note that the Theme argument of the Positive regulation event is a Phosphorylation event, thus forming a recursively nested event structure. (Figure credits: \[4\])

The BioNLP Shared Tasks led to a rapid progress in the development of advanced event extraction systems. Even though much effort was invested in their development, only few of these systems were

---

\(^1\)http://www.pubmed.com
\(^2\)http://www.ncbi.nlm.nih.gov/pmc/
made publicly available and deployed in a real-world setting. We have applied the Turku Event Extraction System [1] to the entire PubMed (22M abstracts) and PubMed Central Open Access subset (460K full text articles), creating a large-scale dataset consisting of 40M biomolecular events among 76M gene and protein text mentions [4].

Event extraction systems treat gene and protein mentions as simple strings. Such an approach does not allow an easy integration with other resources due to the inherent ambiguity of these mentions, whereby one symbol can refer to any number of genes/proteins across many species and conversely a single gene/protein can be referred to by a number of synonymous symbols. In order to be able to utilize the extracted events, we carried out a gene normalization step [5], assigning the gene/protein mentions in text to their Entrez Gene identifier. The events can consequently be aggregated across different documents and, more importantly, related to the wealth of other databases with information about the genes and proteins involved. Finally, through homology-based gene family definitions, such as HomoloGene [3], events can be aggregated over entire gene families allowing for homology-based hypothesis generation. The gene normalization and gene family assignment steps are illustrated in Figure 1 (lower half).

While the recursive nesting of events allows a more faithful formal representation of the underlying textual statements, the events prove difficult to manipulate and integrate with other resources. Therefore, we also introduce a network view which re-interprets the complex event structures into a network form with genes and proteins as nodes and events as edges. This network can be easily integrated with other resources such as co-expression assays, regulatory network predictions, and known interaction databases.

The resulting resource, EVEX [4], can be accessed using a web interface\(^3\) as well as downloaded under an open license.

Acknowledgments: This work was carried out in a close collaboration with Dr. Sofie Van Landeghem, VIB/Ghent University, Belgium.

References


\(^3\)http://www.evexdb.org