Sepinoud Azimi

Computational Models for and from Biology

Simple Gene Assembly and Reaction Systems

Turku Centre *for* Computer Science

TUCS Dissertations
No 205, November 2015

Computational Models for and from Biology

Simple Gene Assembly and Reaction Systems

Sepinoud Azimi

To be presented, with the permission of the Faculty of Science and Engineering of the Åbo Akademi University, for public criticism in Auditorium Gamma on November 06, 2015, at 12 noon.

Åbo Akademi University
Department of Computer Science
Joukahaisenkatu 3-5 A
20520 Turku
Finland

2015

Supervisor

Professor Ion Petre
Åbo Akademi University
Department of Computer Science
Joukahaisenkatu 3-5 A
20520 Turku
Finland

Reviewers

Professor Nataša Jonoska
Department of Mathematics and Statistics
University of South Florida
Fowler Ave, Tampa, FL 33620
United States

Professor Hendrik Jan Hoogeboom
Department of Computer Science
Leiden University
Snellius, Niels Bohrweg 1, 2333 CA Leiden
The Netherlands

Opponent

Professor Nataša Jonoska
Department of Mathematics and Statistics
University of South Florida
Fowler Ave, Tampa, FL 33620
United States

ISBN 978-952-12-3289-3
ISSN 1239-1883

Abstract

The advancement of science and technology makes it clear that no single perspective is any longer sufficient to describe the true nature of any phenomenon. That is why the interdisciplinary research is gaining more attention overtime. An excellent example of this type of research is natural computing which stands on the borderline between biology and computer science. The contribution of research done in natural computing is twofold: on one hand, it sheds light into how nature works and how it processes information and, on the other hand, it provides some guidelines on how to design bio-inspired technologies.

The first direction in this thesis focuses on a nature-inspired process called gene assembly in ciliates. The second one studies reaction systems, as a modeling framework with its rationale built upon the biochemical interactions happening within a cell.

The process of gene assembly in ciliates has attracted a lot of attention as a research topic in the past 15 years. Two main modelling frameworks have been initially proposed in the end of 1990s to capture ciliates' gene assembly process, namely the intermolecular model and the intramolecular model. They were followed by other model proposals such as template-based assembly and DNA rearrangement pathways recombination models. In this thesis we are interested in a variation of the intramolecular model called simple gene assembly model, which focuses on the simplest possible folds in the assembly process. We propose a new framework called directed overlap-inclusion (DOI) graphs to overcome the limitations that previously introduced models faced in capturing all the combinatorial details of the simple gene assembly process. We investigate a number of combinatorial properties of these graphs, including a necessary property in terms of forbidden induced subgraphs. We also introduce DOI graph-based rewriting rules that capture all the operations of the simple gene assembly model and prove that they are equivalent to the string-based formalization of the model.

Reaction systems (RS) is another nature-inspired modeling framework that is studied in this thesis. Reaction systems' rationale is based upon two main regulation mechanisms, facilitation and inhibition, which control

the interactions between biochemical reactions. Reaction systems is a complementary modeling framework to traditional quantitative frameworks, focusing on explicit cause-effect relationships between reactions. The explicit formulation of facilitation and inhibition mechanisms behind reactions, as well as the focus on interactions between reactions (rather than dynamics of concentrations) makes their applicability potentially wide and useful beyond biological case studies. In this thesis, we construct a reaction system model corresponding to the heat shock response mechanism based on a novel concept of dominance graph that captures the competition on resources in the ODE model. We also introduce for RS various concepts inspired by biology, e.g., mass conservation, steady state, periodicity, etc., to do model checking of the reaction systems based models. We prove that the complexity of the decision problems related to these properties varies from P to NP- and coNP-complete to PSPACE-complete. We further focus on the mass conservation relation in an RS and introduce the conservation dependency graph to capture the relation between the species and also propose an algorithm to list the conserved sets of a given reaction system.

Sammanfattning

Utvecklingen av vetenskap och teknologi visar att inget enskilt perspektiv längre är tillräckligt fr att beskriva den sanna karaktären av ett fenomen. På grund av detta har gränsöverskriande forskning fått allt mer uppmärksamhet. Ett braexempel av denna sortens forskning är bioberäkning som befinner sig i gränslinjen mellan biologi och datavetenskap. Bioberäkning:s bidrag till forskningen är tvåfaldigt: å ena sidan uppmärksammar den hur naturen fungerar och hur den processerar information; å andra sidan förser den med riktlinjer om hur man skall designa bio-inspirerade teknologier.

Den första delen av denna avhandling fokuserar på en naturinspirerad process som kallas genuppsättning i ciliater. Den andra delen av avhandlingen studerar reaktionssystem som ett modelleringsramverk vars logiska grund baserar sig på biokemiska interaktioner i en cell.

Genuppsättning processen i ciliater har nått mycket uppmärksamhet inom forskning under de senaste 15 åren. Två huvudsakliga modelleringsramverk har ursprungligen blivit föreslagna i slutet av 1990-talet för att fånga ciliaters genuppsättning process. Dessa är den intermolekulära modellen och den intramolekulära modellen. Dessa följdes av förslag av andra modeller såsom templatbaserad assembly och modeller av DNA-omstruktureingsvägar. I denna avhandling intresserar vi oss för en variation av den intramolekulära modellen som kallas enkel genuppsättning. Den fokuserar på de enklaste möjliga invikningar i uppsättningsprocessen. We föreslår ett nytt ramverk som kallas riktade överlappande inklusionsgrafer (DOI) för att överkomma de begränsningar som de tidigare modellerna stötte på då de försökte fånga alla de kombinatoriska detaljerna av den enkla genuppsättning processen. Vi har studerat ett antal kombinatoriska egenskaper hos dessa grafer, inklusive en nödvändig egenskap av i form av förbjudna härledda subgrafer. Vi introducerar också DOI grafbaserade regler för omskrivningsregle som fångar alla operationer na av den enkla genuppsättning modellen och bevisar att de är ekvivalenta med modellens strängbaserade formalisering.

Reaktionssystem (RS) är ett annat naturinspirerat modelleringsramverk som studeras i denna avhandling. Reaktionssystemens logiska grund baseradr sig på två huvudsakliga regleringsmekanismer: facilitation och inhibi-

tion. Dessa kontrollerar interaktionerna mellan de biokemiska reaktionerna. Reaktionssystem är ett komplementärt modelleringsramverk för de traditionella kvantitativa ramverken. Reaktionssystem fokuserar på explicita orsak-verkan-samband mellan reaktioner. Den explicita formuleringen av facilitations- och inhibitionsmekanismer bakom reaktioner liksom fokuset på interaktioner mellan reaktioner (snarare än koncentrationernas dynamik) gör att modellen potentiellt är vitt användbar även vid andra än biologiska fallstudier. I denna avhandling konstruerar vi en modell för reaktionssystem som motsvarar värmechocksresponsmekanismen baserat på ett nytt konsept av dominansgrafer som fångar konkurrensen om resurser i ODE-modellen. Vi introducerar också för RS diverse begrepp som inspireras av biologin, såsom masskonservation, stabilt tillstånd, periodicitet osv. för att kunna kontrollera de reaktionssystemsbaserade modeller. Vi bevisar att komplexiteten av beslutsproblem relaterade till dessa egenskaper varierar från P, NP och coNP till PSPACE-fullständiga problem. Vi fokuserar vidare på masskonservationsrelationen i RS och introducerar en konservationsberoendegraf för att fånga relationen mellan arterna och föreslår därtill en algoritm för att räkna upp de konserverade mängderna av ett givet reaktionssystem.

Acknowledgements

Looking back at my Phd journey, I see many glittering stones I left behind, each standing for a great moment, a wonderful experience, a memorable lesson. I surely could not have enjoyed this journey, as much as I did, if I would have travelled it alone. Here, I want to thank all those whom I had the chance to enjoy their help and company along the way.

First and foremost, I want to thank my supervisor Professor Ion Petre. I am deeply grateful for helping me build my character as a researcher, for teaching how to approach the problems and how to find the solutions not necessarily in the first obvious places. It was great to know that no matter what problem I was facing, I could always find inspirations, support and guidance in our meetings. Thanks for providing me the opportunity to explore a wide range of topics and to focus on what I was passionate about. I totally appreciate this level of freedom I was granted as a Phd student.

I am very thankful to Professor Nataša Jonoska that she kindly agreed to be a reviewer of my dissertation and that she accepted to act as the opponent at my doctoral defence. I am very grateful to Professor Hendrik Jan Hoogeboom for accepting to be a reviewer of my doctoral thesis. I would like to thank both of them for all the time and effort they dedicated to a thorough review of my dissertation and for their helpful, valuable and accurate remarks as well as encouraging comments.

I want to list here and express my gratitude to all my coauthors and people with whom I collaborated during my PhD studies. I am thankful to Tero Harju, Miika Langille, Vladimir Rogojin, Bogdan Iancu, Cristian Gratie, Sergiu Ivanov, Luca Manzoni, Antonio E. Porreca, Charmi Panchal, Eugen Czeizler and Diana Gratie.

I would also like to express my gratitude to all current and former members of the Computational Biomodelling Laboratory at Åbo Akademi University (COMBIO) with whom I felt at home. Thanks for being my second family in Finland.

I gratefully acknowledge the financial support of my doctoral studies and research provided to me by the Turku Centre for Computer Science (TUCS), Åbo Akademi foundation and the Otto A. Malm foundation.

I am especially grateful to Bogdan Iancu and Diana Gratie without whom this journey would not be as joyous and pleasant as it was. Thanks for making all the great memories that thinking about every single one of them brings smile to my face and reminds me that I have always had friends to rely on and to be happy with.

I am extremely thankful to my parents who from the very early stage of life taught me to be curious and to pursue my dreams, thanks for always being there for me and thanks for the unconditional love and support you provided me which gave me the strength and confidence to never ever be afraid of dreaming big and following my heart.

I am also grateful to my first ever friend and research collaborator, my brother, Ali. I greatly treasure all our moments together and all our joint mischievous little projects. With you I learned to be creative, bold and adventurous.

I am also thankful to my beloved husband, Shahrokh, with whom I learned the value of a great companion. Thanks for helping me grow and for not letting my smile ever die. Thanks for being my soft place to fall.

Sepinoud Azimi
Turku, September 2015

List of original publications

i. Sepinoud Azimi, Tero Harju, Miika Langille, Ion Petre, Vladimir Rogojin, Directed overlap-inclusion graphs as representations of ciliate genes. Fundamenta Informaticae 110(1-4), 29-44, 2011.

ii. Sepinoud Azimi, Tero Harju, Miika Langille, Ion Petre, Simple gene assembly as a rewriting of directed overlap-inclusion graphs. Theoretical Computer Science 454, 30-37, 2012.

iii. Sepinoud Azimi, Ion Petre, The reduction power of simple operations for gene assembly in ciliates. Discrete Mathematics and Computer Science 1, 23-36, 2014.

iv. Sepinoud Azimi, Bogdan Iancu, Ion Petre, Reaction system models for the heat shock response, Fundamenta Informaticae, IOS Press, 131, 1–14, 2014.

v. Sepinoud Azimi, Cristian Gratie, Sergiu Ivanov, Luca Manzoni, Ion Petre, Antonio E. Porreca, Complexity of model checking for reaction systems, Submitted, 2015.

vi. Sepinoud Azimi, Cristian Gratie, Sergiu Ivanov, Ion Petre, Dependency graphs and mass conservation in reaction systems, Theoretical Computer Science 598, 23–39, 2015..

Contents

Chapter 1

Introduction

Should we name only one ingredient for developing science, undoubtedly it would be human curiosity which has won the crown throughout history. Indeed one of the unlimited, ever growing sources of inspiration which have satisfied the desire why and how things work and evolve has been the mother nature from the very starting point of life.

Knowledge of how information is processed in nature, or how nature does the computation has led to the birth of *natural computing* as an interdisciplinary field of science which marries life and computation [1].

The first signs of emerging *natural computing* as a discipline probably can be traced back to Turing, see [74]. Natural computing continued to grow further by the foundation of *finite automata theory* which had its roots in modelling neural networks, see [57, 63]. However, it was Adleman who brought natural computing into spotlight in 1994, see [1], by using DNA molecules to solve an instance of *Hamiltonian path problem*. Natural computing on one hand focuses on developing computing models, methods and the technologies inspired by nature and on the other hand studies how information is processed in nature [38]. The first direction, interested in the nature-inspired models, resulted in emergence of various fields of study, e.g. *molecular computing* [23], *evolutionary computing* [42], *neural networks* [46] and *quantum computing* [47]. The second direction consists of the research done on abstract models which describe the information processes that take place in nature; *cellular automata* [21], *membrane systems* [62] and *reaction systems* [36] are some examples of this research approach.

Natural computing can also be categorized under two main streams: *quantitative computational modeling* and *qualitative computational modeling* [50]. Quantitative computational modeling concerns with the numerical aspects of a biological phenomenon; *ODE-based models* are excellent examples of such models where the modeler mainly focuses on the initial concentrations, kinetic rates, stoichiometries, etc. Qualitative computational

modeling is mostly interested in the dynamics and interactions between different components of a biological system, for example, *biochemical networks* are widely used as a tool in the qualitative approach where the modeler is mainly interested in how chemicals influence each other and how they control the total dynamic of the system.

The studies in this thesis are also inspired by the computations taking place in nature. In the first direction of our research we are focusing on a nature-inspired process called *gene assembly* in ciliates whereas in the second direction we are studying *reaction systems*, see [36], as a modeling framework with its rationale built upon the biochemical interactions happening within a cell.

The process of gene assembly in ciliates has attracted a lot of attention as a research topic in the past one and a half decades. Two main modeling frameworks have been initially proposed in the end of 1990s to capture ciliates' gene assembly process, namely *intermolecular model* [55] and *intramolecular model* [69]. They were followed by other model proposals such as template-based assembly (see [68] and [6]) and DNA rearrangement pathways recombination models [5]. The intermolecular model assumes that several molecules take part in the gene assembly process whereas, in the intramolecular model, all assembly operations happen within a single molecule and the molecule arranges its gene without interacting with any other gene. Although these two models take different approaches toward assembling a ciliate gene, both can successfully provide a strategy to arrange any given gene which indicates the completeness of both models, see [32].

Various paths have been traversed while studying gene assembly process in ciliates. While many studies considered the assembly power of both intra- and inter- molecular models [18, 19, 32, 33], the others focused on their computational power and universality, see [49, 55, 56]. Gene assembly in ciliates has also been studied in relation with formal language theory, see for example [25, 24]. Parallel application of gene assembly operations as a different research direction has also been the center of various studies, see [3, 4, 2].

In our research we focused on a variation of intramolecular model called *simple gene assembly model*, first introduced in [43], which only considers the simplest possible folds in the assembly process. We propose a new framework called *directed overlap-inclusion (DOI) graphs* to capture the gene assembly process in ciliates and define simple operations for this new framework. We further study different properties of these types of graphs.

Reaction systems, first introduced in [36], is another nature-inspired modeling framework that is studied in this thesis. Reaction systems rationale is based upon two main regulation mechanisms, *facilitation* and *inhibition*, in a cell which control the interaction between biochemical reactions. Intuitively, a reaction is enabled when all components needed to

facilitate the reaction are present and all components which inhibit such a facilitation are absent from the environment. The interactions in a reaction system are governed by two main assumptions: *threshold assumption* and *no permanency assumption*. *Threshold assumption* indicates that if an element is a part of the environment, then it is present in abundance and *no permanency assumption* states that an element vanishes from the environment if it is not sustained by any reaction of the system.

Reaction systems, although a very young research topic, has attracted a lot of attention in the past few years and has been developed extensively in both theoretical and practical directions. In [16], an extended version of reaction systems, called *reaction systems with duration*, is proposed to account for the decay of an entity in the system. This new structure proves that reaction systems can efficiently be extended to include more information depending on the dynamics of the given system.

Many interesting results have also been proven for those reaction systems with reactions with minimal number of resources, i.e. *minimal reaction systems*, see for example [29, 72].

The complexity of reaction systems has also been the center of various studies, for example [39] shows that finding global attractors of a reaction system as well as deciding the presence of cycles are PSPACE-complete. The complexity of occurrence and convergence problems and finding fixed points of a reaction system is discussed in [41] and [40] respectively.

The application of reaction systems in different disciplines has also been quite extensive, see for example [22, 73].

The research done on reaction systems by no means is limited to what presented here, see for example [48, 61, 52, 64, 70, 71, 7, 11].

In this thesis we used the reaction systems as a modeling tool to build biological models and also defined various concepts inspired by biology, e.g., mass conservation, steady state, periodicity, etc., to do model checking of the reaction systems based models. We also discussed model checking in the reaction systems framework.

The structure of this thesis is as follows:

In Chapter 2, we provide the preliminary background information on different modeling frameworks proposed to capture the gene assembly process in ciliates.

In Chapter 4, we define related assembly operations for each modeling framework. We also present the two new graph-based modeling frameworks to represent ciliates' genes. We further define simple gene assembly operations and state various results regarding the properties of two new frameworks separately.

In Chapter 5, we introduce reaction systems framework and briefly discuss modelling biochemical networks with reaction systems and discuss their

model checking. Finally, we introduce a software designed for simulating interactions of a reaction systems based model.

In Chapter 6, we briefly state the original contributions of this thesis, and finally in Chapter 7, we conclude with some discussion and provide some directions for future related studies.

Chapter 2

Molecular models for gene assembly in ciliates

Ciliates, originated about 2×10^9 years ago, belong to an ancient group of unicellular eukaryotes which can be found anywhere humid from lakes to rivers to even damp soil, see [14, 65]. Ciliates survive almost any environmental conditions and this makes them suitable candidates as subjects for scientific research. Ciliates are identified by two main characteristics: (1) having hairlike cilia to help them move and grab food and (2) possessing two types of nuclei, one responsible for producing RNA transcription and the other active only in the process of sexual conjugation [14]. The two types of nuclei are *germline nucleus (micronucleus)* and *somatic nucleus (macronucleus)*. The micronucleus contains protein coding DNA segments called *macronuclear destined segments (in short* MDS*s)* which are separated by non-protein coding DNA blocks called *internally eliminated sequences (in short* IES*s)*. Each MDS possesses two specific sequences of nucleotides at its beginning and end which are called *incoming* and *outgoing pointer* respectively. The outgoing pointer of one MDS matches the incoming pointer of another unique MDS. The MDSs in the micronucleus are in scrambled order while the macronucleus has them in an orthodox order, for more information see [66].

Ciliates can reproduce either by cell division or by sexual conjugation. They choose the second way to reproduce whenever food is not sufficiently available in the environment. In the process of sexual conjugation, a cytoplasmic bridge is made between two cells, then micronuclei undergo meiosis and the genetic information is exchanged through the bridge. Each micronuclei contains two copies of every chromosome, i.e. they are diploid. During the meiosis every diploid micronuclei divides two times to form four haploid micronuclei where each of them contains only one copy of the chromosomes. The new haploid micronuclei are fused in the next step to form a new diploid

micronucleus. Finally the macronucleus is divided to complement two micronuclei and by doing so the process of conversion of a germline genome to a somatic one is completed. This process is called *gene assembly*, for more information see [14, 65]. The process of gene assembly in ciliates is known to be the most involved DNA manipulation in a living organism, see [69, 44]. Experiencing such a complex transformation is a reason why ciliates are computationally interesting subjects to study.

Two different models for gene assembly in ciliates have been proposed which have become the basis of various studies over the last one and a half decades, i.e. the *intermolecular model* and the *intramolecular model*. In the intermolecular model, introduced in [55, 56], there may be two molecules involved in the process of gene assembly where some parts of their sequences would be exchanged through recombination. The goal in this model is to both identify the pointers and the process in which the pointers are used to facilitate the gene assembly. In the intramolecular model, see [67, 34], on the other hand, only one molecule participates in the process of DNA manipulation. Therein, the molecule folds on itself and swaps some parts of its sequence through recombination.

2.1 The intermolecular model for gene assembly

The *intermolecular model* for gene assembly in ciliates was first introduced in [55]. In this model more than one copy of the molecule is involved in the assembly process. A molecule, in this model, is represented as a word, e.g. x, u, w etc., over the set of alphabets Σ. A molecule can be either *linear* or *circular*; by a circular molecule we refer to any circular permutation of its corresponding sequence of nucleotide base pairs. We denote the circular molecule w by $w^\bullet$. Gene assembly in intermolecular model is facilitated through two different operations:

i. *intramolecular recombination* can be applied on either a linear or a circular molecule:

   - applying intramolecular recombination on a linear molecule of the form $uxwxv$ is the union of a linear molecule vxu and a circular molecule $wx^\bullet$, this process is presented in Figure 2.1;
   - applying intramolecular recombination on a circular molecule of the form $uxwxv^\bullet$ is the union of a circular molecule $vxu^\bullet$ and a circular molecule $wx^\bullet$, this process is presented in Figure 2.2.

   Also note that intramolecular recombination is reversible. In the case of intramolecular recombination the molecule aligns on two occurrences of sequence x and an exchange of strands lead to formation of a linear and a circular molecule.

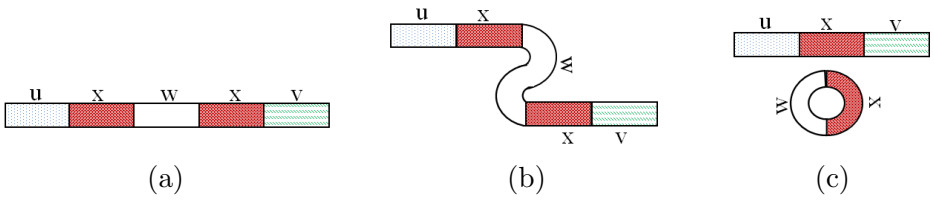


Figure 2.1: An intramolecular recombination: (a) the input linear molecule of the form $uxwxv$ (b) gets aligned on x and (c) the recombination produces a linear molecule uxv and a circular molecule $wx^{\bullet}$.

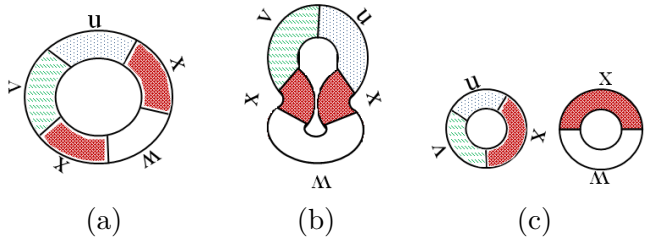


Figure 2.2: An intramolecular recombination: (a) the input circular molecule of the form $uxwxv^{\bullet}$ (b) gets aligned on x and the recombination produces a circular molecule $uxv^{\bullet}$ and a circular molecule $wx^{\bullet}$.

ii. *intermolecular recombination*, also reversible, is applied on two given linear molecules of form u_1xv_1 and u_2xv_2; applying intermolecular recombination on these two molecules results in the formation of two linear molecules of form u_1xv_2 and u_2xv_1 by an strand exchange on the two molecules common sequence and rewriting the corresponding sequences to each input molecule.

For more information on the basic definitions of intermolecular operations we refer to [55, 56]. A series of consecutive or parallel intra- and intermolecular recombination operations is called a *gene assembly process.*

Example 1. *An assembly of actin I gene in Sterkiella nova with intermolecular operations is illustrated in Figures 2.3- 2.12.*

2.2 The intramolecular model for gene assembly

The intramolecular model for gene assembly in ciliates was first introduced in [34, 67]. In this model, unlike the intermolecular model, only one molecule is involved in the process of gene assembly. In this process the IESs are

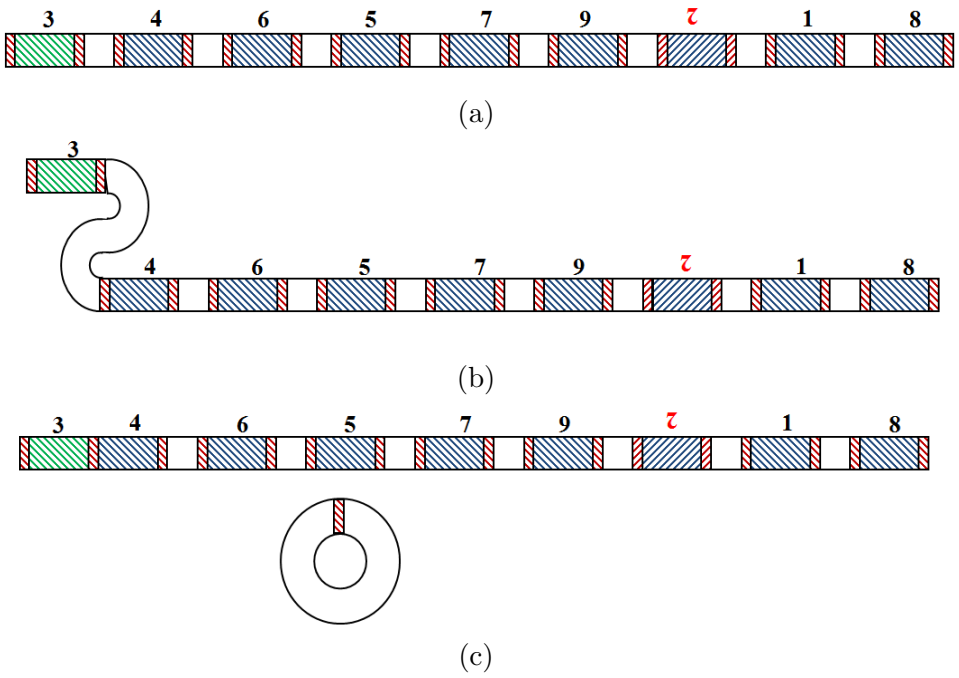


Figure 2.3: (a) The input molecule; (b) the molecule folds to align MDS 3 and MDS 4 and (c) as a result one linear and one circular molecules are produced.

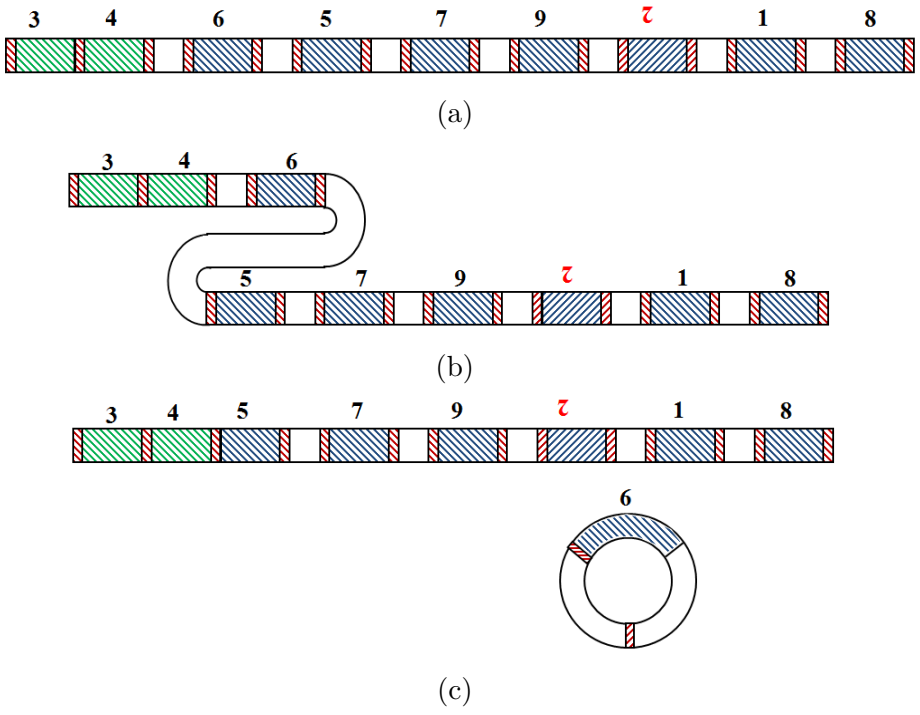


Figure 2.4: (a) The input molecule; (b) the molecule folds to align the composite MDS 3-4 and MDS 5 and (c) as a result one linear and one circular molecules are produced.

excised and the MDSs are spliced on their pointers to be rearranged, see [34, 67]. The rearrangement of gene blocks is done either through excision or by inversion. Two MDSs are spliced when the outgoing pointer of one MDS is aligned with the incoming pointer of the other one. To facilitate the process of gene assembly three operations are proposed in [67, 32]. The three operations are:

i. *loop, direct repeat excision* (ld in short), is applicable when the molecule has a direct repeat pattern (p, p) for pointer p, i.e., two identical sequence of nucleotides, represented by p, are positioned on two different places on the same strand [67, 32]. The operation is illustrated in Figure 2.13. There are two possible ld cases:

   - **Simple ld operation**: The only part separating two pointers is an IES. Applying such an ld operation creates two molecules, one linear and one circular. One copy of the pointer p glues two ends of the part of molecule which contains only one IES and

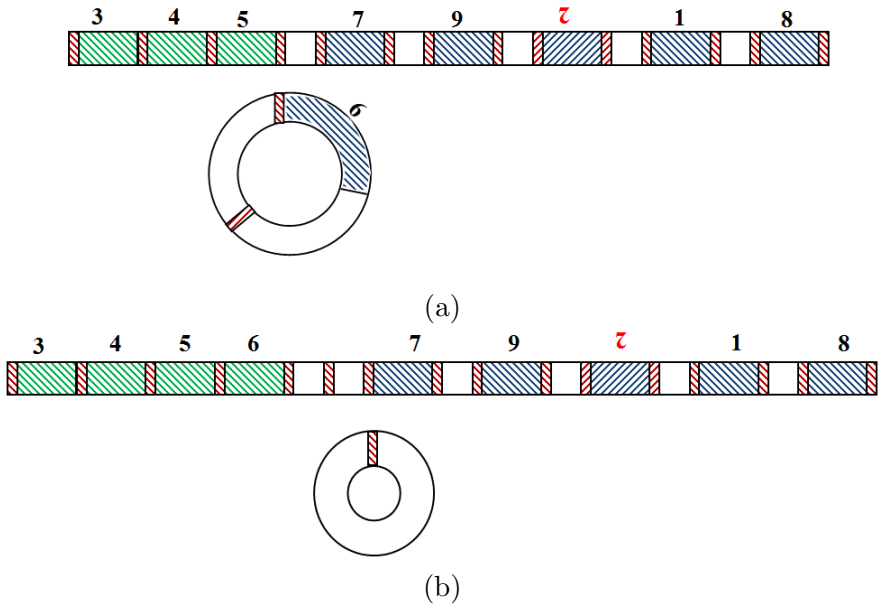


Figure 2.5: (a) The input molecule; (b) the molecule folds to align the composite MDS 3-4-5 and MDS 6 and (c) as a result one linear and one circular molecules are produced.

the circular molecule is excised. The second copy of p binds two MDSs which have p as their incoming or outgoing pointer; and a composite MDS is produced in this way.

- **Boundary ld operation**: In this case the two copies of pointer p are at two ends of the molecule. By applying a boundary ld operation still two molecules, one circular and one linear, are produced. However, in this case the linear molecule contains one composite IES and one copy of the pointer p, while the rest of molecule and the other copy of p are included in the circular molecule.

ii. *hairpin, inverted repeat excision/reinsertion* (hi in short), is applicable to the molecules folded into a hairpin and aligned on an inverted repeat pattern $(p, \bar{p})$ of pointer p. The hi operation applied on such a molecule is presented in Figure 2.14. As the result of applying hi operation, the segment of the molecule placed between two occurrences of p is inverted. In this process two MDSs with the pointer p either as their incoming or outgoing pointer are spliced together and form a new composite MDS while similarly the other occurrence of p glues two

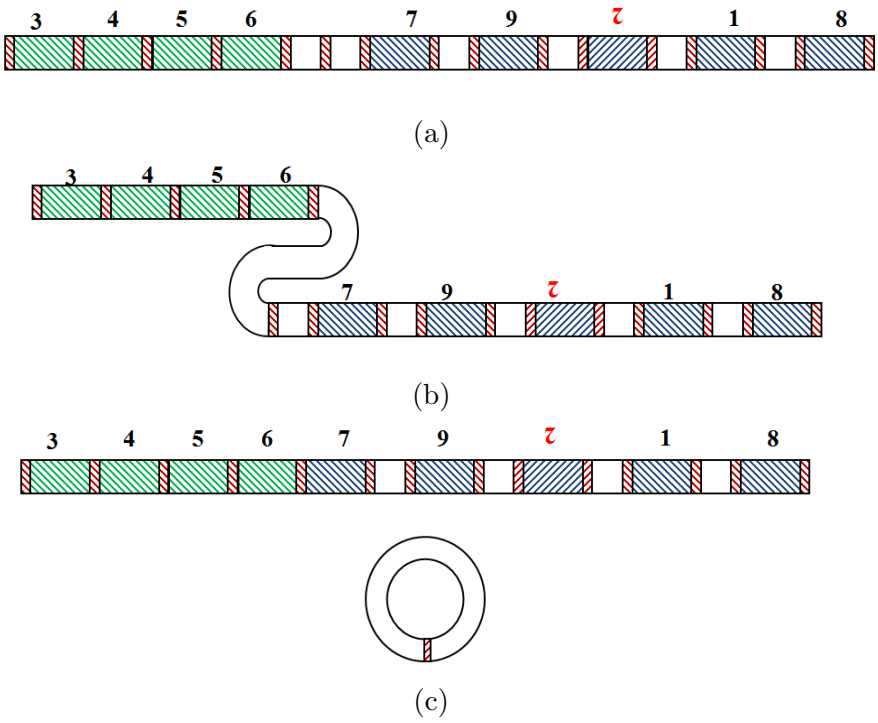


Figure 2.6: (a) The input molecule; (b) the molecule folds to align the composite MDS 3-4-5-6 and MDS 7 and (c) as a result one linear and one circular molecules are produced.

IESs bounded by two copies of p to build a new composite IES. The operation is shown in Figure 2.14.

iii. *double loop, alternating direct repeat excision/reinsertion* (dlad in short), is applied when the molecule is folded into a double loop where each loop is aligned by a pair of direct repeat, in another word, the molecule has an alternating direct repeat (p, q, p, q). In this case, when the operation is applied, the molecule folds into two loops, one aligned by (p, p) and the other one by (q, q). Consequently, excision and reinsertion happens simultaneously and the result would be one molecule in which, the MDSs containing either a copy of p or a copy of q are spliced together to form a new composite MDS. Similarly, the IESs bound by the occurrences of p and q are spliced together and make a new composite IES. As a result, the occurrences of p and q are no longer pointer but parts of the new composite MDS and IES. This process is presented in Figure 2.15.

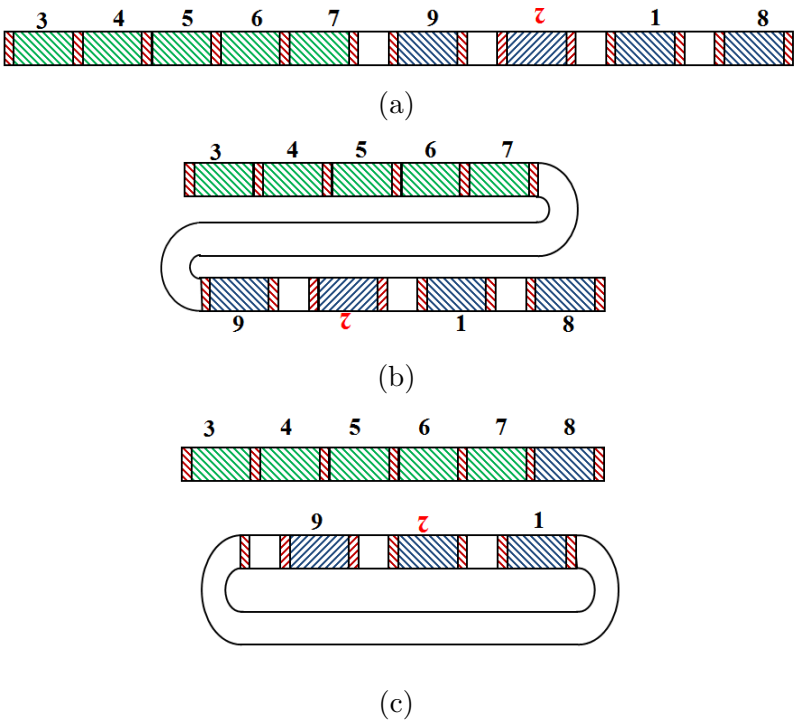


Figure 2.7: (a) The input molecule; (b) the molecule folds to align the composite MDS 3-4-5-6-7 and MDS 8 and (c) as a result one linear and one circular molecules are produced.

Example 2. *An assembly of actin I gene in Sterkiella nova is illustrated in Figures 2.16- 2.22.*

2.3 Template guided recombination

The process of excision of the IESs and rearrangement of the MDSs which lead to the gene assembly is guided by repeated sequences of nucleotides, also known as pointers. In the proposed gene assembly models, the length of nucleotide sequences are considered to be long enough to facilitate the recombination. However, this may not be always the case since the length of sequences vary between 3 and 20 nucleotides, see [68]. Consequently, some nucleotides can be too short to result in an accurate alignment. A *template guided recombination* was proposed in [68] to tackle this problem. In what follows we briefly discuss two variations of this framework: one using a double stranded DNA molecule and the other one a double stranded RNA molecule as the template.

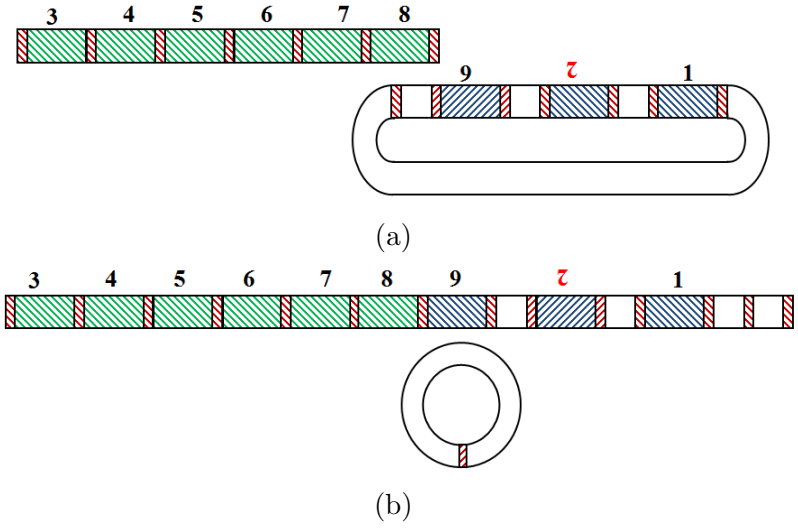


Figure 2.8: (a) Two input molecules (one linear and one circular); (b) the molecules align on MDS 3-4-5-6-7-8 and MDS 9 and as a result one linear and one circular molecules are produced.

DNA guided recombination In this method, the recombination is guided through an old macronuclear molecule which acts as a template. The template is placed between two micronuclear molecules which are to be assembled. The result recombined molecules contain some parts of the template and the template itself is reconstituted during the recombination. The template guided recombination facilitates a precise cut and by doing so it prevents any kind of mutation during the rearrangement process, for more information see [68].

RNA guided recombination This method is proposed as an alternative to the one of [68] in [6]. In *RNA guided recombination* a single stranded RNA plays the role of the template rather than a DNA one. Unlike the previous method, RNA is located on top of the two molecules which are to be assembled. The main difference between this method and the previous one lies in the fact that although the RNA molecule guides the alignment, it does not participate in the actual recombination process, for more information see [6]. An experimental study presented in [60] supports the efficiency of the method proposed in [6].

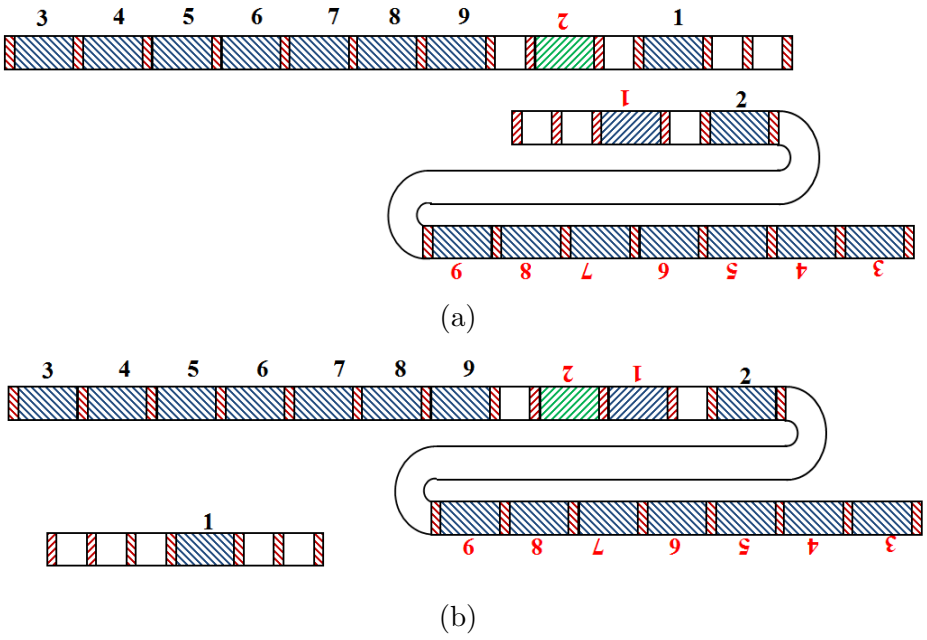


Figure 2.9: (a) Two linear input molecules ; (b) the molecules align on MDS 2 and MDS 1 and as a result two linear molecules are produced.

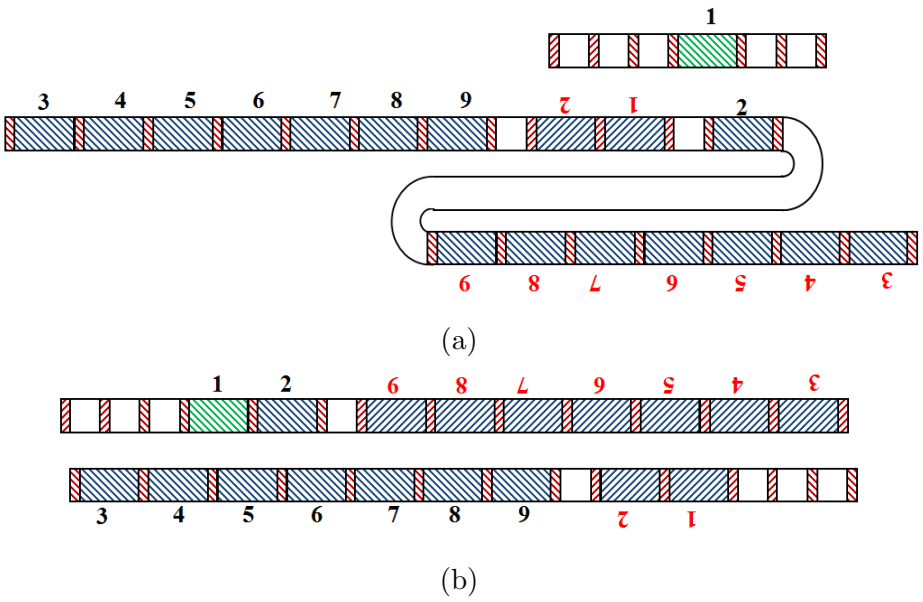


Figure 2.10: (a) Two linear input molecules ; (b) the molecules align on MDS 1 and MDS 2 and as a result two linear molecules are produced.

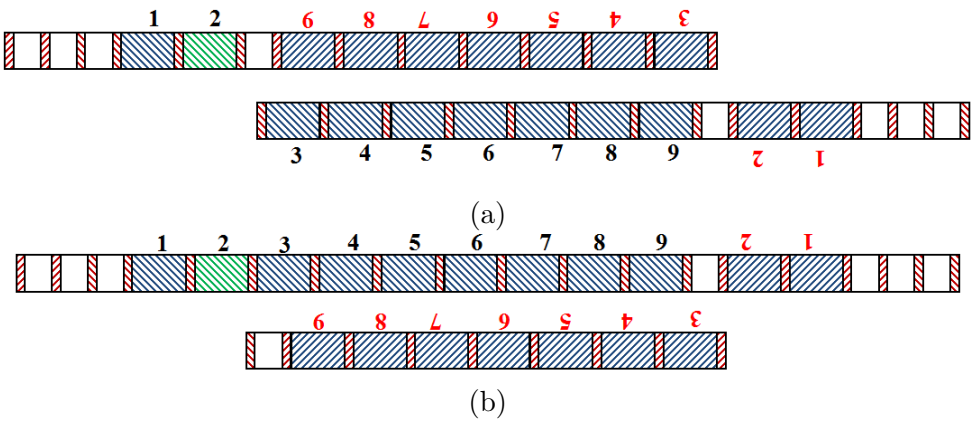


Figure 2.11: (a) Two linear input molecules ; (b) the molecules align on MDS 2 and MDS 3 and as a result two linear molecules are produced.

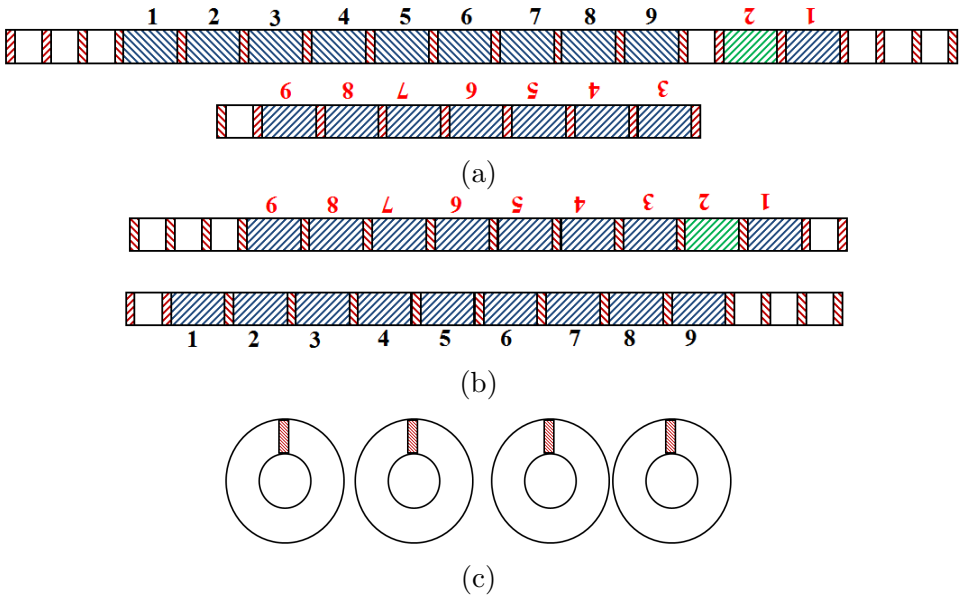


Figure 2.12: (a) Two linear input molecules ; (b) the molecules align on MDS 2 and MDS 3 and as a result two linear molecules are produced, the gene is now assembled; (c) the circular molecules excised throughout gene assembly.

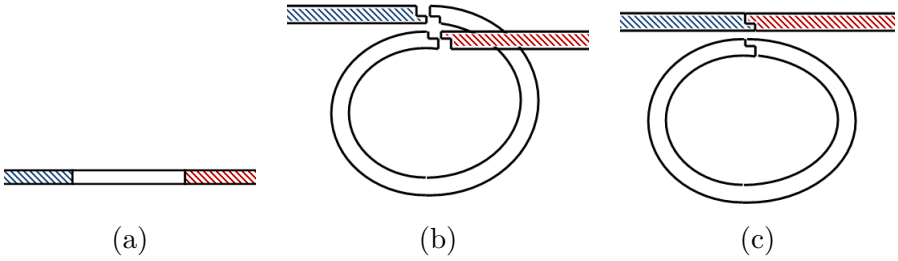


Figure 2.13: The ld operation: (a) The input molecule, (b) loop-fold to align the common pointers, (c) as a result, one circular molecule gets excised from molecule.

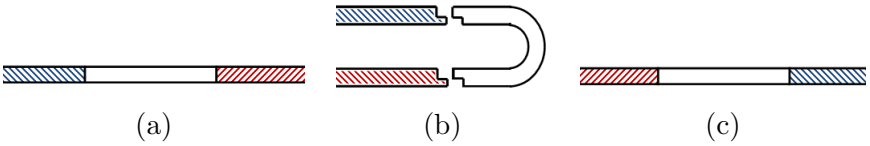


Figure 2.14: The hi operation: (a) The input molecule, (b) haipin-fold to align the common pointers, (c) as a result, a segment of the molecule gets inverted.

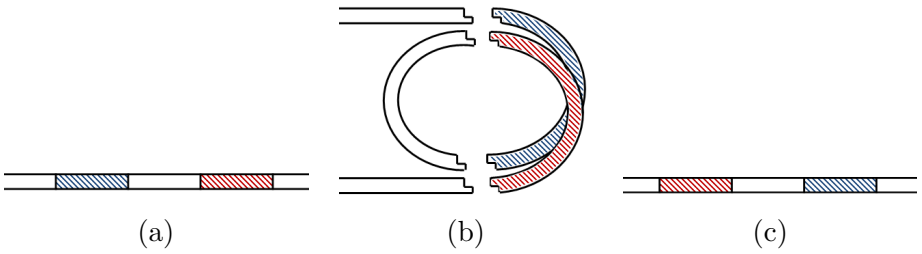


Figure 2.15: The dlad operation: (a) The input molecule, (b) double loop-fold to align the common pointers, (c) as a result, two segments of the molecule get inter-changed.

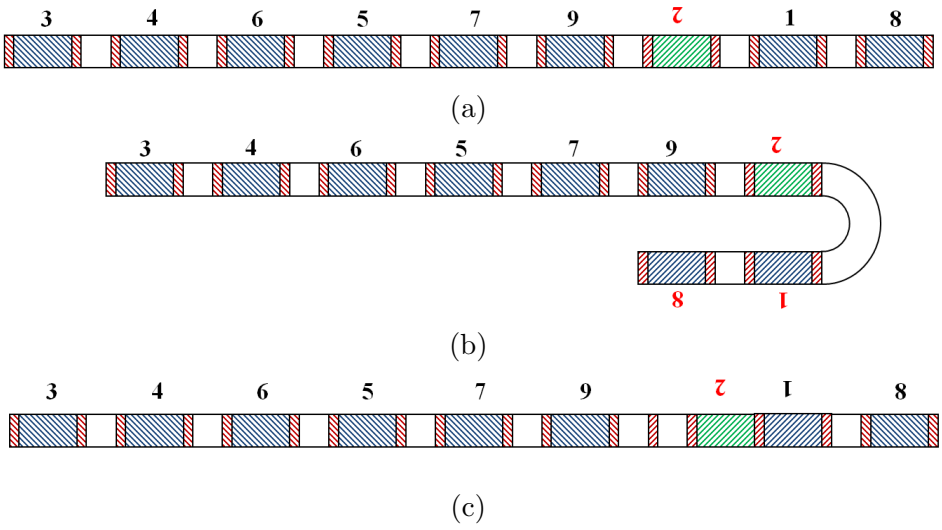


Figure 2.16: (a) The input molecule; (b) the hairpin fold to align the pointers in MDS 1 and MDS 2 and (c) as a result, a segment of the molecule gets inverted.

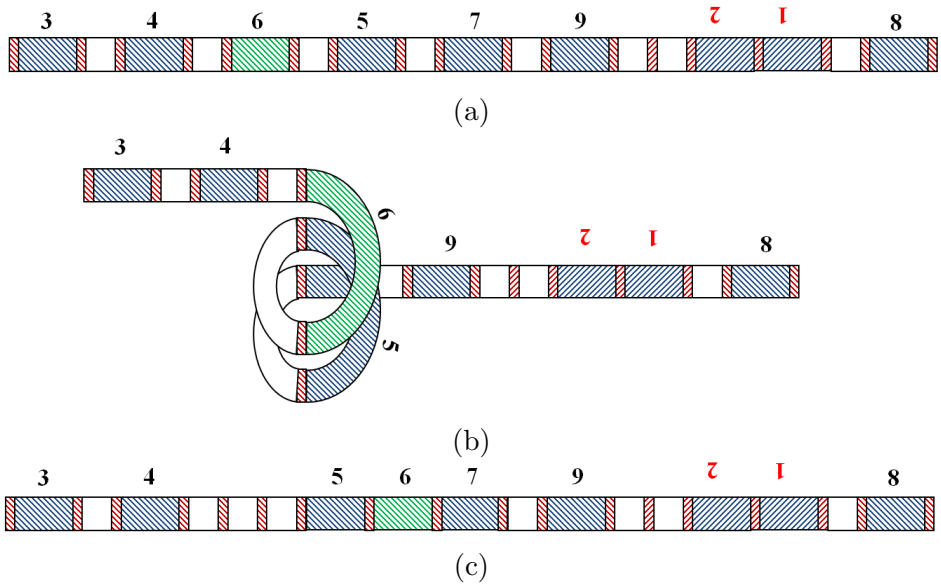


Figure 2.17: (a) The input molecule; (b) the double loop fold to align the pointers in MDS 5 and MDS 6 and (c) as a result, two segments of the molecule get inter-changed.

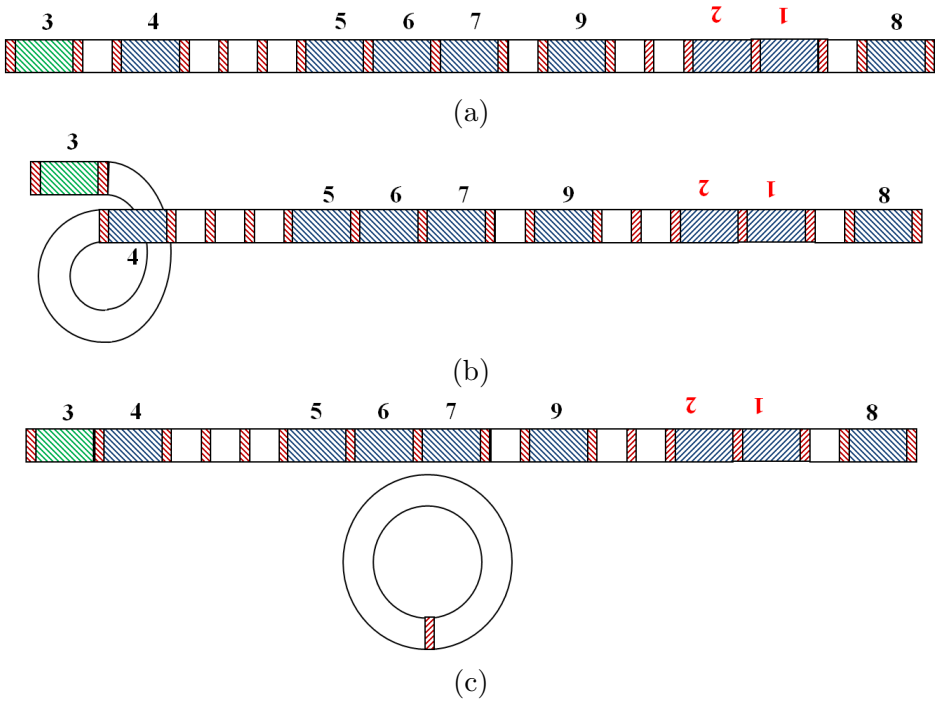


Figure 2.18: (a) The input molecule; (b) the loop fold to align the pointers in MDS 3 and MDS 4 and (c) as a result, one circular molecule gets excised from the original DNA molecule.

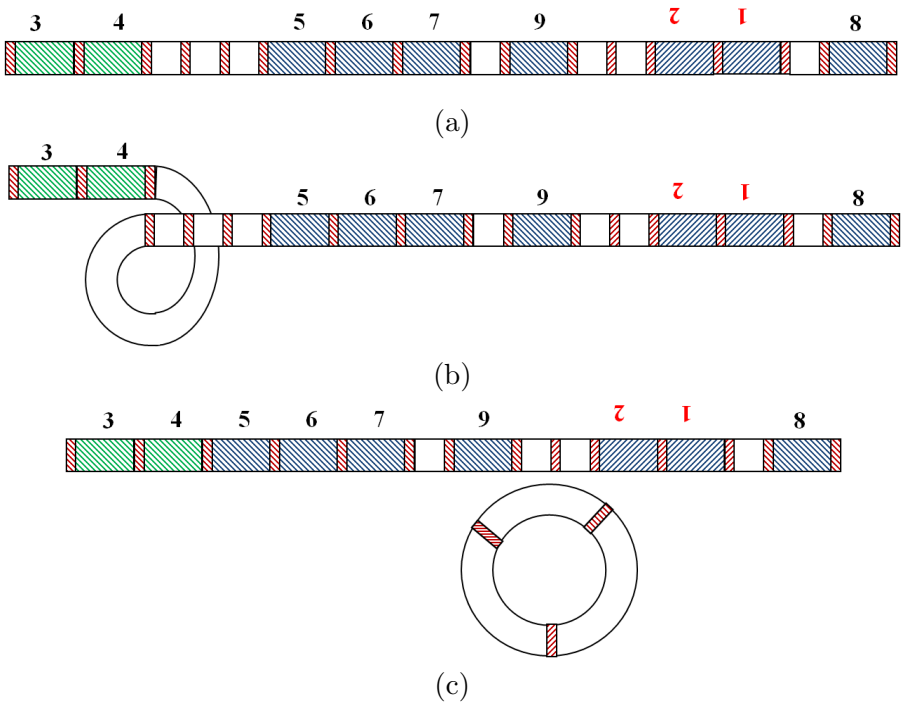


Figure 2.19: (a) The input molecule; (b) the loop fold to align the pointers in the composite MDS 3-4 and the composite MDS 5-6-7 and (c) as a result, one circular molecule gets excised from the original DNA molecule.

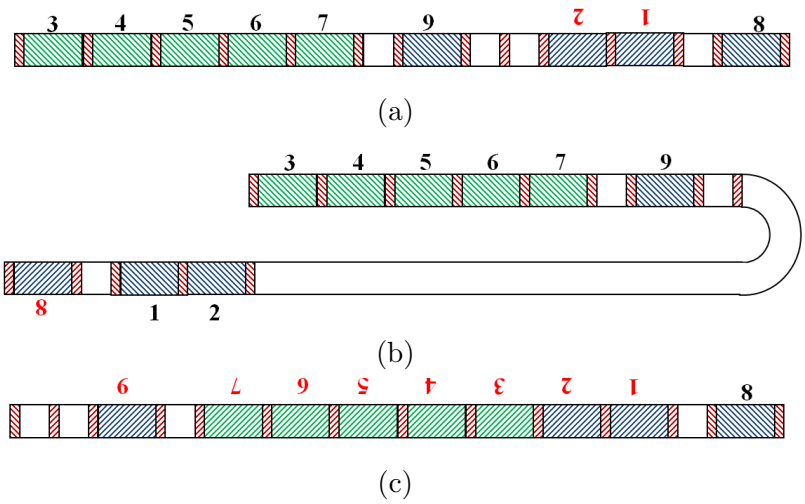


Figure 2.20: (a) The input molecule; (b) the hairpin fold to align the pointers in the composite MDS 1-2 and the composite MDS 3-4-5-6-7 and (c) as a result, a segment of the molecule gets inverted.

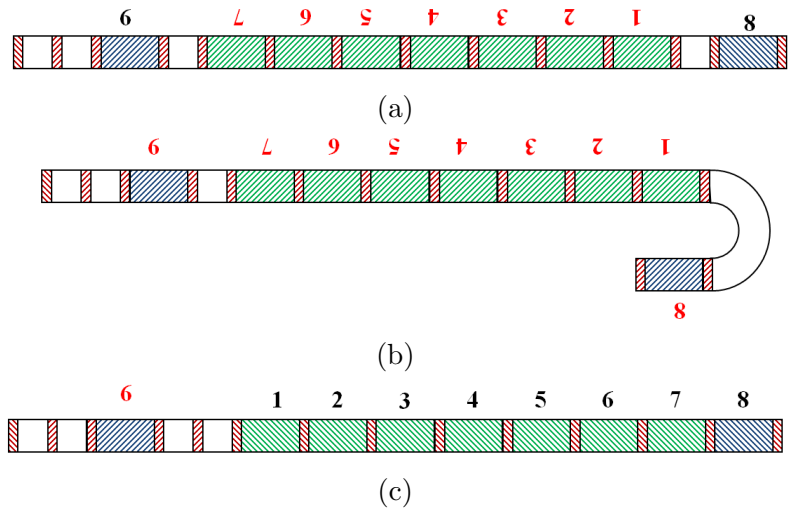


Figure 2.21: (a) The input molecule; (b) the hairpin fold to align the pointers in the composite MDS 1-2-3-4-5-6-7 and MDS 8 and (c) as a result, a segment of the molecule gets inverted.

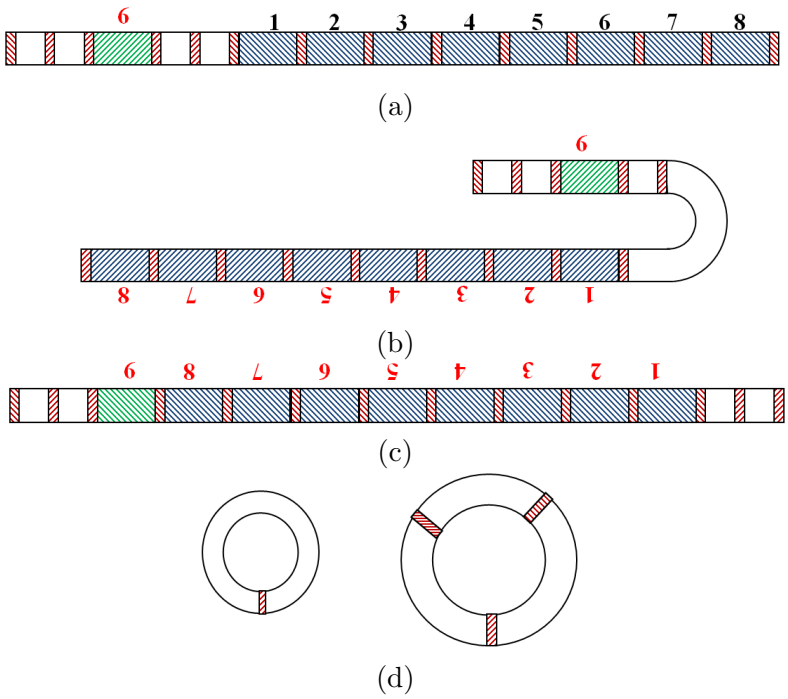


Figure 2.22: (a) The input molecule; (b) the hairpin fold to align the pointers in the composite MDS 1-2-3-4-5-6-7-8 and MDS 9 and (c) as a result, a segment of the molecule gets inverted, the gene is now assembled; (d) the circular molecules excised throughout gene assembly.

Chapter 3

Mathematical models for the structure of ciliate genes

In over a decade, various mathematical frameworks have been proposed to illustrate the intramolecular gene assembly in ciliates. In this section we briefly review the ones more relevant to our study.

3.1 MDS descriptors

One of the first formalisms to represent ciliates' genes is called MDS *descriptors*, introduced in [34] and [67], in which each gene is illustrated as a sequence of ordered pairs of pointers. Each pair corresponds to an MDS in the gene representation where the first pointer corresponds to the incoming pointer of the MDS and the second one corresponds to its outgoing pointer. In the case of inverted MDSs, the corresponding pair would be signed.

The pairs of pointers are defined over the set $\Pi_k = \Delta_k \cup \bar{\Delta}_k$, where $\Delta_k = \{2, 3, \ldots, k\} \cup \{b, e\}$ is an alphabet such that b and e represent the beginning and end markers respectively and $2, 3, \ldots, k$ are the $k-1$ pointers occurring in the gene. By beginning and end pointers we mean the incoming pointer of the first MDS and the outgoing pointer of the last one respectively when the MDSs are in an orthodox order. The set of inverted alphabets is denoted by $\bar{\Delta}_k = \{\bar{2}, \bar{3}, \ldots, \bar{k}\} \cup \{\bar{b}, \bar{e}\}$.

Let $\Gamma_k = \{(i, j) | i \in \Delta_k \cup \{b\} \text{ and } j \in \Delta_k \cup \{e\}\}$, then (i, j) is an ordered pair corresponding to an MDS where its incoming pointer is i and its outgoing one is j. Moreover, the set $\bar{\Gamma}_k = \{(\bar{j}, \bar{i}) | i \in \Delta_k \cup \{b\} \text{ and } j \in \Delta_k \cup \{e\}\}$ corresponds to the set of inverted MDSs of a gene. Let $\Gamma_k^{\maltese} = (\Gamma_k \cup \bar{\Gamma}_k)^*$ be a set of strings, then a string $\delta \in \Gamma_k^{\maltese}$ is called an MDS *descriptor*.

Let $\delta = (i_1, j_1), \ldots, (i_k, j_k)$ be an MDS descriptor, the set of all pointers occurring in δ is denoted by $\mathsf{dom}(\delta)$, i.e. $\mathsf{dom}(\delta) = \{i_1, j_1, \ldots, i_k, j_k\}$. We say δ is *realistic* if any pointer $p \in \Delta_k$ has either 0 or 2 occurrences in

Figure 3.1: Illustration of the MDS descriptor corresponding to Actine I gene.

$\mathsf{dom}(\delta)$. A pointer $p \in \Delta_k$ is called *negative* if the occurrences in $\mathsf{dom}(\delta)$ are either both inverted or both non-inverted, p is called *positive* otherwise.

Example 3. *The* MDS *descriptor corresponding to actin I gene in Sterkiella nova is* $\delta = (3,4)(4,5)(6,7)(5,6)(7,8)(9,e)(\bar{3},\bar{2})(b,2)(8,9)$. *An illustration of Actine I gene is presented in Figure 3.1 where the* MDS *labeled* 2 *is an inverted* MDS. *Note that* $\delta_1 = (2,5)(5,6)(3,4),(\bar{e},\bar{6})(4,2)(b,3)$ *is also an* MDS *descriptor. Although* δ' *does not correspond to any known ciliate gene but it fits the definition of an* MDS *descriptor.*

3.2 Signed strings

Next level of abstraction proposed to present ciliates' gene structure is called *legal strings*, introduced in [33]. A string u over alphabet Δ_k is called a legal string if for any $a \in \mathsf{dom}(u)$, there are exactly two occurrences of a in u. The legal string corresponding to a gene can easily be abstracted from its corresponding MDS descriptor. Let $f : \Gamma_k^{\maltese} \to \Pi_k^*$ be a mapping from the set of MDS descriptors with k pointers onto the set of strings with k pointers and $\delta = (i_1, j_1) \dots (i_k, j_k) \in \Gamma_k^{\maltese}$, we define $f(\delta) = i_1 j_1 \dots i_k j_k$.

We say that u is a *realistic legal string* if there exists a realistic MDS descriptor such that $f(\delta) = u$.

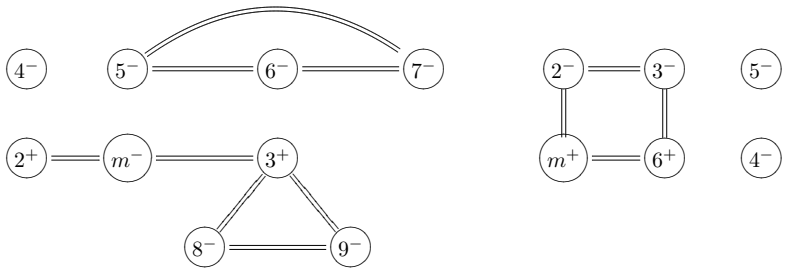
Example 4. *The realistic string corresponding to Actine I gene is* $u = 34456756789e\bar{3}\bar{2}b289$. *The legal string corresponding to* δ_1 *of Example 3.1 is* $u_1 = 255634\bar{e}\bar{6}42b3$.

3.3 Overlap graphs

The overlap relationships of a legal string pointers can be presented through an *overlap graph* (also known as *interlacement graphs*, see, e.g., [26]). The overlap-graph based pointer reduction system was introduced in [27, 33] to model gene assembly in ciliates through rewriting of overlap graphs. We say two pointers p and q overlap in a legal string u if either $pqpq$ or $qpqp$ is a substring of u. We denote the overlap relation between p and q with

$p \Rightarrow_u q$ if the first occurrence of p appears before the first occurrence of q in u, we denote it by $q \Rightarrow_u p$ otherwise. For a legal string u, its overlap graph $G = (V, \sigma, E)$ was defined as follows: $V = \mathsf{dom}(u)$, $\sigma : V \to \{+, -\}$ is the signing of vertices from V (i.e., if $p \in V$ is positive in the corresponding string u, then $\sigma(p) = +$, otherwise $\sigma(p) = -$) and $E = \{\{p, q\} | p \Rightarrow_u q \text{ or } q \Rightarrow_u p\}$. For any vertex $p \in V$, the set of its neighbours in G is denoted by $N_G(p)$ and defined as $N_G(p) = \{q \in V | (p, q) \in E\}$.

Example 5. *The overlap graph corresponding to actin I gene in Sterkiella nova is shown in Figure 3.2(a) and the overlap graph corresponding to* u_1 *of Example 3.2 is illustrated in Figure 3.2(b).*



(a) (b)

Figure 3.2: (a) The overlap graph G corresponding to actin I gene in Sterkiella nova, (b) the overlap graph corresponding to string $u_1 = 255634\bar{e}\bar{6}42b3$.

Chapter 4

Computational models for gene assembly in ciliates

The ultimate goal of defining frameworks to represent ciliates is to model their gene assembly process, therefore, in this chapter we present formal definitions of gene assembly operations in different frameworks that we have introduced in Chapter 2. We also introduce the notion of *simple gene assembly* and present two different modeling frameworks which are specifically designed to facilitate simple gene assembly in ciliate.

4.1 Gene assembly in different frameworks

In this section we formulate the formal definitions of gene assembly operations in different modeling frameworks.

4.1.1 Gene assembly with MDS descriptors

In this section we present formal definitions of gene assembly operations in MDS descriptor framework.

i. Id **operation** As mentioned in Section 2.2, there are two variations of Id operations, simple and boundary, the corresponding operations on MDS descriptors are defined as follows:

- Simple Id: let $p \in \Delta_k$ and $\delta \in \Gamma_k^{\Join}$, we say $\mathsf{Id}_p(\delta)$ is applicable if
  - p is a negative pointer of δ and
  - $\delta = \delta_1(q,p)(p,r)\delta_2$, such that $\delta_1, \delta_2 \in \Gamma_k^{\Join}$ and either $q, r \in \Delta_k$ or $q, r \in \bar{\Delta}_k$,

  then $\mathsf{Id}_p(\delta) = \delta_1(q,r)\delta_2$.

- Boundary ld: let $p \in \Delta_k$ and $\delta \in \Gamma_k^{\Phi}$, we say $\mathsf{ld}_p(\delta)$ is applicable if
  - p is a negative pointer of δ and
  - $\delta = (p, q)\delta_1(r, p)$, such that $\delta_1 \in \Gamma_k^{\Phi}$ and either $q, r \in \Delta_k$ or $q, r \in \bar{\Delta}_k$,

  then $\mathsf{ld}_p(\delta) = (r, q)\delta_1$.

The set of all ld operations is denoted by $\mathsf{Ld} = \{\mathsf{ld}_p \,|p \in \Pi_k\}$.

ii. hi **operation** Let $\delta \in \Gamma_k^{\Phi}$ and p a positive pointer in $\mathsf{dom}(\delta)$, then $\mathsf{hi}_p(\delta)$ is applicable in two following cases:

- $\delta = \delta_1(p, q)\delta_2(\bar{p}, \bar{r})\delta_3$ and
- $\delta = \delta_1(q, p)\delta_2(\bar{r}, \bar{p})\delta_3$,

where $\delta_1, \delta_2, \delta_3 \in \Gamma_k^{\Phi}$ and either $q, r \in \Delta_k$ or $q, r \in \bar{\Delta}_k$. The result of applying $\mathsf{hi}_p(\delta)$ would be $\delta_1\bar{\delta_2}(\bar{q}, \bar{r})\delta_3$ and $\delta_1(q, r)\bar{\delta_2}\delta_3$ respectively.

The set of all hi operations is denoted by $\mathsf{Hi} = \{\mathsf{hi}_p \,|p \in \Pi_k\}$.

iii. dlad **operation** Let $\delta \in \Gamma_k^{\Phi}$ and p, q negative pointers in $\mathsf{dom}(\delta)$, then $\mathsf{dlad}_{p,q}(\delta)$ is applicable in the following cases:

- $\delta = \delta_1(p, r_1)\delta_2(q, r_2)\delta_3(r_3, p)\delta_4(r_4, q)\delta_5$ and $\mathsf{dlad}_{p,q}(\delta) = \delta_1\delta_4(r_4, r_2)\delta_3(r_3, r_1)\delta_2\delta_5$,
- $\delta = \delta_1(p, r_1)\delta_2(r_2, q)\delta_3(r_3, p)\delta_4(q, r_4)\delta_5$ and $\mathsf{dlad}_{p,q}(\delta) = \delta_1\delta_4\delta_3(r_3, r_1)\delta_2(r_2, r_4)\delta_5$,
- $\delta = \delta_1(r_1, p)\delta_2(q, r_2)\delta_3(p, r_3)\delta_4(r_4, q)\delta_5$ and $\mathsf{dlad}_{p,q}(\delta) = \delta_1(r_1, r_3)\delta_4(r_4, r_2)\delta_3\delta_2\delta_5$,
- $\delta = \delta_1(r_1, p)\delta_2(r_2, q)\delta_3(p, r_3)\delta_4(q, r_4)\delta_5$ and $\mathsf{dlad}_{p,q}(\delta) = \delta_1(r_1, r_3)\delta_4\delta_3\delta_2(r_2, r_4)\delta_5$,
- $\delta = \delta_1(p, r_1)\delta_2(q, p)\delta_4(r_4, q)\delta_5$ and $\mathsf{dlad}_{p,q}(\delta) = \delta_1\delta_4(r_4, r_1)\delta_2\delta_5$,
- $\delta = \delta_1(p, q)\delta_3(r_3, p)\delta_4(q, r_4)\delta_5$ and $\mathsf{dlad}_{p,q}(\delta) = \delta_1\delta_4\delta_3(r_3, r_4)\delta_5$,
- $\delta = \delta_1(r_1, p)\delta_2(q, r_2)\delta_3(p, q)\delta_5$ and $\mathsf{dlad}_{p,q}(\delta) = \delta_1(r_1, r_2)\delta_3\delta_2\delta_5$.

The set of all dlad operations is denoted by $\mathsf{Dlad} = \{\mathsf{dlad}_{p,q} \,|p, q \in \Pi_k\}$.

We call $\phi = \phi_1 \circ \phi_2 \circ \ldots \circ \phi_n$ a *reduction strategy* if for any $1 \leq i \leq n$, $\phi_i \in \mathsf{Ld} \cup \mathsf{Hi} \cup \mathsf{Dlad}$. A reduction strategy ϕ is successful for an MDS descriptor δ if either $\phi(\delta) = (b, e)$ or $\phi(\delta) = (\bar{e}, \bar{b})$.

Example 6. *A reduction strategy for Actin I gene in Sterkiella nova is as follows:*

$\delta' = \mathsf{hi}_2(\delta) = (3,4)(4,5)(6,7)(5,6)(7,8)(9,e)(\bar{3},\bar{b})(8,9)$,
$\delta'' = \mathsf{hi}_3(\delta') = (\bar{e},\bar{9})(\bar{8},\bar{7})(\bar{6},\bar{5})(\bar{7},\bar{6})(\bar{5},\bar{4})(4,5)(8,9)$,
$\delta''' = \mathsf{dlad}_{7,6}(\delta'') = (\bar{e},\bar{9})(\bar{8},\bar{5})(\bar{5},\bar{4})(\bar{4},\bar{b})(8,9)$,
$\delta^4 = \mathsf{ld}_5(\delta''') = (\bar{e},\bar{9})(\bar{8},\bar{4})(\bar{4},\bar{b})(8,9)$,
$\delta^5 = \mathsf{ld}_4(\delta^4) = (\bar{e},\bar{9})(\bar{8},\bar{b})(8,9)$,
$\delta^6 = \mathsf{hi}_8(\delta^5) = (\bar{e},\bar{9})(b,9)$,
$\delta^7 = \mathsf{hi}_9(\delta^6) = (\bar{e},\bar{b})$.

Therefore, $\phi = \mathsf{hi}_9\,\mathsf{hi}_8\,\mathsf{ld}_4\,\mathsf{ld}_5\,\mathsf{dlad}_{7,6}\,\mathsf{hi}_3\,\mathsf{hi}_2$ *is a successful reduction strategy for* δ.

A reduction strategy for δ_1 *of Example 3.1 is as follows:*

$\delta_1' = \mathsf{dlad}_{2,3}(\delta_1) = (b,4)(\bar{e},\bar{6})(4,5)(5,6)$,
$\delta_1'' = \mathsf{hi}_6(\delta_1') = (b,4)(\bar{e},\bar{5})(\bar{5},\bar{4})$,
$\delta_1''' = \mathsf{ld}_5(\delta_1'') = (b,4)(\bar{e},\bar{4})$,
$\delta_1^4 = \mathsf{hi}_4(\delta_1''') = (\bar{e},\bar{b})$.

Therefore, $\phi_1 = \mathsf{hi}_4\,\mathsf{ld}_5\,\mathsf{hi}_6\,\mathsf{dlad}_{2,3}$ *is a successful reduction strategy for* δ.

4.1.2 Gene assembly with legal strings

In this section we present the formal definitions of string reduction operations. The advantage of this framework over the previous one is the substantially reduced number of its reduction cases.

snr operation: The *string negative rule (*snr*)* corresponds to ld operation and is defined as follows:

Let p be a pointer of legal string u, $\mathsf{snr}_p(u)$ is applicable if:

- p is a negative pointer of u and
- $u = u_1 p p u_2$, where $u_1, u_2 \in \Pi_k^*$.

We define $\mathsf{snr}_p(u) = u_1 u_2$.

The set of all snr operations is denoted by $\mathsf{Ssn} = \{\mathsf{snr}_p \,|p \in \Pi_k\}$.

spr operation: The *string positive rule (*spr*)* corresponds to hi operation and is defined as follow:

Let p be a pointer of legal string u, $\mathsf{spr}_p(u)$ is applicable if:

- p is a positive pointer of u and
- $u = u_1 v p u_2 \bar{p} u_3$, where $u_1, u_2, u_3 \in \Pi_k^*$.

We define $\mathsf{spr}_p(u) = u_1 \bar{u_2} u_3$.

The set of all spr operations is denoted by $\mathsf{Spr} = \{\mathsf{spr}_p \,|p \in \Pi_k\}$.

sdr operation: The *string double rule (*sdr*)* corresponds to dlad operation and is defined as follow:

Let p, q be two pointers of legal string u, $\mathsf{sdr}_{p,q}(u)$ is applicable if:

- p and q are negative pointers of u and
- $u = u_1 p u_2 q u_3 p u_4 q u_5$, where $u_1, u_2, u_3, u_4, u_5 \in \Pi_k^*$.

We define $\mathsf{sdr}_p(u) = u_1 u_4 u_3 u_2 u_5$.

The set of all sdr operations is denoted by $\mathsf{Sdr} = \{\mathsf{sdr}_{p,q} \,|p, q \in \Pi_k\}$.

We call $\phi = \phi_1 \circ \phi_2 \circ \ldots \circ \phi_n$ a *reduction strategy* if for any $1 \leq i \leq n$, $\phi_i \in \mathsf{Ssn} \cup \mathsf{Spr} \cup \mathsf{Sdr}$. A reduction strategy ϕ is successful for a string u if either $\phi(u) = be$ or $\phi(u) = \bar{e}\bar{b}$.

Example 7. *A reduction strategy for Actin I gene is as follows:* $u' = \mathsf{spr}_2(u) = 34456756789e\bar{3}\bar{b}89$,

$u'' = \mathsf{spr}_3(u') = \bar{e}9\bar{8}\bar{7}6\bar{5}\bar{7}6\bar{5}\bar{4}4589$,

$u''' = \mathsf{sdr}_{7,6}(u'') = \bar{e}9\bar{8}\bar{5}\bar{5}\bar{4}\bar{4}\bar{b}89$,

$u^4 = \mathsf{snr}_5(u''') = \bar{e}9\bar{8}\bar{4}\bar{4}\bar{b}89$,

$u^5 = \mathsf{snr}_4(u^4) = \bar{e}9\bar{8}\bar{b}89$,

$u^6 = \mathsf{spr}_8(u^5) = \bar{e}\bar{9}b9$,

$u^7 = \mathsf{spr}_9(u^6) = \bar{e}\bar{b}$.

Therefore, $\phi = \mathsf{spr}_9\, \mathsf{spr}_8\, \mathsf{snr}_4\, \mathsf{snr}_5\, \mathsf{sdr}_{7,6}\, \mathsf{spr}_3\, \mathsf{spr}_2$ *is a successful reduction strategy for* u.

A reduction strategy for u_1 *from Example 3.2 is as follows:*

$u_1' = \mathsf{sdr}_{2,3}(u_1) = b4\bar{e}\bar{6}4556$,

$u_1'' = \mathsf{spr}_6(u_1') = b4\bar{e}\bar{5}\bar{5}\bar{4}$,

$u_1''' = \mathsf{snr}_5(u_1'') = b4\bar{e}\bar{4}$,

$u_1^4 = \mathsf{spr}_4(u_1''') = \bar{e}\bar{b}$.

Therefore, $\phi_1 = \mathsf{spr}_4\, \mathsf{snr}_5\, \mathsf{spr}_6\, \mathsf{sdr}_{2,3}$ *is a successful reduction strategy for* u.

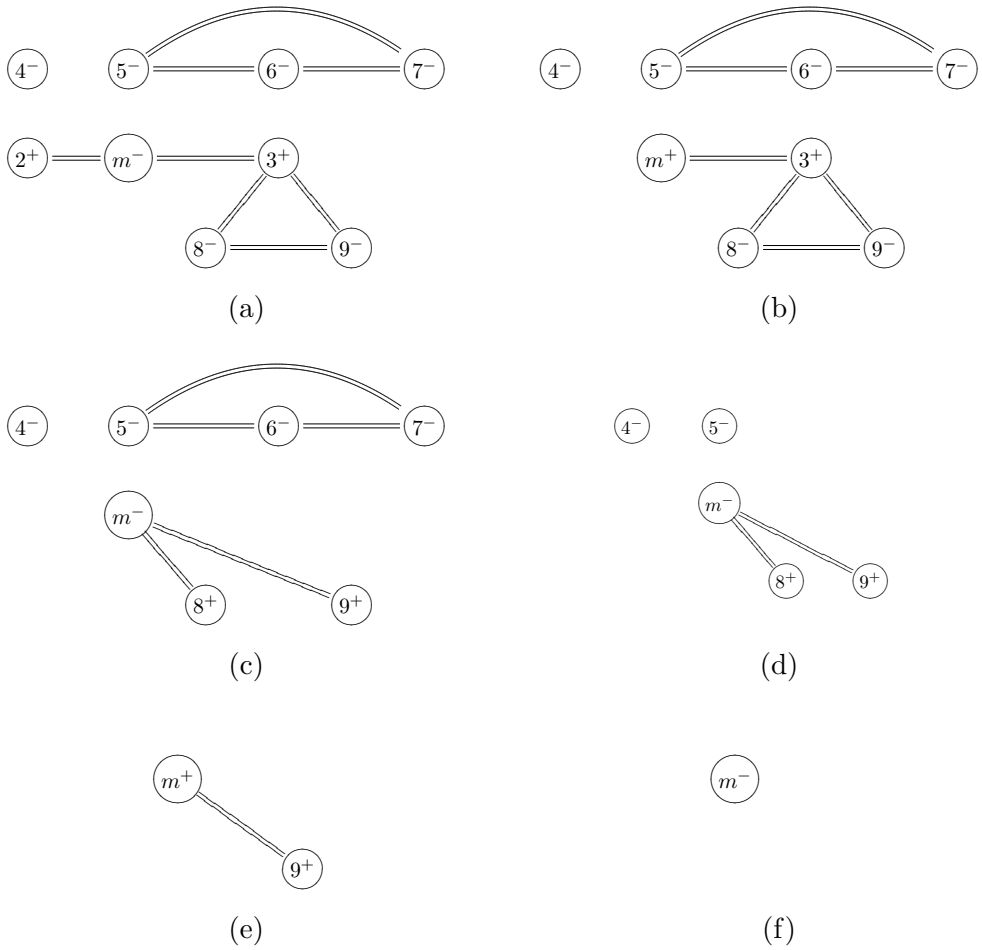


Figure 4.1: (a) The overlap graph G corresponding to Actin I gene in Sterkiella nova; (b) $G' = \mathsf{gpr}_2(G)$; (c) $G'' = \mathsf{gpr}_3(G')$; (d) $G''' = \mathsf{gdr}_{7,6}(G'')$; (e) $G^4 = \mathsf{gpr}_8\,\mathsf{gnr}_4\,\mathsf{gnr}_5(G''')$; (f) $G^5 = \mathsf{sgp}_9(G^4)$.

4.1.3 Gene assembly with overlap graphs

In this section we present the formal definitions of graph reduction operations. This framework provides the modeler with a much more smaller set of cases than the MDS descriptor reduction systems and from the string reduction systems.

gnr operation The *graph negative rule* corresponds to ld operation and defined as follows:

Let $p \in V$ be an isolated vertex of G, gnr_p is applicable if:

- p is negative and

We define $G' = \mathsf{gnr}_p(G)$, where:

- $G' = (V', \sigma', E')$,
- $V' = V \setminus \{p\}$,
- $E' = E$ and
- for every $q \in V$, such that $q \neq p$, $\sigma'(q) = \sigma(q)$.

The set of all gnr operations is denoted by $\mathsf{GNR} = \{\mathsf{gnr}_p \,|p \in \Pi_k\}$.

gpr **operation** The *graph positive rule* corresponds to hi operation and defined as follows: Let $p \in V$ be a vertex of G, gpr_p is applicable if:

- p is positive and

We define $G' = \mathsf{gpr}_p(G)$, where:

- $G' = (V', \sigma', E')$,
- $V' = V \setminus \{p\}$,
- $E' = E \setminus q, p|q \in N_G(p)$ and
- for every $q \in V$, such that $q \notin N_G(p)$, $\sigma'(q) = \sigma(q)$ and $\sigma'(q) = -\sigma(q)$ otherwise.

The set of all gpr operations is denoted by $\mathsf{GPR} = \{\mathsf{gpr}_p \,|p \in \Pi_k\}$.

gdr **operation** The *graph double rule* corresponds to dlad operation and defined as follows:
Let $p, q \in V$ be a vertex of G, gdr_p is applicable if:

- p and q are negative and
- $\{p, q\} \in E$

We define $G' = \mathsf{gdr}_p(G)$, where:

- $G' = (V', \sigma', E')$,
- $V' = V \setminus \{p, q\}$,

- $E' = \{\{s,t\}|s \in N_G(p) \cup N_G(q) \text{ or } t \in N_G(p) \cup N_G(q)\} \cup \{\{s,t\}|\{s,t\} \in E, s \in N_G(p) \setminus N_G(q), \text{ and } y \in N_G(q)\} \cup \{\{s,t\}|\{s,t\} \in E, s \in N_G(p) \cap N_G(q), \text{ and } y \in N_G(p) \bigoplus N_G(p)\} \cup \{\{s,t\}|\{s,t\} \in E, s \in N_G(q) \setminus N_G(p), \text{ and } t \in N_G(p)\}$,

- for every $q \in V$, $\sigma'(q) = \sigma(q)$.

The set of all gdr operations is denoted by $\mathsf{GDR} = \{\mathsf{gdr}_{p,q} | p, q \in \Pi_k\}$.

We call $\phi = \phi_1 \circ \phi_2 \circ \ldots \circ \phi_n$ a *reduction strategy* if for any $1 \leq i \leq n$, $\phi_i \in \mathsf{GNR} \cup \mathsf{GPR} \cup \mathsf{GDR}$. A reduction strategy ϕ is successful for a graph G if $\phi(G)$ is the isolated negative marker node.

Example 8. *A reduction strategy for overlap graph corresponding to Actin I gene, G, is presented in Figure 4.1. As it can be seen from Figure 4.1* $\phi = \mathsf{gpr}_9\,\mathsf{gpr}_8\,\mathsf{gnr}_4\,\mathsf{gnr}_5\,\mathsf{gdr}_{7,6}\,\mathsf{gpr}_3\,\mathsf{gpr}_2$ *is a successful reduction strategy for G. A reduction strategy for the overlap graph corresponding to* u_1 *of Example 3.2,* G_1*, is illustrated in 4.2. As it can be seen,* $\mathsf{gpr}_4\,\mathsf{gnr}_5\,\mathsf{gpr}_6\,\mathsf{gdr}_{2,3}$ *is a successful reduction strategy for* G_1*.*

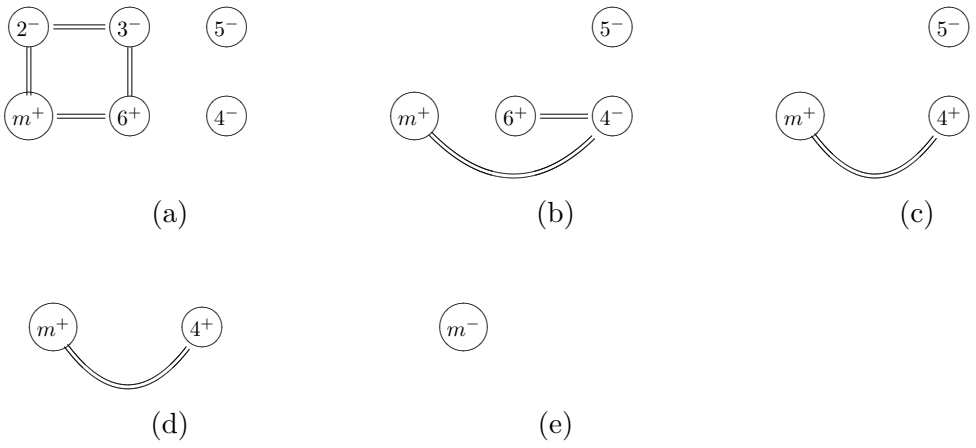


Figure 4.2: (a) The overlap graph G corresponding to string $u_1 = 255634\bar{e}\bar{6}42b3$; (b) $G'_1 = \mathsf{gdr}_{2,3}(G_1)$; (c) $G'' = \mathsf{gpr}_6(G'_1)$; (d) $G'''_1 = \mathsf{gnr}_5(G''_1)$; (e) $G_1^4 = \mathsf{gpr}_4(G'''_1)$.

4.2 Simple gene assembly in ciliates

The gene assembly presented so far was based on arbitrary folds and there were no restrictions on the number of nucleotides present between two pointers. In this section we present a different assembly strategy, called *simple*

intramolecular model, first introduced in [43], which allows only for the simplest possible folds: in the simple intramolecular model there is only a minimal number of other pointers between the pointers involved in the alignment process.

It is perhaps surprising but at the same time very interesting that the simple model turns out to be enough to explain the assembly of all know genes (with the exception of the still incomplete gene pattern of αTBP in Sterkiella Nova, see [59]).

In the simple intramolecular model, there is no other pointer between the aligned ones in the case of simple ld and simple dlad and there is only one in the case of simple hi. The authors in [43] demonstrate that this model can successfully explain the assembly of all currently known ciliate genes in [20]; we refer to [28] and [14] for more details. Different modeling frameworks have been proposed to represent the simple intramolecular operations, e.g., signed permutations [45], string rewriting system [18, 19], graph-based models [17, 10]. Here, we first introduce two graph based frameworks called *directed overlap inclusion (DOI) graphs* introduced in [10] and later investigate the assembly power of them.

4.2.1 Simple gene assembly with legal strings

First we recall the *string pointer reduction system* from Chapter 2 and on what follows we introduce the simple version of assembly operations for legal strings. Note that $p, q \in \Delta_k$ are distinct pointers and $u_1, u_2, u_3 \in \Pi_k^*$.

i. The *simple string negative* rule ssn_p is defined as follows:

$$\mathsf{ssn}_p(u_1\widetilde{p}\widetilde{p}u_2) = u_1u_2,$$

where $\widetilde{p} \in \{p, \bar{p}\}$. We denote $\mathsf{Ssn} = \{\mathsf{ssn}_p \mid p \geq 2\}$.

ii. The *simple string positive* rule ssp_p is defined as follows:

$$\mathsf{ssp}_p(u_1\widetilde{p}\widetilde{q}\bar{\widetilde{p}}u_3) = u_1\bar{\widetilde{q}}u_3,$$

where $\widetilde{p} \in \{p, \bar{p}\}$ and $\widetilde{q} \in \{q, \bar{q}\}$. We denote $\mathsf{Ssp} = \{\mathsf{ssp}_p \mid p \geq 2\}$.

iii. The *simple string double* rule $\mathsf{ssd}_{p,q}$ is defined as follows:

$$\mathsf{ssd}_{p,q}(u_1\widetilde{p}\widetilde{q}u_3\widetilde{p}\widetilde{q}u_5) = u_1u_3u_5,$$

where $\widetilde{p} \in \{p, \bar{p}\}$ and $\widetilde{q} \in \{q, \bar{q}\}$. We denote $\mathsf{Ssd} = \{\mathsf{ssd}_{p,q} \mid p, q \geq 2\}$.

Let $\phi = \phi_r \circ \phi_{r-1} \circ \ldots \circ \phi_1$ be a composition of rules $\phi_i \in \mathsf{Ssn} \cup \mathsf{Ssp} \cup \mathsf{Ssd}$, for all $1 \leq i \leq r$; we call any such composition a *reduction strategy*. We say that ϕ is *successful* for legal string u if $\phi(u) = mm$ or $\phi(u) = \bar{m}\bar{m}$.

Example 9. *Note that as mentioned earlier simple operations are capable to capture the gene assembly of known ciliates gene. Indeed the operations used in Example 7 to reduce the legal string corresponding to Actin I gene in Striekella nova are all simple and therefore Actin I gene can be reduced successfully using only simple operations. On the other hand, for string* u_1 *of Example 7, one only can apply* $\mathsf{ssn}_5(u_1)$ *in the first step, but there is no simple operation applicable on the resulting string that is* $u_1' = 2634\bar{e}\bar{6}42b3$. *Therefore, there is no successful reduction strategy using only simple operations for string* u_1.

4.2.2 Overlap inclusion graphs

Since on one hand, overlap graphs are abstracted from string reduction system in a way that some local information are lost and on the other hand simple operations specifically focus on local relationships between the pointers, the overlap graphs could not be employed to study simple operations. Also graph based frameworks are generally favoured over the string based ones due to the fact that the process of gene assembly in the former one is less involved comparing to the latter one. That was why an attempt to define a suitable graph based substitute was made in [17] to both use both the simplicity of graph based models and to include more information regarding the local interactions between the pointers in the model.

The overlap and the inclusion relations between the pointers of a legal string can be captured through *overlap-inclusion graphs* as defined in [17]. For a legal string u its overlap-inclusion graph G was defined as follows: $V = \mathsf{dom}(u)$ and $E = \{\{p,q\} | p \Rightarrow_u q \text{ or } q \Rightarrow_u p)\} \bigcup \{(p,q) | p \rightarrow_u q\}$. In this way, for any pair of overlapping pointers $\{p,q\}$ in u there is an undirected edge in G between p and q, and for any pointer p whose interval includes in the interval of some pointer q, G has the edge $p \rightarrow_G q$ from p to q. For any $p \in V$, $\sigma(p) = -$ if p corresponds to a negative pointer in string u and $\sigma(p) = +$ otherwise.

Example 10. *The OI graph corresponding to Actin I gene in Sterkiella nova is given in Figure 4.3(a).*

4.2.3 Gene assembly with overlap-inclusion graphs

The *simple graph reduction operations* for overlap-inclusion graphs are defined as follows:

The simple graph negative operations sgn for p, denoted sgn_p, is applicable to G if:

- $\sigma(p) = -$ and

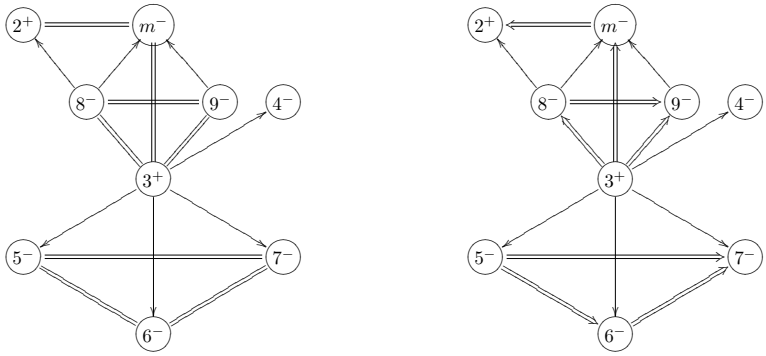


Figure 4.3: (a) The OI graph corresponding to Actin I gene in Sterkiella nova, (b) the DOI graph corresponding to Actin I gene in Sterkiella nova.

- there is no outgoing directed edge from p and there is no undirected edge with p as an endpoint.

In this case, $\mathsf{gnr}_p(G) = G \setminus \{p\}$.

The simple graph positive operation sgpr for p, denoted sgpr_p, is applicable to G if:

- $\sigma(p) = +$,
- there is no outgoing directed edge from p and there is exactly one undirected edge with p as an endpoint.

Let either (p, q) or (q, p) be an undirected edge of G. In this case, $\mathsf{sgpr}_p(G)$ is the graph obtained from $G \setminus \{p\}$ by switching the sign of q.

The simple graph double operation could not be defined with the use of information provided by overlap-inclusion graphs.

Equivalence of this partial model with its corresponding string based framework is proven in [17].

Assembly power of overlap-inclusion graphs Although the framework was not successful in capturing all three simple operations, it could provide interesting results on characterizing the graphs reducible using: (1) only simple graph negative operations, (2) only simple graph positive operations and (3) both simple graph negative and positive operations.

4.2.4 Directed overlap inclusion graphs

To overcome the difficulty of defining *simple graph double operation* we proposed a new type of graph in [10] in the form of an extension to the overlap-inclusion graph representation of micronuclear gene patterns. This extension was performed by introducing a minimal change in the graph, i.e. the undirected edges were substituted by directed ones to represent overlap relation between pointers. This change empowered us to define all three simple operations for gene assembly in ciliates as an abstraction in the level of graphs. This new type of graphs called *directed overlap-inclusion (in short* DOI*)* is defined as follows.

Let u be a legal string. The DOI graph $G_u = (V, E_o, E_i, \sigma)$ corresponding to string u is defined as follows: $V = \mathsf{dom}(u)$ is the set of vertices, $\sigma : V \to \{+, -\}$ is the signing of its vertices such that for each $p \in V$, $\sigma(p) = +$ if p is a positive pointer in u and $\sigma(p) = -$ otherwise. E_o and E_i are sets of its directed edges, $E_o = \{(p, q) | p \Rightarrow_u q\}$ and $E_i = \{(p, q) | p \rightarrow_u q\}$.

Example 11. *Let u be the legal string corresponding to Actin I gene in Sterkiella nova. Its corresponding* DOI *graph is given in Figure 4.3(b).*

4.2.5 Gene assembly with directed overlap-inclusion graphs

We introduced the simple gene assembly operations for DOI-graph-based model of simple gene assembly in ciliates in [9]. The formal definitions are as follows. The following notations are used in what follows and defined as:

- $outSet_o(p)$ denotes the set of outgoing overlap edges from p,
- $inSet_o(p)$ denotes the set of incoming overlap edges to p,
- $outSet_i(p)$ denotes the set of outgoing inclusion edges from p,
- $inSet_i(p)$ denotes the set of incoming inclusion edges to p,
- $outDeg_o(p)$ denotes the number of outgoing overlap edges from p,
- $inDeg_o(p)$ denotes the number of incoming overlap edges to p,
- $outDeg_i(p)$ denotes the number of outgoing inclusion edges from p,
- $inDeg_i(p)$ denotes the number of incoming inclusion edges to p.

Let $G = (V, \sigma, E_o, E_i)$ be a directed, vertex- and edge-labeled graph. For any distinct vertices $p, q \in V \setminus \{m\}$, the graph operations sgn_p, sgp_p and $\mathsf{sgd}_{p,q}$ are defined on G as follows:

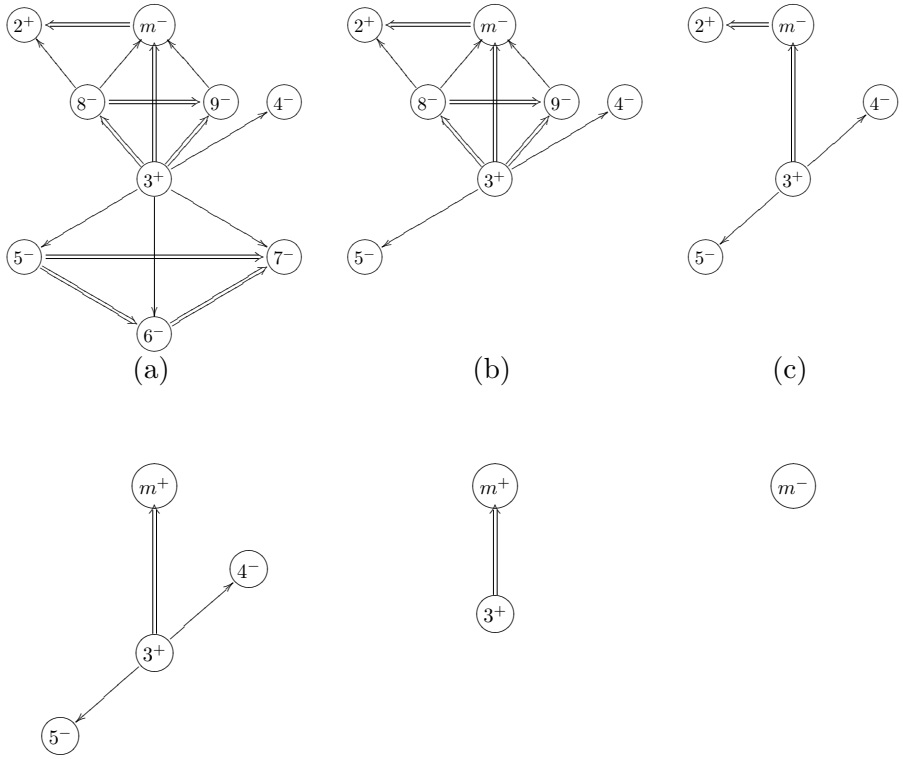


Figure 4.4: (a) The DOI graph G corresponding to Actin I gene in Sterkiella nova; (b) $G' = \mathsf{sgd}_{6,7}(G)$; (c) $G'' = \mathsf{sgd}_{8,9}(G')$; (d) $G''' = \mathsf{sgp}_2(G'')$; (e) $G^4 = \mathsf{sgn}_4\,\mathsf{sgn}_5(G''')$; (f) $G^5 = \mathsf{sgp}_3(G^4)$.

- *The simple graph negative rule* sgn for p, denoted sgn_p, is applicable to G if:
  - $\sigma(p) = -$ and
  - $\mathsf{inDeg}_o(p) = \mathsf{outDeg}_o(p) = \mathsf{outDeg}_i(p) = 0$.

  In this case, $\mathsf{sgn}_p(G) = G \setminus \{p\}$. We denote $\mathsf{Sgn} = \{\mathsf{sgn}_p \mid p \in \Delta_k, p \geq 2\}$. We say that sgn_p *corresponds* to the string-rewriting rule snr_p.

- *The simple graph positive rule* sgp for p, denoted sgp_p, is applicable to G if:
  - $\sigma(p) = +$,
  - $\mathsf{inDeg}_o(p) + \mathsf{outDeg}_o(p) = 1$, and

– $\mathsf{outDeg}_i(p) = 0$.

Let q be the vertex with the property $\mathsf{inSet}_o(p) \cup \mathsf{outSet}_o(p) = \{q\}$. In this case, $\mathsf{sgp}_p(G)$ is the graph obtained from $G \setminus \{p\}$ by switching the label of q: q is negative in $\mathsf{sgp}_p(G)$ if and only if it is positive in G. We denote $\mathsf{Sgp} = \{\mathsf{sgp}_p \mid p \in \Delta_k, p \geq 2\}$. We say that sgp_p *corresponds* to the string-rewriting rule ssp_p.

- *The simple graph double rule* sgd for p, q, denoted $\mathsf{sgd}_{p,q}$, is applicable to G if:

  – $\sigma(p) = \sigma(q) = -$,
  – $q \in \mathsf{outSet}_o(p)$,
  – $\mathsf{inSet}_o(p) \cup \{p\} = \mathsf{inSet}_o(q)$,
  – $\mathsf{outSet}_o(p) = \mathsf{outSet}_o(q) \cup \{q\}$,
  – $\mathsf{inSet}_i(p) = \mathsf{inSet}_i(q)$ and
  – $\mathsf{outSet}_i(p) = \mathsf{outSet}_i(q)$.

  In this case, $\mathsf{sgd}_{p,q}(G) = G \setminus \{p, q\}$. We denote $\mathsf{Sgd} = \{\mathsf{sgd}_{p,q} \mid p, q \in \Delta_k, p, q \geq 2, p \neq q\}$. We say that $\mathsf{sgd}_{p,q}$ corresponds to the string-rewriting rule $\mathsf{ssd}_{p,q}$.

A *reduction strategy* is a composition of simple graph rules $\phi = \phi_{p_n} \circ \ldots \circ \phi_{p_2} \circ \phi_{p_1}$. We say that ϕ is *successful* on G if $\phi(G)$ is the graph having only vertex m where m is negative. Let $\Omega \subseteq \{\mathsf{Sgn}, \mathsf{Sgp}, \mathsf{Sgd}\}$ be a set of types of graph rules. If all rules in ϕ are from the rule sets indicated by Ω, then we say that ϕ is an Ω-*strategy* and that ϕ is Ω-*successful*, resp. We also say that a directed, vertex- and edge-labeled graph is Ω-*reducible* if it has an Ω-successful reduction strategy.

Example 12. *Let G be the DOI graph corresponding to Actin I gene in Sterkiella nova. A successful reduction strategy of G is presented in Figure 4.4.*

Assembly power of directed overlap-inclusion graphs We discussed the characterizing of DOI graphs in [12]. We showed that similar to the results for overlap-inclusion graphs, the characterization of the graphs using: (1) only simple graph negative operations, (2) only simple graph positive operations and (3) both simple graph negative and positive operations to reduce is feasible. Yet the results concerting the other combination of assembly operations remain open.

Chapter 5

Reaction systems

Exploring various functionalities of a living cell has been one of the popular research topics throughout the years, for example see [51, 77, 63, 13]. These functionalities are mainly dictated by the interactions between biochemical reactions, hence having a full understanding of such interactions is of utmost importance. *Reaction systems*, first introduced in [36], is a qualitative framework inspired by two main regulation mechanisms, *facilitation* and *inhibition*, in a cell which control the interaction between biochemical reactions. Intuitively a reaction is enabled when all components needed to facilitate the reaction are present and all components which inhibit such a facilitation are absent from the environment. Based on this intuition a reaction is formalized as a triplet $a = (R_a, I_a, P_a)$ where R_a represents the set of reactants, I_a the set of inhibitors and P_a the set of products corresponding to reaction a. This definition is well supported by how a biochemical reaction functions, i.e. every single reaction transforms its set of reactants to the corresponding set of products if nothing suppresses such a transformation. Let the result of applying a reaction a on a set T be denoted by $\mathsf{res}_a(T)$, if R_a is included in T and I_a is disjoint from R_a, then reaction a is enabled and as a result $\mathsf{res}_a(T) = P_a$ which may or may not be disjoint from either R_a or I_a.

There are two main assumptions considered in reaction systems framework as follows:

- *Threshold assumption:* It is assumed that either an element is present in the environment in abundance or it is absent from it. This implies that there is no counting in RS framework and as a result reaction systems are qualitatively analyzed rather than quantitatively.

- *No permanency assumption:* It is assumed that if no reaction is enabled to preserve an element of the environment vanishes from it.

Diverse studies have been done on various reaction systems related topics, see for example [48, 61, 52, 64, 70, 71, 7, 11, 22]. However two main directions can be identified among them. The first direction is concerned with more theoretical aspects of reaction systems (e.g., [31, 70, 71]). In this direction, the researchers have been more concerned with the theoretical foundations of the reaction systems. The theory-based research varies from studying the properties of the reaction systems, see [15, 30, 35], to extending reaction systems, minimal reaction systems for example [37, 16], to characterizing special reaction systems, for example [29, 72], to studying the complexity of reaction systems [41, 39, 40].

The second direction has taken a more practical path. In this second direction, the reaction systems are either employed as a platform to do bio-modeling, e.g., [7, 11], or have been used to do modeling in other disciplines, like computer science and number theory, see [22, 73].

In this thesis we have chosen to walk along the second path and use the reaction systems as a bio-modeling tool.

5.1 The reaction systems framework

In this section a more formal definition of reaction systems is presented. Let R_a, I_a and P_a be finite non-empty subsets of finite set S such that $R_a \cap I_a = \varnothing$. The tuple $a = (R_a, I_a, P_a)$ is called a *reaction* and its components are as follows:

- R_a is the set of reactants of reaction a;
- I_a is the set of inhibitors of reaction a and
- P_a is the set of products of reaction a.

A reaction a is said to be *enabled* on set $T \subseteq S$ if $R_a \subseteq T$ and $I_a \cap T = \varnothing$. We denote by $res_a(T)$ the result of applying reaction a on set T; $res_a(T) = P_a$ if reaction a is enabled on T and it is empty otherwise. The result of applying a set of reactions A on $T \subseteq S$, denoted by $res_A(T)$, is the collective result of applying each reaction a of A on T independently, i.e. $res_A(T) = \bigcup_{a \in A} res_a(T)$.

A *reaction system* is denoted by $\mathcal{A} = (S, A)$ where S is the set of resources and A is the set of reactions.

The next step after defining the set of reactions and building the system is to get to know the dynamics of such a system. To understand how a system functions it is crucial to know the interaction between its reactions and the outcome of those interactions. In the reaction systems framework *interactive processes* are the tools to provide the researcher with that knowledge.

We denote by $\pi = (\gamma, \delta)$ an *interactive process* of reaction system $\mathcal{A}$ where:

- γ is the *context sequence* of the interactive process π. The context of i^{th} state of π is denoted by C_i;

- δ is the *result sequence* of the interactive process π. The result of i^{th} state of π is denoted by D_i and is defined as $D_i = res_A(C_{i-1} \cup D_{i-1})$.

We say that $W_i = C_i \cup D_i$ is the i^{th} state of the interactive process π; W_0 and W_n are called the initial state and the final state of π respectively.

The interactive processes can be interpreted in different ways, for example, one way of looking at the process is to consider what is produced in the result state as the "knowledge" obtained and the elements of the current state, including given context, as what is "observed" by the researcher. In this way, the knowledge acquired from the interactions happening in the system is the direct outcome of the data which is either produced or provided in the previous state.

5.2 Modeling chemical reaction networks with reaction systems

The simple structure of reaction systems raises the question whether such a straight forward formulation can capture the complex, e.g. ODE-based, models. This concern makes the necessity of having a real life example modeled in reaction systems framework clear. We took in [11], the heat shock response as a case study and translated the biochemical network to a reaction systems based model. The heat shock response is a defense mechanism by which the cell reacts to elevated temperatures. The key players of such a mechanism are heat shock proteins, hsp, which are responsible for facilitating the refolding process of misfolded proteins, mfp, by operating as molecular chaperons. The heat shock proteins form hsp: mfp complexes and help mfp to refold. For more details we refer to [75]. By the proper use of inhibitors, the proposed reaction systems based model proves to be successful in capturing the biological properties of the original ODE-based model, ranging from *mass conservation* to *steady states*.

Part of the model is illustrated in the next example. The full reaction systems based model of heat shock response can be found in [11].

Example 13. *[11] The following reaction systems, $\mathcal{A} = (S, A)$, captures the interactions in the misfolding and refolding steps of the heat shock response process:*

$$
\begin{aligned}
S &= \{\,\mathsf{stress}, \mathsf{prot}, \mathsf{mfp}, \mathsf{hsp{:}\ mfp}, \mathsf{d_I}\}\\
A &= \{(\{\mathsf{prot}, \mathsf{stress}\}, \{\mathsf{d_I}\}, \{\mathsf{prot}, \mathsf{mfp}\}),\\
&\qquad (\{\mathsf{prot}\}, \{\mathsf{d_I}\}, \{\mathsf{prot}\}),\\
&\qquad (\{\mathsf{hsp}, \mathsf{mfp}\}, \{\mathsf{d_I}\}, \{\mathsf{hsp{:}\ mfp}\}),\\
&\qquad (\{\mathsf{mfp}\}, \{\mathsf{hsp}\}, \{\mathsf{mfp}\}),\\
&\qquad (\{\mathsf{hsp{:}\ mfp}\}, \{\mathsf{d_I}\}, \{\mathsf{hsp}, \mathsf{prot}\})\}.
\end{aligned}
$$

The following example presents the steps of an interactive process of Example 13 in which refolding of mfp's in heat shock response is portrayed.

Example 14. *This example is a three step window of a interactive process for the reaction system based model of the heat shock response after misfolding proteins which leads to refolding of* mfp*'s.*

Table 5.1: An interactive process for refolding of mfp's in heat shock response.

State	C_i	D_i	W_i	r_i
0	$\{\mathsf{hsp}, \mathsf{mfp}\}$	$\varnothing$	$\{\mathsf{hsp}, \mathsf{mfp}\}$	$(\{\mathsf{hsp}, \mathsf{mfp}\}, \{\mathsf{d_I}\}, \{\mathsf{hsp{:}\ mfp}\})$
1	$\varnothing$	$\{\mathsf{hsp{:}\ mfp}\}$	$\{\mathsf{hsp{:}\ mfp}\}$	$(\{\mathsf{hsp{:}\ mfp}\}, \{\mathsf{d_I}\}, \{\mathsf{hsp}, \mathsf{prot}\})$
2	$\varnothing$	$\{\mathsf{hsp}, \mathsf{prot}\}$	$\{\mathsf{hsp}, \mathsf{prot}\}$	$(\{\mathsf{prot}\}, \{\mathsf{d_I}\}, \{\mathsf{prot}\})$

This practice shows that, although qualitative in nature, reaction systems are flexible enough to capture the fundamental properties of quantitative frameworks like ODE-based models. To be able to do such a migration from a quantitative world to a qualitative one, the modeler needs to focus on the assumption within the qualitative system which are hidden behind the numerical setup. Inhibitors are another valuable tool that reaction systems provide for the researcher. By proper use of inhibitors one can have control over the causality relations between different reactions in the system. As a result it seems that although reaction systems uses only a small number of properties to define a model, it can be highly flexible in picturing the real life case studies.

5.3 Model checking with reaction systems

Although a very young field of research, reaction systems has attracted a lot of attention in the past few years, and various studies have been conducted

using the reaction systems framework as the basis of their research varying from probabilistic and quantum reaction systems to reaction automata to membrane systems, computer science and biology, see [48], [61], [22], [52] and [64]. The list is quite extensive and this by itself proves the flexibility of the framework and promises strong applicability of it. These studies, however varied, have the element of modeling in common and the very natural question arises in dealing with any new proposed model is to verify and validate the model. This question opens the door to checking the reaction system based models.

One of the first attempts to tackle this question was made in [58] where a temporal logic was introduced to define and check temporal properties of reaction systems. As a result of the study, model checking in this logic was proven to be PSPACE-complete. On the other hand in [7] we considered formalizing various concepts related to biomodeling, i.e. mass conservation, invariants, steady states, stationary processes, elementary fluxes, and periodicity. The definitions closely corresponds to the standard definitions in the quantitative framework, see, e.g. [53]. The computational complexity of deciding these properties for a given reaction system, unlike the correspondent quantitative ones, prove to be difficult. We showed that for a reaction system $\mathcal{A}$ deciding if $M \subseteq S$ is conserved or an invariant set of $\mathcal{A}$ is coNP-complete. We also proved that deciding if $\mathcal{A} = (S, A)$ conserves a non-empty set or has a non-empty invariant set is a coNP-hard problem contained in Σ_2^{P}; deciding if $\mathcal{A}$ has a non-empty steady state or for a steady state W of $\mathcal{A}$, deciding the existence of an elementary flux of size at most k is NP-complete. Also for a steady state W of $\mathcal{A}$, counting the elementary fluxes of size at most k is #P-complete and deciding the periodicity of an interactive process of $\mathcal{A}$ with a periodic context sequence is PSPACE-complete. On the other hand deciding the stationarity of an interactive process of $\mathcal{A}$ with a periodic context sequence proves to be quite easy and is in P.

Although the decision problems stated in [7] are proven to be coNP-complete, we showed that in some cases these problems could be solved in a more efficient way. For example for mass conservation problems we proposed an algorithm which performed better than naive exponential approach. Also sufficient polynomial criterion was presented which answered to the decision problem of whether a given set of species was not conserved very quickly. We also proposed a simulator which automates the process of running reaction systems, and provides a list of conserved sets whenever it is computationally possible. The simulator can be accessed in [76].

Chapter 6

Our original contribution

i. Sepinoud Azimi, Tero Harju, Miika Langille, Ion Petre, Vladimir Rogojin, Directed overlap-inclusion graphs as representations of ciliate genes. Fundamenta Informaticae 110(1–4), 2944, 2011.

   - We introduced directed signed overlap-inclusion graphs to represent ciliate genes.
   - We explored some of their basic properties in connection to the other modeling frameworks for ciliate genes: strings, overlap graphs, and overlap-inclusion graphs.

ii. Sepinoud Azimi, Tero Harju, Miika Langille, Ion Petre, Simple gene assembly as a rewriting of directed overlap-inclusion graphs. Theoretical Computer Science 454, 30-37, 2012.

   - We introduced a graph-based formalization of the simple gene assembly operations. The formalization is based on DOI graphs as a model for the pointer structure of ciliate genes.
   - We proved that our DOI graph-based rules are equivalent with the string-based rewriting system for simple gene assembly and are cable to successfully capture all three operations of the model.

iii. Sepinoud Azimi, Bogdan Iancu, Ion Petre, Reaction system models for the heat shock response, Fundamenta Informaticae, IOS Press, 131, 1–14, 2014.

   - We introduced the notion of dominance graph to represent the affinity between the species of the environment.
   - We further used the dominance graph to develop an approach which allowed us to translate the quantitative behaviour of the ODE models to the qualitative behaviour of the reaction systems.

iv. Sepinoud Azimi, Ion Petre, The reduction power of simple Operations for gene assembly in ciliates. Discrete Mathematics and Computer Science 1, 23-36, 2014.

- We characterized the DOI graphs that are reducible using simple operations of different types.
- We showed that the strategies using simple operations are confluent for DOI graphs.

v. Sepinoud Azimi, Cristian Gratie, Sergiu Ivanov, Luca Manzoni, Ion Petre, Antonio E. Porreca, Complexity of model checking for reaction systems, Submitted, 2015.

- We formulated for reaction system some biomodeling properties, i.e. mass conservation, invariants, steady states, stationary processes, elementary fluxes, and periodicity.
- We focused on the computational complexity of deciding the above mentioned properties for a given reaction system and we established their complexity classes.

vi. Sepinoud Azimi, Cristian Gratie, Sergiu Ivanov, Ion Petre, Dependency graphs and mass conservation in reaction systems, Theoretical Computer Science, Accepted, 2015.

- We discussed how mass conservation is related with the inner structure of the reaction system and based on that we proposed the notion of conservation dependency graph.
- We described the algorithm for listing the conserved sets based on the conservation dependency graph.
- We conjectured that for reaction system models inspired by the real-life examples, our proposed algorithm only produces a polynomial number of non-conserved candidate sets (in terms of the number of species).
- We provided a negative polynomial heuristics for both mass conservation problem and a generalized conservation problem.
- We gave a short presentation of our reaction systems simulator.

Chapter 7

Discussion

The borderlines of scientific fields have been fading away more and more over time. Disciplines cease to exist in an isolation and interdisciplinary research gains more attention day by day. The reason lies in the fact that with the advancement of science and technology no single perspective is capable to capture the true nature of any phenomena. Natural computing, standing on the border between biology and computer science, is an excellent example of an interdisciplinary research area.

The contribution of research done in natural computing is twofold: on one hand, it sheds light into how nature works and how it processes the information and on the other hand, it provides some guidelines on how to design bio-inspired technologies.

In this thesis on one hand we explored a natural computing area in which computational tools are used to explain a biological phenomena, i.e. gene assembly in ciliates. Gene assembly in ciliates has received a lot of attention many years in various directions and exploring the assembly process has led to new results in e.g., providing insight into evolution mechanism at the gene level, see [54].

On the other hand, we investigated a field in which computation is inspired by a biological construct, i.e. reaction systems. Reaction systems, although a very young research topic, has attracted a lot of attention in the past few years. They can be viewed as a tool which adds a qualitative perspective to the most quantitative problems. To be able to interpret models in the reaction systems framework, one needs to decrypt the true role of numbers in the quantitative models. In this process a much deeper understanding of the phenomenon under study will be provided.

The studies on reaction systems by no means is limited to either computer science or biology. It covers a wide spectrum of science ranging from mathematics to quantum computing to logic, see for example [48, 61, 52, 71].

The first model that we took into consideration in this thesis was the *intramolecular model* for gene assembly in ciliates proposed in [67]. We considered a variation of the intramolecular model called *simple gene assembly model*, first introduced in [43], which only allows the minimal possible folds in the assembly process. Simple intramolecular model was proved to be effective in assembling all known genes, see [43]. In [10], we proposed a new representation for ciliate genes, called *directed overlap-inclusion (DOI) graphs* and we further discussed various properties of such graphs. In [9] we defined simple gene assembly operations within the DOI graph framework. Finally, in [12] we discussed the reduction power of simple operations for gene assembly in ciliates and characterized the genes which could be arranged by using: (1) only simple graph negative operations, (2) only simple graph positive operations and (3) both simple graph negative and positive operations.

The computational power of restricted subsets of simple operations which include simple graph double rules is still open and would be part of the future work.

In this thesis we also used *reaction systems*, first introduced in [36], as a modeling tool to build biological models. In [11], we developed an approach where we could capture the affinity between species by choosing proper inhibitors rather than the ODE-models kinetic rates. We also introduced the notion of *dominance graph* which facilitated the translation of the quantitative behaviour of the ODE model to the qualitative behaviour of the reaction system.

In [7], we defined for specific biomodeling properties for reaction systems, i.e. mass conservation, invariants, steady states, stationary processes, elementary fluxes, and periodicity and focused on the computational complexity of deciding these properties.

In [8], we studied the relation between mass conservation and the inner structure of the reaction system, and proposed the notion of *conservation dependency graph* and based on this new notion, we presented an algorithm to list the conserved sets. We also conjectured that for reaction systems models of real-life biochemical networks, our algorithm only produces a polynomial number of non-conserved candidate sets (in terms of the number of species). Finally, we presented our reaction systems simulator.

As a part of future work it would be interesting to expand our study on building biological models in reaction systems framework and generalize our approach into a well defined algorithm which enables us to build an RS-based model for any given biochemical network.

Bibliography

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

[2] Artiom Alhazov, Chang Li, and Ion Petre. Computing the graph-based parallel complexity of gene assembly. *Theoretical Computer Science*, 411(25):2359–2367, 2010.

[3] Artiom Alhazov, Ion Petre, and Vladimir Rogojin. Solutions to computational problems through gene assembly. *Natural Computing*, 7(3):385–401, 2008.

[4] Artiom Alhazov, Ion Petre, and Vladimir Rogojin. The parallel complexity of signed graphs: Decidability results and an improved algorithm. *Theoretical Computer Science*, 410(24):2308–2315, 2009.

[5] Angela Angeleska, Nataša Jonoska, and Masahico Saito. Rewriting rule chains modeling DNA rearrangement pathways. *Theoretical Computer Science*, 454:5–22, 2012.

[6] Angela Angeleska, Nataša Jonoska, Masahico Saito, and Laura F. Landweber. Rna-guided DNA assembly. *Journal of Theoretical Biology*, 248(4):706–720, 2007.

[7] Sepinoud Azimi, Cristian Gratie, Sergiu Ivanov, Luca Manzoni, Ion Petre, and Antonio E. Porreca. Complexity of model checking for reaction systems. Technical Report 1122, Turku Centre for Computer Science, 2014.

[8] Sepinoud Azimi, Cristian Gratie, Sergiu Ivanov, and Ion Petre. Dependency graphs and mass conservation in reaction systems. *Theoretical Computer Science*, 2015.

[9] Sepinoud Azimi, Tero Harju, Miika Langille, and Ion Petre. Simple gene assembly as a rewriting of directed overlap-inclusion graphs. *Theoretical Computer Science*, 454:30–37, 2012.

[10] Sepinoud Azimi, Tero Harju, Miika Langille, Ion Petre, and Vladimir Rogojin. Directed overlap-inclusion graphs as representations of ciliate genes. *Fundamenta Informaticae*, 110(1):29–44, 2011.

[11] Sepinoud Azimi, Bogdan Iancu, and Ion Petre. Reaction system models for the heat shock response. *Fundamenta Informaticae*, 131(3):299–312, 2014.

[12] Sepinoud Azimi and Ion Petre. The reduction power of simple operations for gene assembly in ciliates. *Discrete Mathematics and Computer Science*, 1:2336, 2014.

[13] James M. Bower and Hamid Bolouri. *Computational Modeling of Genetic and Biochemical Networks*. MIT press, 2004.

[14] Robert Brijder, Mark Daley, Tero Harju, Nataša Jonoska, Ion Petre, and Grzegorz Rozenberg. Computational nature of gene assembly in ciliates. *Handbook of Natural Computing*, pages 1233–1280, 2012.

[15] Robert Brijder, Andrzej Ehrenfeucht, and Grzegorz Rozenberg. A note on causalities in reaction systems. *Electronic Communications of the EASST*, 30, 2010.

[16] Robert Brijder, Andrzej Ehrenfeucht, and Grzegorz Rozenberg. Reaction systems with duration. In *Computation, cooperation, and life*, pages 191–202. Springer, 2011.

[17] Robert Brijder and Hendrik Jan Hoogeboom. Combining overlap and containment for gene assembly in ciliates. *Theoretical Computer Science*, 411(6):897–905, 2010.

[18] Robert Brijder, Miika Langille, and Ion Petre. A string-based model for simple gene assembly. In Erzsbet Csuhaj-Varju and Zoltan Esik, editors, *Fundamentals of Computation Theory*, volume 4639 of *Lecture Notes in Computer Science*, pages 161–172. Springer, 2007.

[19] Robert Brijder, Miika Langille, and Ion Petre. Extended strings and graphs for simple gene assembly. *Theoretical Computer Science*, 411(4-5):730–738, 2010.

[20] Andre R. Cavalcanti, Thomas H. Clarke, and Laura F. Landweber. Mds_ies_db: a database of macronuclear and micronuclear genes in spirotrichous ciliates. *Nucleic acids research*, 33(suppl 1):D396–D398, 2005.

[21] Bastien Chopard and Michel Droz. *Cellular Automata*. Springer, 1998.

[22] Luca Corolli, Carlo Maj, Fabrizio Marini, Daniela Besozzi, and Giancarlo Mauri. An excursion in reaction systems: From computer science to biology. *Theoretical Computer Science*, 454:95–108, 2012.

[23] Mark J. Daley and Lila Kari. DNA computing: Models and implementations. *Comments on Theoretical Biology*, 7(3):177–198, 2002.

[24] Jürgen Dassow and Markus Holzer. Language families defined by a ciliate bio-operation: Hierarchies and decision problems. *International Journal of Foundations of Computer Science*, 16(04):645–662, 2005.

[25] Jürgen Dassow, Victor Mitrana, and Arto Salomaa. Operations and language generating devices suggested by the genome evolution. *Theoretical Computer Science*, 270(1):701–738, 2002.

[26] Hubert De Fraysseix and Patrice O. De Mendez. A short proof of a Gauss problem. In *Graph Drawing*, pages 230–235. Springer, 1997.

[27] Andrzej Ehrenfeucht, Tero Harju, Ion Petre, David M. Prescott, and Grzegorz Rozenberg. Formal systems for gene assembly in ciliates. *Theoretical Computer Science*, 292(1):199–219, 2003.

[28] Andrzej Ehrenfeucht, Tero Harju, Ion Petre, David M. Prescott, and Grzegorz Rozenberg. *Computation in Living Cells: Gene Assembly in Ciliates*. Springer, 2004.

[29] Andrzej Ehrenfeucht, Jetty Kleijn, Maciej Koutny, and Grzegorz Rozenberg. Minimal reaction systems. In *Transactions on Computational Systems Biology XIV*, pages 102–122. Springer, 2012.

[30] Andrzej Ehrenfeucht, Michael Main, and Grzegorz Rozenberg. Combinatorics of life and death for reaction systems. *International Journal of Foundations of Computer Science*, 21(03):345–356, 2010.

[31] Andrzej Ehrenfeucht, Michael Main, and Grzegorz Rozenberg. Functions defined by reaction systems. *International Journal of Foundations of Computer Science*, 22(01):167–178, 2011.

[32] Andrzej Ehrenfeucht, Ion Petre, David M. Prescott, and Grzegorz Rozenberg. Circularity and other invariants of gene assembly in ciliates. In *Words, semigroups, and transductions*, pages 81–97, 2001.

[33] Andrzej Ehrenfeucht, Ion Petre, David M. Prescott, and Grzegorz Rozenberg. String and graph reduction systems for gene assembly in ciliates. *Mathematical Structures in Computer Science*, 12(02):113–134, 2002.

[34] Andrzej Ehrenfeucht, David M. Prescott, and Grzegorz Rozenberg. Computational aspects of gene (un) scrambling in ciliates. In *Evolution as Computation*, pages 216–256. Springer, 2002.

[35] Andrzej Ehrenfeucht and Grzegorz Rozenberg. Events and modules in reaction systems. *Theoretical Computer Science*, 376(1):3–16, 2007.

[36] Andrzej Ehrenfeucht and Grzegorz Rozenberg. Reaction systems. *Fundamenta Informaticae*, 75(1):263–280, 2007.

[37] Andrzej Ehrenfeucht and Grzegorz Rozenberg. Introducing time in reaction systems. *Theoretical Computer Science*, 410(4):310–322, 2009.

[38] Andrzej Ehrenfeucht and Grzegorz Rozenberg. Processes inspired by the functioning of living cells: Natural computing approach. In Giancarlo Mauri, Alberto Dennunzio, Luca Manzoni, and Antonio E. Porreca, editors, *Unconventional Computation and Natural Computation*, volume 7956 of *Lecture Notes in Computer Science*, pages 3–5. Springer Berlin Heidelberg, 2013.

[39] Enrico Formenti, Luca Manzoni, and Antonio E. Porreca. Cycles and global attractors of reaction systems. In *Descriptional Complexity of Formal Systems*, pages 114–125. Springer, 2014.

[40] Enrico Formenti, Luca Manzoni, and Antonio E. Porreca. Fixed points and attractors of reaction systems. In *Language, Life, Limits*, pages 194–203. Springer, 2014.

[41] Enrico Formenti, Luca Manzoni, and Antonio E. Porreca. On the complexity of occurrence and convergence problems in reaction systems. *Natural Computing*, pages 1–7, 2014.

[42] Ashish Ghosh and Shigeyoshi Tsutsui. *Advances in Evolutionary Computing: Theory and Applications*. Springer Science & Business Media, 2003.

[43] Tero Harju, Ion Petre, Vladimir Rogojin, and Grzegorz Rozenberg. Patterns of simple gene assembly in ciliates. *Discrete Applied Mathematics*, 156(14):2581–2597, 2008.

[44] Tero Harju, Ion Petre, and Grzegorz Rozenberg. Two models for gene assembly in ciliates. In Juhani Karhumki, Hermann Maurer, Gheorghe Pun, and Grzegorz Rozenberg, editors, *Theory Is Forever*, volume 3113 of *Lecture Notes in Computer Science*, pages 89–101. Springer Berlin Heidelberg, 2004.

[45] Tero Harju and Grzegorz Rozenberg. Computational processes in living cells: gene assembly in ciliates. *Lecture Notes in Computer Science*, 2450:1–20, 2003.

[46] Simon S. Haykin. *Neural Networks and Learning Machines*, volume 3. Pearson Education Upper Saddle River, 2009.

[47] Mika Hirvensalo. *Quantum Computing*. Springer Berlin, 2001.

[48] Mika Hirvensalo. On probabilistic and quantum reaction systems. *Theoretical Computer Science*, 429:134–143, 2012.

[49] Tseren-Onolt Ishdorj, Ion Petre, and Vladimir Rogojin. Computational power of intramolecular gene assembly. *International Journal of Foundations of Computer Science*, 18(05):1123–1136, 2007.

[50] Lila Kari and Grzegorz Rozenberg. The many facets of natural computing. *Communications of the ACM*, 51(10):72–83, 2008.

[51] Hiroaki Kitano et al. *Foundations of Systems Biology*. MIT press Cambridge, MA;, 2001.

[52] Jetty Kleijn and Maciej Koutny. Membrane systems with qualitative evolution rules. *Fundamenta Informaticae*, 110(1):217–230, 2011.

[53] Edda Klipp, Ralf Herwig, Axel Kowald, Christoph Wierling, and Hans Lehrach. *Systems Biology in Practice: Concepts, Implementation and Application*. John Wiley & Sons, 2008.

[54] Laura F Landweber. Why genomes in pieces? *Science*, 318(5849):405, 2007.

[55] Laura F. Landweber and Lila Kari. The evolution of cellular computing: Natures solution to a computational problem. *Biosystems*, 52(1):3–13, 1999.

[56] Laura F. Landweber and Lila Kari. Universal molecular computation in ciliates. In *Evolution as Computation*, pages 257–274. Springer, 2002.

[57] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

[58] Artur Meski, Wojciech Penczek, and Grzegorz Rozenberg. Model checking temporal properties of reaction systems. *Information Sciences*, 2015.

[59] Jennifer L Mitcham, Aenoch J Lynn, and David M Prescott. Analysis of a scrambled gene: the gene encoding alpha-telomere-binding protein in oxytricha nova. *Genes & development*, 6(5):788–800, 1992.

[60] Mariusz Nowacki, Vikram Vijayan, Yi Zhou, Klaas Schotanus, Thomas G. Doak, and Laura F. Landweber. Rna-mediated epigenetic programming of a genome-rearrangement pathway. *Nature*, 451(7175):153–158, 2008.

[61] Fumiya Okubo, Satoshi Kobayashi, and Takashi Yokomori. Reaction automata. *Theoretical Computer Science*, 429:247–257, 2012.

[62] Gheorghe Păun. *Membrane Computing: An Introduction*. Springer Science & Business Media, 2002.

[63] Gheorghe Păun. Bio-inspired computing paradigms (natural computing). *Lecture Notes in Computer Science*, 3566:155–160, 2005.

[64] Gheorghe Păun, Mario J Pérez-Jiménez, and Grzegorz Rozenberg. Bridging membrane and reaction systems–further results and research topics. *Fundamenta Informaticae*, 127(1):99–114, 2013.

[65] David M. Prescott. The DNA of ciliated protozoa. *Microbiological reviews*, 58(2):233, 1994.

[66] David M. Prescott. Genome gymnastics: unique modes of DNA evolution and processing in ciliates. *Nature Reviews Genetics*, 1(3):191–198, 2000.

[67] David M. Prescott, Andrzej Ehrenfeucht, and Grzegorz Rozenberg. Molecular operations for DNA processing in hypotrichous ciliates. *European Journal of Protistology*, 37(3):241–260, 2001.

[68] David M. Prescott, Andrzej Ehrenfeucht, and Grzegorz Rozenberg. Template-guided recombination for ies elimination and unscrambling of genes in stichotrichous ciliates. *Journal of Theoretical Biology*, 222(3):323–330, 2003.

[69] David M. Prescott and Grzegorz Rozenberg. How ciliates manipulate their own DNA–a splendid example of natural computing. *Natural Computing*, 1(2-3):165–183, 2002.

[70] Arto Salomaa. Functions and sequences generated by reaction systems. *Theoretical Computer Science*, 466:87–96, 2012.

[71] Arto Salomaa. Functional constructions between reaction systems and propositional logic. *International Journal of Foundations of Computer Science*, 24(1):147–159, 2013.

[72] Arto Salomaa. Minimal and almost minimal reaction systems. *Natural Computing*, 12(3):369–376, 2013.

[73] Arto Salomaa. Applications of the chinese remainder theorem to reaction systems with duration. *Theoretical Computer Science*, 2015.

[74] Alan M Turing. Intelligent machinery, a heretical theory. *The Turing Test: Verbal Behavior as the Hallmark of Intelligence*, page 105, 1948.

[75] Richard Voellmy and Frank Boellmann. Chaperone regulation of the heat shock protein response. In Peter Csermely and László Vígh, editors, *Molecular Aspects of the Stress Response: Chaperones, Membranes and Networks*, volume 594 of *Advances in Experimental Medicine and Biology*, pages 89–99. Springer New York, 2007.

[76] The web interface of the reaction system simulator. `http://combio.abo.fi/research/reaction-systems/reaction-system-simulator/`.

[77] Hector Zenil. A behavioural foundation for natural computing and a programmability test. In *Computing Nature*, pages 87–113. Springer, 2013.

Paper I

Sepinoud Azimi, Tero Harju, Miika Langille, Ion Petre, Vladimir Rogojin, Directed overlap-inclusion graphs as representations of ciliate genes. Fundamenta Informaticae 110(1-4), 29-44, 2011.

Fundamenta Informaticae 110 (2011) 29–44
DOI 10.3233/FI-2011-526
IOS Press

Directed Overlap-inclusion Graphs as Representations of Ciliate Genes

Sepinoud Azimi*
Computational Biomodeling Laboratory
Turku Centre for Computer Science
Åbo Akademi University, Turku, Finland
sepinoud.azimi@abo.fi

Miika Langille
Computational Biomodeling Laboratory
Turku Centre for Computer Science
Åbo Akademi University, Turku, Finland
miika.langille@gmail.com

Vladimir Rogojin
Computational Systems Biology Laboratory
University of Helsinki, Helsinki, Finland
vladimir.rogojin@helsinki.fi

Tero Harju
Department of Mathematics
University of Turku, Turku, Finland
tero.harju@utu.fi

Ion Petre
Computational Biomodeling Laboratory
Turku Centre for Computer Science
Åbo Akademi University, Turku, Finland
ion.petre@abo.fi

Abstract. The simple intramolecular model for gene assembly in ciliates consists of three molecular operations based on local DNA manipulations. It was shown to predict correctly the assembly of all currently known ciliate gene patterns. Mathematical models in terms of signed permutations and signed strings proved limited in capturing some of the combinatorial details of the simple gene assembly process. A different formalization in terms of overlap-inclusion graphs, recently introduced by Brijder and Hoogeboom, proved well-suited to describe two of the three operations of the model and their combinatorial properties. We introduce in this paper an extension of the framework of Brijder and Hoogeboom in terms of directed overlap-inclusion graphs where more of the linear structure of the ciliate genes is described. We investigate a number of combinatorial properties of these graphs, including a necessary property in terms of forbidden induced subgraphs.

Keywords: Directed overlap-inclusion graphs, gene assembly in Ciliates, simple operations

*Address for correspondence: Computational Biomodeling Laboratory, Turku Centre for Computer Science, Åbo Akademi University, Turku 20520, Finland

1. Introduction

Ciliates form a large and old group of unicellular eukaryotes. One of their characteristics is that each ciliate contains two types of functionally different nuclei: the germline nuclei (micronuclei) and the somatic nuclei (the macronuclei), each having multiple copies. The genes are differently organized in the two types of nuclei: micronuclear genes are split into blocks (called MDSs), which are separated by noncoding blocks. The MDSs come in a shuffled order, some of them being also inverted. Each MDS M ends with a short sequence of nucleotides (called *pointer*) that has a second occurrence in the beginning of the MDS that should follow M in the orthodox order. Macronuclear genes have all the MDSs spliced together (or *assembled*) on their common pointers. During sexual reproduction, all macronuclei are destroyed and new macronuclei are formed starting from a copy of a micronucleus. During this process, micronuclear genes get transformed into macronuclear genes by having excised all noncoding blocks and assembling the MDSs in the orthodox order. The process is called *gene assembly* and has been subject to intense combinatorial and computational research in the last decade. We refer for details to [6], [1], and [23] and references therein.

Several molecular models were considered for the gene assembly process, see [1]. Among them is the simple intramolecular model introduced in [8]. Unlike the other models, the simple intramolecular model postulates that gene assembly takes place as a result of local interactions, where only neighboring MDSs are able to interact with each other. The model was shown in [15] to predict correctly the assembly of all currently known gene patterns, see the database discussed in [5] for an up-to-date list. The simple model was modeled mathematically as a sorting of signed permutations in [16], and as a string rewriting system in [3, 4]. Both formal frameworks turned out to be limited in capturing the details of the local interactions postulated by the simple model and made it difficult to characterize, e.g., all gene patterns that can be assembled through simple operations. A similar difficulty in the case of the general intramolecular model was overcome by extending the model to signed overlap graphs, see [6]. In the case of simple operations, signed overlap graphs seem however insufficient to capture unambiguously the information about the distance among various pointers and MDSs, a crucial ingredient in the very definition of the simple model. A partial solution was introduced in [2] where genes were modeled as signed overlap-inclusion graphs. However, only two of the three operations of the simple model could be modeled in this context.

In this paper we extend the graph framework of [2] and introduce *directed* signed overlap-inclusion graphs as a model for ciliate genes. We explore some of their basic properties in connection to the other modeling frameworks for ciliate genes: strings, overlap graphs, and overlap-inclusion graphs. We also prove a number of combinatorial results about the directed signed overlap-inclusion graphs such as a necessary property for these graphs in terms of forbidden induced subgraphs. Even though the difference with respect to the framework of [2] may seem relatively minor, in modeling the overlap relationship among pointer intervals as directed rather than undirected edges, the properties of the directed overlap-inclusion graphs are remarkably different. In particular, they are able to support defining all three operations of the simple intramolecular model, which was not possible in the framework of [2]. Due to lack of space, we only briefly discuss the modeling of the simple model operations in this new framework and rather focus in this paper on its combinatorial properties.

2. Preliminaries

We recall in this section some of the basic definitions we need throughout the paper. For more details we refer to [6].

2.1. Legal strings

For an alphabet Σ and two strings u, v over Σ, we say that v is a *scattered subsequence* of u if $u = a_1a_2\dots a_n$ and $v = a_{i_1}a_{i_2}\dots a_{i_k}$, for some $0 \leq k \leq n$, $1 \leq i_1 < \dots < i_k \leq n$, and $a_j \in \Sigma$, for all $1 \leq j \leq n$.

Let $\Delta_k = \{2, 3, \dots, k\}$ be an alphabet of *pointers*, $M = \{b, e\}$ a set of markers and $\Sigma_k = \Delta_k \cup M$, for some $k \geq 1$. Without risk of confusion, we will often omit the subscript k and simply write Σ instead of Σ_k. We denote by $\overline{\Sigma}_k = \{\overline{2}, \dots, \overline{k}, \overline{b}, \overline{e}\}$ a signed copy of Σ_k and let $\Sigma_k^{\maltese} = (\Sigma_k \cup \overline{\Sigma}_k)^*$.

We say that a string u in $\Sigma_k^{\maltese}$ is *legal* if for any $a \in \Delta_k$, u contains either 0, or 2 occurrences from the set $\{a, \overline{a}\}$ and moreover, u contains exactly one occurrence from the set $\{b, \overline{b}\}$ and one occurrence from the set $\{e, \overline{e}\}$. If u contains occurrences from the set $\{a, \overline{a}\}$, for some $a \in \Sigma_k$, then we say that a occurs in u and denote it $a \in u$. We define the *domain* of u as $\mathrm{dom}(u) = \{a \in \Sigma_k \mid a \in u\}$.

Let $p \in \Sigma$ and let $u \in \Sigma^{\maltese}$ be a legal string. If u contains both substrings p and $\overline{p}$ then p is said to be *positive* in u; otherwise, it is said to be *negative*. If $u = u_1p'u_2p''u_3$, with $p', p'' \in \{p, \overline{p}\}$, then the *$p$-interval* of u is the substring u_2.

For any distinct $p, q \in u$, p and q have one of the following relations:

- p and q *overlap* if exactly one occurrence from $\{p, \overline{p}\}$ can be found in the q-interval of u. We denote the overlapping relation by $p \Rightarrow_u q$, if the first occurrence from $\{p, \overline{p}\}$ occurs in u before the first occurrence from $\{q, \overline{q}\}$ and we denote it by $q \Rightarrow_u p$ otherwise;
- q is *included* in p if the two occurrences from $\{q, \overline{q}\}$ are found within the p-interval. This relation is denoted by $p \rightarrow_u q$;
- p and q are *disjoint* if they do not overlap and neither is included in the other in u.

For details on how to associate a legal string to a ciliate gene we refer to [6]. For example, the legal string corresponding to actin I gene in Sterkiella nova is $34456756789e\bar{3}\ \bar{2}b289$, see [6].

2.2. Overlap graphs

The overlap relationships of the pointers of a legal string can be presented through an *overlap graph* (also known as *interlacement graphs*, see, e.g., [11]). The overlap-graph based pointer reduction system was introduced in [10, 7] to model gene assembly in ciliates through rewriting of overlap graphs. For a legal string u, its overlap graph $G = (V, \sigma, E)$ was defined as follows: $V = \mathrm{dom}(u)$, $\sigma : V \to \{+, -\}$ is the signing of vertices from V (i.e., if $p \in V$ is positive in the corresponding string u, then $\sigma(p) = +$, otherwise $\sigma(p) = -$) and $E = \{\{p, q\} | p \Rightarrow_u q \text{ or } q \Rightarrow_u p\}$.

Example 2.1. The overlap graph corresponding to actin I gene in Sterkiella nova is shown in Figure 1(a).

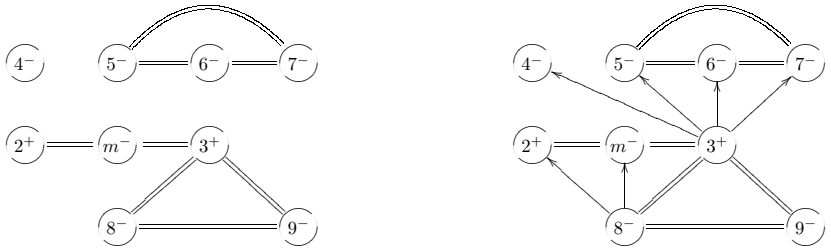


Figure 1. (a) The overlap graph corresponding to actin I gene in Sterkiella nova, (b) The overlap-inclusion graph corresponding to actin I gene in Sterkiella nova.

2.3. Overlap-inclusion graphs

The overlap and the inclusion relations between the pointers of a legal string can be captured through *overlap-inclusion graphs* as defined in [2]. For a legal string u its overlap-inclusion graph G was defined as follows: $V = \text{dom}(u)$ and $E = \{\{p, q\}|p \Rightarrow_u q \text{ or } q \Rightarrow_u p)\} \bigcup \{(p, q)|p \rightarrow_u q\}$. In this way, for any pair of overlapping pointers $\{p, q\}$ in u there is an undirected edge in G between p and q, and for any pointer p whose interval includes in the interval of some pointer q, G has the edge $p \rightarrow_G q$ from p to q. Note that in [2], the authors used the reverse orientation for the inclusion edges.

Example 2.2. The overlap-inclusion graph corresponding to actin I gene in Sterkiella nova is shown in Figure 1(b).

3. Directed overlap-inclusion graphs

We introduce in this section a new type of graph to represent the overlap and the inclusion relations among the pointers of a legal string. We extend the overlap-inclusion graph representation of micronuclear gene patterns introduced in [2]. The change we introduce is minimal: we substitute undirected edges which represent overlap relation between pointers with directed edges. This change is however enough to be able to define all three simple operations for gene assembly in ciliates on the level of graphs, a problem left (partially) open in [2]. Due to lack of space, we only focus in this paper on the properties of the directed overlap-inclusion graphs and only briefly discuss the graph-based modeling of the simple gene assembly operations.

3.1. Definitions and basic results

We define the directed overlap-inclusion graphs as follows.

Definition 3.1. Let u be a legal string. The directed overlap-inclusion (in short *DOI*) graph $G_u = (V, E_o, E_i, \sigma)$ corresponding to u is defined as follows: $V = \text{dom}(u)$ is the set of vertices, $\sigma : V \rightarrow \{+, -\}$ is the signing of its vertices such that for each $p \in V$, $\sigma(p) = +$ if p is a positive pointer in u and $\sigma(p) = -$ otherwise. E_o and E_i are sets of its directed edges, $E_o = \{(p, q)|p \Rightarrow_u q\}$ and $E_i = \{(p, q)|p \rightarrow_u q\}$.

For a *DOI* graph G and any string u such that $G = G_u$ we say that u corresponds to G.

Example 3.1. The *DOI* graph corresponding to actin I gene in Sterkiella nova is shown in Figure 2.

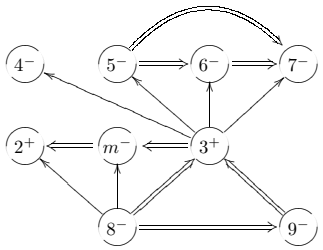


Figure 2. The directed overlap-inclusion graph corresponding to actin I gene in Sterkiella nova.

Example 3.2. Note that more than one string may correspond to a *DOI* graph, for example

$$u = 6224335546 \text{ and } v = 6224553346$$

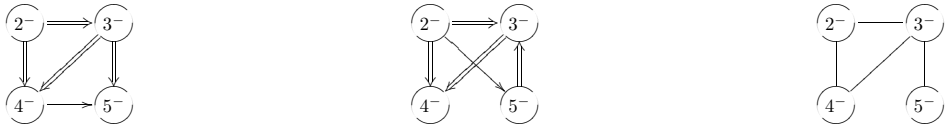
have the same *DOI* graph.

Definition 3.2. Let G be a directed labeled graph with $\{+, -\}$ as vertex labels and {'overlap', 'inclusion'} as edge labels. The *underlying digraph* of G is the graph obtained by removing edge labels, and the *underlying graph* of G is the graph obtained by removing edge labels and orientations.

We prove now several basic results about *DOI* graphs. The following result gives the connection between the directed overlap-inclusion graph of a string and its overlap graph. The result is straightforward to prove based on Definition 3.1.

Lemma 3.1. Let G_u be the *DOI* graph corresponding to string u and G'_u the graph constructed from G_u by removing its inclusion edges, and replacing its directed overlap edges with undirected ones. Then G'_u is the overlap graph corresponding to u.

Example 3.3. There are distinct *DOI* graphs having the same underlying overlap graph, see Figure 3.



(G_1) (G_2) (G)

Figure 3. The *DOI* graphs G_1 and G_2 (corresponding to $u_1 = 23425354$ and $u_2 = 25354234$, resp.) have the same underlying overlap graph G.

Lemma 3.2. Every induced subgraph of a *DOI* graph is a *DOI* graph.

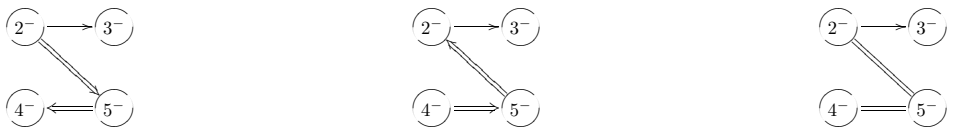
Proof:
Let $G = (V, E_o, E_i, \sigma)$ be the *DOI* graph corresponding to u. Let G' be its induced subgraph on vertices $V' = \{v_1, v_2, ..., v_k\} \subseteq V$. Let u' be the string obtained from u by removing all occurrences of every $v \in V \setminus V'$. We claim that the *DOI* graph corresponding to u' is G'. Let p, q be two overlapping pointers in u' and $p \Rightarrow_{u'} q$. It follows that $p \Rightarrow_u q$ and so, $p \Rightarrow_G q$. Thus, since $p, q \in V'$, we have $p \Rightarrow_{G'} q$. Take now an overlap edge of G', $r \Rightarrow_{G'} s$. It follows that $r \Rightarrow_G s$ and, $r \Rightarrow_u s$. Since $r, s \in V'$, we obtain that $r \Rightarrow_{u'} s$. □

Lemma 3.3. Let $G_u = (V, E_o, E_i, \sigma)$ be the *DOI* graph corresponding to legal string u. Let E'_o be the set of undirected edges over V defined as follows:

$$E'_o = \{\{p, q\} \mid p, q \in V \text{ and either } (p, q) \in E_o, \text{ or } (q, p) \in E_o\}.$$

Then the graph $H = (V, E'_o, E_i, \sigma)$ is the overlap-inclusion graph corresponding to u.

Example 3.4. There are distinct *DOI* graphs having the same underlying overlap-inclusion graph, see Figure 4.



(G_1) (G_2) (G)

Figure 4. The two *DOI* graphs G_1 and G_2 (corresponding to $u_1 = 23352454$ and $u_2 = 45425332$, resp.) have the same underlying overlap-inclusion graph G.

Theorem 3.1. Any *DOI* graph G is a directed acyclic graph.

Proof:
Since the direction of an edge is always determined by the order in which the elements occur in the double occurrence string u, the *DOI* graph, G_u, corresponding to u is acyclic, i.e., there are are no directed cycles in G_u. □

Corollary 3.1. Any connected component of a *DOI* graph G, is *rooted*, i.e., the underlying digraph is acyclic and there exists exactly one vertex, called the *root* of G, of indegree zero.

It turns out that the directed overlap relation between pointers establishes their order in any corresponding string.

Lemma 3.4. Let G be a *DOI* graph that contains the following path $s_1 \Rightarrow_G s_2 \Rightarrow_G \cdots \Rightarrow_G s_n$, $s_i \neq s_j$ for $i \neq j$. Then the first occurrences of the pointers in string u corresponding to G appear in the order $s_1 s_2 \cdots s_n$. The same holds also for the sequence of their second occurrences.

Proof:
According to the definition of the overlap relation, if $s_i \Rightarrow_G s_{i+1}$, then in all strings u corresponding to G, $s_i s_{i+1} s_i s_{i+1}$ is a scattered subsequence of u, for all $1 \leq i \leq n-1$. □

Lemma 3.5. Let G be a *DOI* graph. If $s_1 \Rightarrow_G s_n$, $s_1 \Rightarrow_G s_2 \Rightarrow_G ... \Rightarrow_G s_n$, then any legal string u corresponding to G has the following (scattered) subsequence:

$$s_1 s_2 \cdots s_n s_1 s_2 \cdots s_n.$$

Proof:
Let G be a *DOI* graph with edges $s_1 \Rightarrow_G s_n$, $s_1 \Rightarrow_G s_2 \Rightarrow_G ... \Rightarrow_G s_n$. Since $s_1 \Rightarrow_G s_n$, pointers s_1 and s_n occur in order $s_1 s_n s_1 s_n$ in any string corresponding to G. Since we have $s_1 \Rightarrow_G s_2 \Rightarrow_G ... \Rightarrow_G s_n$, by Lemma 3.4 pointers $s_1, s_2, \ldots, s_n$ occur in order $s_1 s_2 s_n s_1 s_2 s_n$. □

Lemma 3.5 has the following additional implication.

Corollary 3.2. Let G be a *DOI* graph. If $s_1 \Rightarrow_G s_n$, $s_1 \Rightarrow_G s_2 \Rightarrow_G ... \Rightarrow_G s_n$, then $s_i \Rightarrow_G s_j$, for all $2 \leq i < j \leq n$.

Proof:
By Lemma 3.5 there is a scattered sequence of pointers

$$s_1 s_2 \cdots s_n s_1 s_2 \cdots s_n$$

in any string corresponding to G. In this way, for any i and j, $1 \leq i < j \leq n$ pointers s_i and s_j occur in order $s_i s_j s_i s_j$ in any string corresponding to G. Then $s_i \Rightarrow_G s_j$. □

The following three results correspond Lemma 3.4, Lemma 3.5 and Corollary 3.2 for inclusion edges.

Lemma 3.6. Let G be a *DOI* graph that contains the following path $s_1 \rightarrow_G s_2 \rightarrow_G \cdots \rightarrow_G s_n$, $s_i \neq s_j$ for $i \neq j$. Then the first occurrences of the pointers in string u corresponding to G appear in the order $s_1 s_2 \cdots s_n$. the second occurrences of the pointers in string u corresponding to G appear in the order $s_n s_{n-1} \cdots s_1$.

Lemma 3.7. Let G be a *DOI* graph. If $s_1 \rightarrow_G s_n$, $s_1 \rightarrow_G s_2 \rightarrow_G ... \rightarrow_G s_n$, then any legal string u corresponding to G has the following (scattered) subsequence:

$$s_1 s_2 \cdots s_n s_n s_{n-1} \cdots s_1.$$

Corollary 3.3. Let G be a *DOI* graph. $s_1 \rightarrow_G s_n$, $s_1 \rightarrow_G s_2 \rightarrow_G ... \rightarrow_G s_n$, then $s_i \rightarrow_G s_j$, is an inclusion edge for all $2 \leq i < j \leq n$.

3.2. Forbidden subgraphs

In this section we introduce the concept of forbidden subgraphs of directed overlap-inclusion graphs.

Definition 3.3. Let G be a directed, vertex- and edge-labeled graph. We say that G is forbidden if there is no string u such that G is the *DOI* graph corresponding to u.

Definition 3.4. Let u be a legal string. If $u = a_1 a_2 \ldots a_n$, then $u^R = a_n \ldots a_2 a_1$ is the reversal of string u. If G is the *DOI* graph corresponding to legal string u, then G^R is the graph corresponding to u^R.

The following result is straightforward.

Lemma 3.8. A minimal (in number of vertices) forbidden directed, vertex- and edge-labeled graph is connected, i.e., its underlying graphs is connected.

Lemma 3.9. For any *DOI* graph G, G^R is also a *DOI* graph.

Theorem 3.2. Let G be a directed labeled graph with $\{+, -\}$ as vertex labels and {'overlap','inclusion'} as edge labels. If G is a 3-vertex, acyclic graph, then G is forbidden if and only if it is isomorphic to one of the graphs in Figure 5.

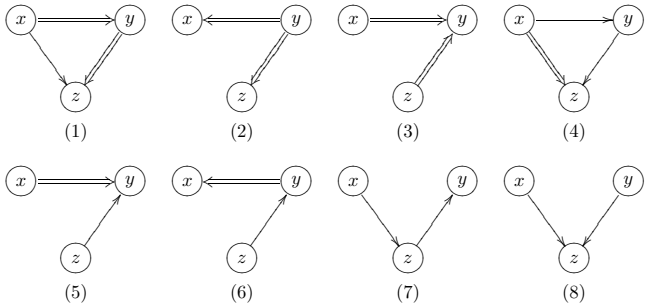


Figure 5. All 3-vertex, acyclic, forbidden graphs. Inclusion edges are illustrated as simple arrows and overlap edges as double arrows.

Proof:
Let $\{x, y, z\}$ be the vertices of G. Depending on the type of edges that G consists of, we consider separately the following ten cases.

i. G **consists of three overlap edges.** All acyclic graphs with three vertices and three overlap edges are isomorphic to the graph with $x \Rightarrow_G y$, $z \Rightarrow_G y$ and $z \Rightarrow_G x$. The string $zxyzxy$ corresponds to G, therefore, G is not forbidden.

ii. G **consists of two overlap edges and one inclusion edge.** It is straightforward to see that an acyclic graph with two overlap edges and one inclusion edge is isomorphic to one of the graphs in Figure 6. Graphs H_1 and H_2 are not forbidden: strings $yxzyzx$ and $xzyzxy$ correspond to them, respectively. In the case of H_3, let u be a string corresponding to it. Then $xyxy$ and $yzyz$ are scattered subsequences of u. Thus, $u = xyzxyz$ or $u = xyxzyz$. In neither of these strings does the x-interval include the z-interval.

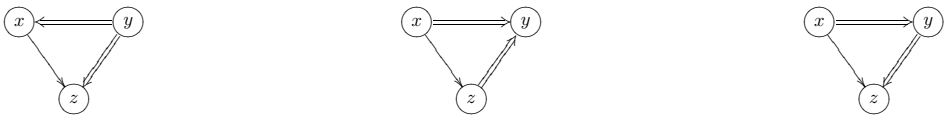
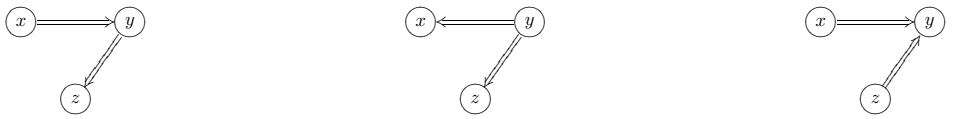


Figure 6. All graphs with two overlap edges and one inclusion edge of the type considered in Theorem 3.2.

iii. G **consists of two overlap edges.** The graph can only be isomorphic to one of the graphs in Figure 7. String $xyxzyz$ corresponds to H_4, so it is not forbidden.

Any string u corresponding to H_5 has $yxyx$ and $yzyz$ as scattered subsequences. Thus, there should be either an overlap, or inclusion relation between x and z. This is a contradiction.

Any string u corresponding to H_6 has $xyxy$ and $zyzy$ as scattered subsequences. Thus, there should be either an overlap, or inclusion relation between x and z. This is a contradiction.

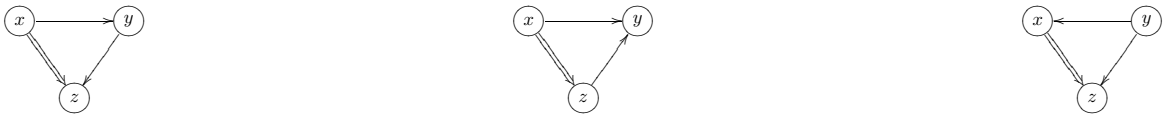


(H_4) (H_5) (H_6)

Figure 7. All graphs with two overlap edges of the type considered in Theorem 3.2.

iv. G **consists of one overlap edge and two inclusion edges.** The graph can only be isomorphic to one of the graphs in Figure 8. String corresponding to H_7 has $xyyx$ and $yzzy$ as scattered subsequences. Therefore, $u = xyzzyx$, contradicting $x \Rightarrow_G z$. Thus, H_7 is forbidden.

Strings $xzyyxz$ and $yxzxzy$ correspond to H_8 and H_9, respectively.



(H_7) (H_8) (H_9)

Figure 8. All graphs with one overlap edge and two inclusion edges of the type considered in Theorem 3.2.

v. G **consists of one overlap edge and one inclusion edge.** The graph can only be isomorphic to one of the graphs in Figure 9. The corresponding strings to H_{11} and H_{12} are $xyxzzy$ and $yzzxyx$ respectively, so they are not forbidden.

In the case of H_{10} and H_{13} we have $zyyz$ as a scattered subsequence of any corresponding string. Also x should have an occurrence in the y-interval. Consequently there must be an edge(of some kind and orientation) between x and z, a contradiction. Thus, H_{10} and H_{13} are forbidden.

(H_{10}) (H_{11}) (H_{12}) (H_{13})

Figure 9. All graphs with one overlap edge and one inclusion edge of the type considered in Theorem 3.2.

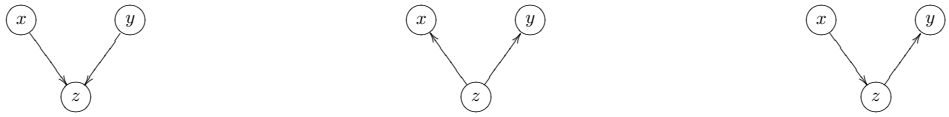
vi. G **consists of one overlap edge.** The graph is isomorphic to the *DOI* graph corresponding to $xyxyzz$ and thus, it is a *DOI* graph.

vii. G **consists of three inclusion edges.** All acyclic graphs with three vertices and three inclusion edges are isomorphic to the graph with $x \rightarrow_G y$, $y \rightarrow_G z$ and $x \rightarrow_G z$. The string $xyzzyx$ corresponds to G, therefore, G is not forbidden.

viii. G **consists of two inclusion edges.** The graph can only be isomorphic to one of the graphs in Figure 10. String $zxxyyz$ corresponds to H_{15}, so it is not forbidden.

For graph H_{14} we have scattered subsequences $xzzx$ and $yzzy$. On the other hand, the x-interval and the y-interval of u are disjoint. This is a contradiction so, H_{14} is forbidden.

For graph H_{16} we have scattered subsequences $xzzx$ and $zyyz$. On the other hand, the x-interval and the y-interval of u are disjoint. This is a contradiction so, H_{16} is forbidden.



(H_{14}) (H_{15}) (H_{16})

Figure 10. All graphs with two inclusion edges of the type considered in Theorem 3.2.

ix. G **consists of one inclusion edge.** The graph is isomorphic to the *DOI* graph corresponding to $xyyxzz$ and thus, it is a *DOI* graph.

x. G **consists of no edges.** The graph is isomorphic to the *DOI* graph corresponding to $xxyyzz$ and thus, it is a *DOI* graph. □

Corollary 3.4. A graph G with an induced 3-vertex subgraph isomorphic to one the graphs in Figure 5 is forbidden.

Proof:
This follows easily from Lemma 3.2 and Theorem 3.2. □

Example 3.5. The opposite direction of Corollary 3.4 is not generally true as it can be seen in Figure 11. By Theorem 3.2, none of the induced subgraphs of G is forbidden. On the other hand, G is forbidden. To prove it, assume that there is a string u corresponding to G. Then we have $xwzxwz$ as a scattered substring of u. Since $w \rightarrow_G y$, both occurrences of y come in the $x - interval$ of u. It follows now that there should exist edges of some kind between $\{y, z\}$ and $\{y, x\}$, a contradiction. We denote the graph in Figure 11 by DF^4.

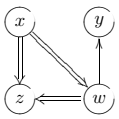


Figure 11. DF^4, an example of a forbidden 4-vertex *DOI* graph that has no forbidden 3-vertex induced subgraph.

Definition 3.5. An undirected graph G is called an *interval graph* if its vertices can be put into one-to-one correspondence with a set of intervals I of a linearly ordered set (like the real line) such that two vertices are connected by an edge of G if and only if their corresponding intervals have nonempty intersection [12].

It is straightforward to conclude the following lemma from the Definition 3.5.

Lemma 3.10. Interval graphs are exactly the underlying graphs of *DOI* graphs. In other words, the *DOI* graphs are *edge-colored orientations* of interval graphs.

The following result is a characterization of [22] of interval graphs in term of forbidden graphs.

Lemma 3.11. Let G be a *DOI* graph. If the underlying graph of G has an induced subgraph isomorphic to either C_{n+4} (a directed cycle with $n \geq 0$) or one of the graphs in Figure 12, then G is forbidden.

(T_2) (X_{31}) (XF_2^{n+1}) (XF_3^n)

Figure 12. Four forbidden interval graphs.

Let G be *DOI* graph and p, q two vertices of G. If $p \Rightarrow_G q$ or $p \rightarrow_G q$, then we write $p \rightsquigarrow_G q$.

Lemma 3.12. A forbidden graph G of four or more vertices is rooted (with a unique root) or it contains a directed cycle C_n for some $n \geq 3$.

Proof:
Suppose that G is acyclic. Then the underlying digraph is acyclic, and it contains one or more vertices with indegree 0. Suppose there are two such vertices p and q. By Theorem 3.2, p and q do not have a common neighbour. Let t be a vertex such that there is a directed path from p to t and from q to t such that the sum of the lengths is minimal. Then there are vertices t_p and t_q with $t_p \rightsquigarrow_G t$ and $t_q \rightsquigarrow_G t$ such that t_p is on the path from p and t_q is on the path from q. By the forbidden triplets we must have also $t_p \rightsquigarrow_G t_q$ or $t_q \rightsquigarrow_G t_p$. However, this contradicts the minimality assumption, since now it should be $t = t_q$ or $t = t_p$. □

Lemma 3.13. Let G be a forbidden graph, and let p be its root. Then the digraph $G - p$ is connected, and there is a vertex q such that $p \Rightarrow_G q$.

Proof:
Let p be the unique root of G provided by Lemma 3.12, and let $A_1, \cdots, A_k$ be the connected components of the underlying graph of the *DOI* graph $G - p$. Suppose $k \geq 2$. By Lemma 3.15, the degree of p is at least two (and it has no incoming edges). Hence the subgraphs G_i induced by $A_i \cup \{p\}$ are *DOI* graphs, and thus $G_i = G(w_i)$ for some double occurrence string w_i.

If for all neighbors $q \in A_i$ of p, we have $p \rightarrow q$, then clearly we must have $w_i = pv_ip$, since the digraph induced by A_i is connected, and in this case $p \rightarrow q$ for all A_i. Moreover, if this holds for all i, then $w = pv_1v_2 \cdots v_kp$ is a double occurrence string such that $G = G(w)$; a contradiction on the choice

of G. Hence there must be one component, say A_1, such that $p \Rightarrow_G q$ for some $q \in A_1$. Now $G_1 = G(w_1)$, where $w_1 = pu_1qu_2pu_3qu_4$ for some strings u_1, u_2, u_3, u_4. By the forbidden triplets, the index 1 is the only one with this property. Hence $p \to t$ for all $t \in \cup_{i=2}^{k} A_i$. Now $w = pv_2 \cdots v_k u_1 q u_2 p u_3 q u_4$ satisfies $G = G(w)$; again a contradiction. □

Lemma 3.14. Let DF_m be a graph of order $m + 4$ for $m \geq 1$ consisting of vertices p, q, s, t and $t_1, t_2, \ldots, t_m$ such that $t \Rightarrow q \Rightarrow s$, $t \Rightarrow t_1 \Rightarrow \ldots \Rightarrow t_m \Rightarrow s$, $q \to p$, $q \to t_i$ for each $i = 1, 2, \ldots, m$. DF_m is forbidden.

Proof:
Assume that there is a string u corresponding to DF_m. We have $tqt_1tt_2t_1t_3t_2t_3 \cdots t_{m-1}t_mt_{m-1}st_mqs$ as a scattered string of u. Since $q \to p$, both occurrences of p come within the q-interval, therefore there should be an edge of some kind between p and t, which is a contradiction. Thus, DF_m is forbidden. □

Lemma 3.15. Let G be a minimal forbidden graph with a vertex of degree one. Then G has one of the graphs from Figure 5, DF^4 or DF_m for some $m \geq 1$ as an induced subgraph.

Proof:
The graph G is connected, suppose $\deg(p) = 1$ and let q be the unique neighbor of p. Consider the graph $G - p$ where the vertex p is removed. By assumption of minimality, $G - p$ is a *DOI* graph, and hence there exists a double occurrence string $w = -q - q-$ such that $G - p$ is the *DOI* graph corresponding to w . (1) If $p \to_G q$ is the only outgoing edge of p, then, let t be a neighbor of q different from p. Now, $\{p, q, t\}$ forms a forbidden triple as can be seen from Theorem 3.2. (2) Let $p \leftarrow_G q$. Now, add pp after the first occurrence of q to obtain $w^{(0)} = -qpp - q-$. Since G is not a *DOI* graph, $G \neq G_{(w^{(0)})}$, and hence there must exist a vertex t in G such that pp belongs to the t-interval in $w^{(0)}$. This can happen only if $t \to_G q$ or $t \Rightarrow_G q$. The first option gives a forbidden nontransitive triple $t \to_G q \to_G p$ in G. Hence $t \Rightarrow_G q$.

Choose t such that its second occurrence is the last one with $t \Rightarrow_G q$. (It is in the q-interval.)

Then add pp in the original w after the second occurrence of t to obtain $w^{(1)} = -t - q - tpp - q-$. Again, since G is forbidden and therefore not corresponding to $w^{(1)}$, there exists a vertex t_1 in G such that pp belongs to the t_1-interval in $w^{(1)}$. If the second occurrence of t_1 is not in the q-interval, then we necessarily have the forbidden subgraph DF^4. Hence the second t_1 occurs between pp and the second q, and thus $q \to t_1$ and $t \Rightarrow t_1$ hold. Choose t_1 to be the last element with this property.

Consider the original w and replace the last t_1 by t_1pp. Once again, there exists a t_2 such that pp occurs in the t_2-interval. Now the t-interval and the t_2-interval must be disjoint by the choice of t and t_1. Hence the elements t_2 have two choices:

$$-t - q - t_1 - t - t_2 - t_1pp - t_2 - q-\,,$$
$$-t - q - t_1 - t - t_2 - t_1pp - q - t_2 -\ .$$

The second one creates a forbidden DF_5 in G. In the first case, let t_2 be the last one with $t_1 \Rightarrow t_2$ and $q \to t_2$.

We proceed inductively. Let m be the first index for which $t_{m-1} \Rightarrow t_m$ and $q \to t_m$, and there exists an element s the second occurrence of s does not belong to the q-interval, and $t_m \Rightarrow s$ holds. (This s is

obtained by considering the word $w^{(m)}$ obtained from w replacing the last occurrence of t_m by $t_m pp$.) Now $q \Rightarrow s$, since $s \rightarrow q$ would result to the forbidden induced subgraph $s \rightarrow q \rightarrow p$. The word w is now of the form

$$w = -t - q - t_1 - t - t_2 - t_1 - \cdots - s - t_m - q - s- \,,$$

which is the forbidden DF_m.

(3) Assume $p \Rightarrow_G q$. Then the forbidden triples yield that the indegree of q is one. Replace the first occurrence of q in w by pqp. Since G is not a *DOI* graph, there is a vertex t such that $t \Rightarrow_G q$ or $t \rightarrow_G q$. However, this not possible by the indegree of q.

(4) Assume $p \Leftarrow_G q$. This case is a dual case of (3), i.e., it reduces to (3) by considering the reverse string w^R of w. □

4. Discussion

In this paper we proposed a new type of graphs, directed overlap-inclusion graph, as a model for the pointer structure of ciliate genes. The main goal of introducing the model was to be able to investigate the combinatorial properties of the simple intramolecular model for gene assembly (such as characterizing the gene patterns that can be assemble through applications of simple operations), which was not possible in terms of permutations or strings, and only partially possible in terms of overlap-inclusion graphs.

In particular, all three operations of the simple model can easily be defined in terms of signed directed overlap-inclusion graphs as follows.

Let G be a *DOI* graph and p an arbitrary node of G. We denote by $inSet_i(p)$ the set of vertices with an inclusion edge ending in p. Moreover, $inDeg_i(p)$ is the number of vertices in $inSet_i(p)$. Similarly, we use $outSet_i(p)$ and $outDeg_i(p)$ to denote the set and number of vertices with an inclusion edge starting from p. We use the notation $inSet_o(p)$, $inDeg_o(p)$, $outSet_o(p)$, and $outDeg_o(p)$, resp. to denote the corresponding notions for overlap edges adjacent to p.

For a *DOI* graph $G = (V, E_o, E_i, \sigma)$ and vertices $p, q \in V$ we define the following operations:

1. *The simple graph negative rule* sgn_p:
   - sgn_p can be applied to G if $\sigma(p) = -$ and $inDeg_o(p) + outDeg_o(p) + outDeg_i(p) = 0$;
   - If $G' = \mathsf{sgn}_p(G)$, then $V' = V \setminus \{p\}$, $\sigma'(r) = \sigma(r)$ for all $r \in V'$, $E'_o = E_o$ and $E'_i = E_i \setminus \{(q,p) | q \in inSet_i(p)\}$;
2. *The simple graph positive rule* sgp_p:
   - sgp_p can be applied to G if $\sigma(p) = +$, $inDeg_o(p) + outDeg_o(p) = 1$, and $outDeg_i(p) = 0$;
   - If $G' = \mathsf{sgp}_p(G)$, then $V' = V \setminus \{p\}$, $\sigma'(r) = \sigma(r)$ for all $r \in V' \setminus \{q\}$, $\sigma'(q) = -\sigma(q)$, $E'_o = E_o \setminus \{(p,q),(q,p)\}$ and $E'_i = E_i \setminus \{(r,p) | r \in inDeg_i(p)\}$, where $inSet_o(p) \cup outSet_o(p) = \{q\}$;
3. *The simple graph double rule* $\mathsf{sgd}_{p,q}$:
   - $\mathsf{sgd}_{p,q}$ can be applied to G if $\sigma(p) = \sigma(q) = -$, $q \in outSet_o(p)$ and $inSet_o(p) \cup p = inSet_o(q)$, $outSet_o(p) = outSet_o(q) \cup q$, $inSet_i(p) = inSet_i(q)$ and $outSet_i(p) = outSet_i(q)$;

- If $G' = \mathsf{sgd}_{p,q}(G)$, then $V' = V \setminus \{p, q\}$, $\sigma'(r) = \sigma(r)$ for all $r \in V'$, $E'_o = E_o \setminus \{(p, q)\} \cup \{(p, s), (s, p), (t, q), (q, t) | s \in inSet_o(p) \cup outSet_o(p), t \in inSet_o(q) \cup outSet_o(q)\}$, $E'_i = E_i \setminus \{(p, s), (s, p), (t, q), (q, t) | s \in inSet_i(p) \cup outSet_i(p), t \in inSet_i(q) \cup outSet_i(q)\}$,

Example 4.1. Consider the *DOI* graph G corresponding to string $u = b234566e\bar{3}\,\bar{2}45$. Its *DOI* graph-based simple assembly is illustrated in Figure 13.

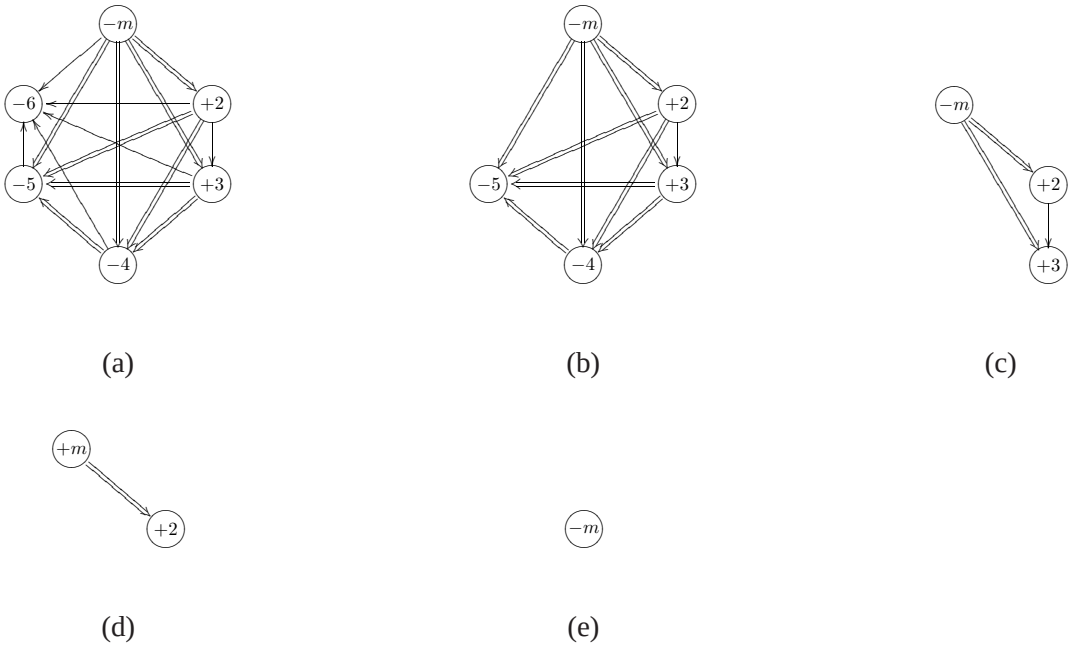


Figure 13. (a) G is the *DOI* graph corresponding to $u = b234566e\bar{3}\,\bar{2}45$, (b) $G' = \mathsf{sgn}_6(G)$ corresponds to $u' = b2345e\bar{3}\,\bar{2}45$, (c) $G'' = \mathsf{sgd}_{4,5}(G')$ corresponds to $u'' = b23e\bar{3}\,\bar{2}$, (d) $G''' = \mathsf{sgp}_3(G'')$ corresponds to $u''' = b2\overline{e}\overline{2}$, (e) $G^{iv} = \mathsf{sgp}_2(G''')$ corresponds to $u^{iv} = be$.

Due to lack of space we do not investigate in this paper the computational and combinatorial properties of the *DOI*-based model for simple gene assembly.

We proved in this paper that distinct signed double occurrence strings may have the same corresponding *DOI* graph. Characterizing all such strings corresponding to a given *DOI* graph remains however an open problem.

References

[1] Brijder, R., Harju,T., Jonoska, N., Petre, I., Rozenberg, G.: Gene assembly in ciliates. In: G. Rozenberg, T.H.W. Bck, J.N. Kok (Eds.): *Handbook of Natural Computing*, Springer, to appear, 2011.

[2] Brijder, R., Hoogeboom, H.J.: Combining overlap and containment for gene assembly in ciliates. *Theoretical Computer Science*, **411**(6), pp. 897–905. *doi:10.1016/j.tcs.2009.07.047*

[3] Brijder, R., Langille, M., Petre,I. : A string-based model for simple gene assembly. In: E. Csuhaj-Varju and Z. Ésik (Eds.): Proceedings of FCT 2007, Springer, Lecture Notes in Computer Science 4639, 161-172, 2007.

[4] Brijder, R., Langille, M., Petre, I.: Extended strings and graphs for simple gene assembly. *Theoret Comp Sci*, **411**, 730-738, 2010.

[5] Cavalcanti, A., Clarke, T.H., Landweber, L.: MDS_IES_DB: a database of macronuclear and micronuclear genes in spirotrichous ciliates. *Nucleic Acids Research* **33** (2005) D396–D398.

[6] Ehrenfeucht, A., Harju, T., Petre, I., Prescott, D. M., Rozenberg, G.: *Computation in Living Cells: Gene Assembly in Ciliates*, Springer (2003).

[7] Ehrenfeucht, A., Petre, I., Prescott, D. M., Rozenberg, G.: String and graph reduction systems for gene assembly in ciliates. *Math. Structures Comput. Sci.*, **12**, (2001), pp. 113–134.

[8] Ehrenfeucht, A., Petre, I., Prescott, D. M., Rozenberg, G.: Universal and simple operations for gene assembly in ciliates. In: V. Mitrana and C. Martin-Vide (eds.) *Words, Sequences, Languages: Where Computer Science, Biology and Linguistics Meet*, Kluwer Academic, Dortrecht, (2001) pp. 329–342.

[9] Ehrenfeucht, A., Prescott, D. M., Rozenberg, G.: Computational aspects of gene (un)scrambling in ciliates. In: L. F. Landweber, E. Winfree (eds.) *Evolution as Computation*, Springer, Berlin, Heidelberg, New York (2001) pp. 216–256.

[10] Ehrenfeucht, A., Harju, T., Petre, I., Prescott, D. M., Rozenberg, G.: Formal systems for gene assembly in ciliates. *Theoret. Comput. Sci.* **292** (2003) 199–219.

[11] De Fraysseix, H., Ossona de Menzdez, P.: A short proof of a Gauss problem. *Lecture Notes in Computer Science*, **1353**, (1997), pp. 230–235.

[12] Golumbic, M. C.: Algorithmic graph theory and perfect graphs, *Academic Press*,(1980), ISBN 0-12-289260-7.

[13] Harju, T., Li, C., Petre, I., Rozenberg, G.: Complexity Measures for Gene Assembly In: K. Tuyls (Eds.), Proceedings of the Knowledge Discovery and Emergent Complexity in Bioinformatics workshop, Springer, Lecture Notes in Bioinformatics 4366, 42-60, 2007.

[14] Harju, T., Petre, I., Rozenberg, G.: Formal properties of gene assembly: Equivalence problem for overlap graphs. *Lecture Notes in Comput. Sci*, **2950** (2004) 202–212.

[15] Harju, T., Li, C., Petre, I., Rozenberg, G.: Modelling simple operations for gene assembly. In: Junghuei Chen, Natasha Jonoska, Grzegorz Rozenberg (Eds), Nanotechnology: Science and Computation, 361-376, Springer, 2006.

[16] Harju, T., Petre, I., Rogojin, V., Rozenberg, G.: Patterns of simple gene assembly in Ciliates, *Discrete Applied Mathematics*, **156**(14), Elsevier, (2008), pp. 2581–2597.

[17] Harju, T., Rozenberg, G.: Computational processes in living cells: gene assembly in ciliates. *Lecure Notes in Comput. Sci.* **2450** (2003) 1–20.

[18] Landweber, L. F., Kari, L.: The evolution of cellular computing: Nature's solution to a computational problem. In: *Proceedings of the 4th DIMACS Meeting on DNA-Based Computers*, Philadelphia, PA (1998) pp. 3–15.

[19] Landweber, L. F., Kari, L.: Universal molecular computation in ciliates. In: L. F. Landweber and E. Winfree (eds.) *Evolution as Computation*, Springer, Berlin Heidelberg New York (2002).

[20] Langille, M., Petre,I.: Simple gene assembly is deterministic. *Fundamenta Informaticae* **72** (2006) 1–12, IOS Press.

[21] Langille, M., Petre,I., Rogojin, V.: Three models for gene assembly in ciliates: a comparison. Computer Science Journal of Moldova, **18** (1), 1-26, 2010.

[22] Lekkerkerker, C., Boland, J.: Representation of a finite graph by a set of intervals on the real line, *Fundam. Math.* **51** (1962), 45–64

[23] Petre, I., Rozenberg, G.: Gene assembly in ciliates. Scholarpedia **5** (1), 9269, 2010.

[24] Ehrenfeucht, Prescott, D. M., Rozenberg, G.: Molecular operations for DNA processing in hypotrichous ciliates. *Europ. J. Protistology* **37** (2001) 241–260.

Paper II

Sepinoud Azimi, Tero Harju, Miika Langille, Ion Petre, Simple gene assembly as a rewriting of directed overlap-inclusion graphs. Theoretical Computer Science 454, 30-37, 2012.

Theoretical Computer Science 454 (2012) 30–37

Contents lists available at SciVerse ScienceDirect

Theoretical Computer Science

journal homepage: www.elsevier.com/locate/tcs

Simple gene assembly as a rewriting of directed overlap-inclusion graphs

Sepinoud Azimi [a,c], Tero Harju [b,c], Miika Langille [a], Ion Petre [a,c,*]

[a] Department of IT, Åbo Akademi University, Turku, Finland

[b] Department of Mathematics, University of Turku, Turku, Finland

[c] Turku Centre for Computer Science, Finland

ARTICLE INFO

Keywords:
Simple gene assembly
Directed overlap inclusion graphs
Graph-based simple operations

ABSTRACT

The simple intramolecular model for gene assembly in ciliates consists of three molecular operations, simple ld, simple hi and simple dlad. Mathematical models in terms of signed permutations and signed strings proved limited in capturing some of the combinatorial details of the simple gene assembly process. Brijder and Hoogeboom introduced a new model in terms of overlap-inclusion graphs which could describe two of the three operations of the model and their combinatorial properties. To capture the third operation, we extended their framework to directed overlap-inclusion (*DOI*) graphs in Azimi et al. (2011) [1]. In this paper we introduce DOI graph-based rewriting rules that capture all three operations of the simple gene assembly model and prove that they are equivalent to the string-based formalization of the model.

1. Introduction

Ciliates, an old group of unicellular eukaryotes, part of the phylum Ciliophora, have developed into a wide variety of lineages [7]. Ciliates possess two types of nuclei: macronuclei (abbreviated as *MAC*) and micronuclei (*MIC*) [7,2]. The ciliate genome is very differently organized in the two types of nuclei. The macronuclear genes are presented as contiguous sequences of nucleotides. In contrast, the micronuclear genes are split into blocks (macronuclear destined sequences, or MDSs) that are separated by noncoding sequences (internally eliminated sequences, or IESs), and may even be inverted. Following conjugation, ciliates destroy all their macronuclei and rebuild their macronuclear genes by assembling in the orthodox order all MDSs from their micronuclear genes in a process called gene assembly. The process is facilitated by some short, specific nucleotide sequences called *pointers* that are repeated at the end of an MDS and at the beginning of the MDS that should follow it in the orthodox order. We refer to [7,12] and references therein for details.

The gene assembly of ciliates has been investigated through several different molecular models, see [2]. One of them is the simple intramolecular model, originally introduced in [8], consisting of three molecular operations: the ld (*loop, direct-repeat excision*), the simple hi (*hairpin, inverted-repeat recombination*), and the simple dlad (*double-loop alternating direct-repeat recombination*) operations. This model could successfully predict the assembly of all currently known genes, see [10]. The simple model was modeled mathematically as a sorting of signed permutations in [11], a string rewriting system in [4,5] and as signed overlap-inclusion graphs in [3]. All these models were limited in capturing the details of the local interactions postulated by the simple model, especially in capturing the details of the simple dlad operations. To address this difficulty we extended in [1] the framework of [3] and defined the *directed* overlap-inclusion (in short, DOI) graphs as a model for

* Corresponding author at: Department of IT, Åbo Akademi University, Turku, Finland. Tel.: +358 22153361.
E-mail addresses: sepinoud.azimi@abo.fi (S. Azimi), tero.harju@utu.fi (T. Harju), miika.langille@gmail.com (M. Langille), ion.petre@abo.fi, ipetre@abo.fi (I. Petre).

0304-3975/$ – see front matter © 2012 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2012.04.018

ciliate genes. The extension is in fact quite small with respect to the overlap-inclusion graphs of [3]: we simply replace their undirected overlap edges with directed overlap edges where we additionally represent the order in which they occur in the string. In this way we replace a 'hybrid' directed/undirected graph with a directed graph that turns out to capture enough information to be able to represent all three operations of the simple model.

2. Preliminaries

We introduce here some of the notions and notations we use throughout the paper. For more details we refer to [7].

2.1. The simple gene assembly model

We focus in this paper on the intramolecular model for gene assembly in ciliates, introduced in [9,13]. The model consists of three molecular operations, all conjecturing the folding of the gene in a specific pattern (a loop, a hairpin, or a double loop) so that a pair of pointers (two pairs in the case of dlad) are aligned. Recombination on those pointers is thus enabled and as a result, two (or, in some special cases of dlad, even three) MDSs get spliced together to form a longer coding block. The gene assembly process is thus modeled as a computational process in which the the coding blocks get longer in each step, to eventually yield as an output the assembled gene.

The simple version of the intramolecular model, introduced in [10], assumes that the folds involved in each of the three operations are as simple as possible: in-between the aligned pointers there is a minimal number of other pointers (zero in the case of simple ld and simple dlad, one in the case of simple hi). The resulting model was shown in [10] to be capable of explaining the assembly of all currently known ciliate genes in [6]. We refer to [7] and [2] for more details, including figures of the folds of the three molecular operations of the model.

2.2. Legal strings

For an alphabet Σ and two strings u, v over Σ, we say that v is a *scattered substring* of u if $u = a_1a_2\dots a_n$ and $v = a_{i_1}a_{i_2}\dots a_{i_k}$, for some $0 \le k \le n$, $1 \le i_1 < \cdots < i_k \le n$, and $a_j \in \Sigma$, for all $1 \le j \le n$.

Let $\Delta_k = \{2, 3, \dots, k\}$ be an alphabet the elements of which are called *pointers* and let $\Sigma_k = \Delta_k \cup \{m\}$, for some $k \ge 1$ and $\overline{\Sigma_k} = \overline{\Delta_k} \cup \overline{m}$ be a signed copy of Σ_k, where letter m refers to a marker. We make no distinction on this level between the beginning and the ending markers – we simply treat them as being two occurrences of the special symbol m. For $a \in \Sigma_k \cup \overline{\Sigma_k}$, we define its unsigned copy $||a|| \in \Sigma_k$ to be a if $a \in \Sigma_k$ and $\overline{a}$ if $a \in \overline{\Sigma_k}$.

Let Σ_k^* be the set of all strings over Σ_k and let $\Sigma_k^{\pm} = (\Sigma_k \cup \overline{\Sigma_k})^*$ where for each $p \in \Sigma_k$, $\overline{\overline{p}} = p$.

We say that a string u in $\Sigma_k^{\pm}$ is *legal* if for any $p \in \Sigma_k$, u contains either 0 or 2 occurrences from the set $\{p, \overline{p}\}$. If u contains occurrences from the set $\{p, \overline{p}\}$, for some $p \in \Sigma_k$, then we say that p occurs in u. We define the *domain* of u as $\mathrm{dom}(u) = \{p \in \Sigma_k \mid p \text{ occurs in } u\}$. The ith occurrence in u (when scanned from left to right) of a pointer/marker from $\{p, \overline{p}\}$ is denoted by p^i, where $i = 1$ or $i = 2$. We say that u is *sorted* if $\mathrm{dom}(u) = \{m\}$; in other words, u is sorted if it only contains the two markers and no pointers.

For a legal string $u = p_1 \dots p_n$, with $p_i \in \Sigma_k \cup \overline{\Sigma_k}$, $1 \le i \le n$, we define its unsigned copy $||u|| = ||p_1|| \cdots ||p_n||$.

Let $p \in \Sigma_k \cup \overline{\Sigma_k}$ and let $u \in \Sigma_k^{\pm}$ be a legal string. If u contains both substrings p and $\overline{p}$ then p is said to be *positive* in u; otherwise, it is said to be *negative*. If $u = u_1p^1u_2p^2u_3$, with u_i, $i = 1, 2, 3$ strings over $\Sigma_k \cup \overline{\Sigma_k}$ and $p^1, p^2 \in \{p, \overline{p}\}$, then the *p-interval* of u is the substring u_2.

For any distinct $p, q \in \mathrm{dom}(u)$, p and q have one of the following relations:

- p and q *overlap* if exactly one occurrence from $\{p, \overline{p}\}$ can be found in the q-interval of u. We denote the overlap relation by $p \Rightarrow_u q$, if the first occurrence of p occurs in u before the first occurrence of q, and we denote it by $q \Rightarrow_u p$ otherwise;
- q is *included* in p if the two occurrences from $\{q, \overline{q}\}$ are found within the p-interval. This relation is denoted by $p \rightarrow_u q$;
- p and q are *disjoint* if they do not overlap and neither is included in the other in u.

2.3. Overlap inclusion graphs

Overlap-inclusion graphs have been introduced first in [3]. Using our notation, for a legal string u its overlap-inclusion graph $G = (V, \sigma, E)$ is defined as follows: $V = \mathrm{dom}(u)$, $\sigma : V \to \{+, -\}$ is the signing of its vertices such that for each $p \in V$, $\sigma(p) = +$ if p is a positive pointer in u and $\sigma(p) = -$ otherwise and $E = \{\{p, q\} \mid p \Rightarrow_u q \text{ or } q \Rightarrow_u p)\} \bigcup \{(p, q) \mid q \rightarrow_u p\}$. In this way, for any pair of overlapping pointers $\{p, q\}$ in u there is an undirected edge in G between p and q, and for any pointer q whose interval is included in the interval of some pointer p, G has the edge $p \rightarrow_G q$ from p to q.

Example 1. The overlap-inclusion graph corresponding to $u = 255m\overline{2}343m\overline{4}$ is shown in Fig. 1(a).

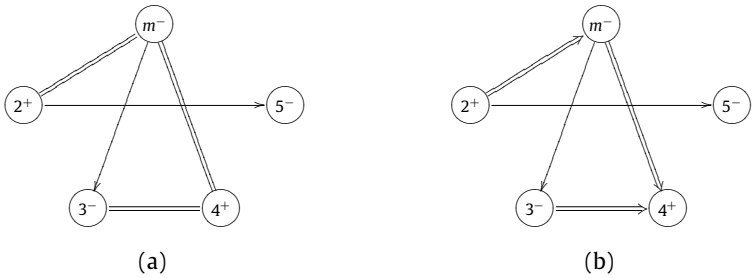


Fig. 1. (a) The overlap-inclusion graph and (b) the directed overlap-inclusion graph corresponding to string $u = 255m\bar{2}343m\bar{4}$.

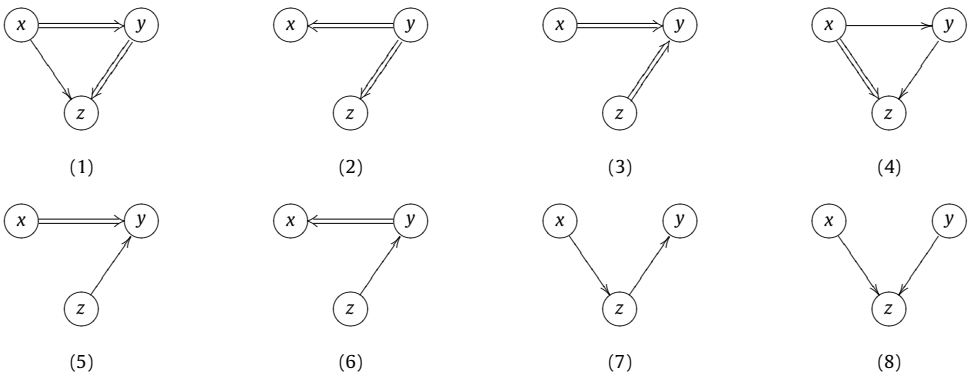


Fig. 2. All 3-vertex, acyclic, forbidden graphs. Inclusion edges are illustrated as simple arrows and overlap edges as double arrows.

2.4. Directed overlap-inclusion graphs

We recall here the notion of directed overlap-inclusion *(DOI)* graphs introduced in [1] and recall some of their properties. We first recall that a directed graph G is called *connected* if for any distinct vertices u and v of G, there is either a (directed) path from u to v, or a (directed) path from v to u. We also recall that for a set of vertices U of G we denote by $G \setminus U$ the graph obtained from G by removing all vertices in U and all edges incident to them.

Definition 1 (*[1]*)**.** Let u be a legal string over some Σ_k. The *directed overlap-inclusion (DOI) graph* $G_u = (V, \sigma, E_o, E_i)$ corresponding to u is defined as follows: $V = \mathrm{dom}(u)$ is the set of vertices, $\sigma : V \to \{+, -\}$ is the signing of its vertices such that for each $p \in V$, $\sigma(p) = +$ if p is a positive pointer in u and $\sigma(p) = -$ otherwise. E_o and E_i are sets of its directed edges, $E_o = \{(p, q) \mid p \Rightarrow_u q\}$ and $E_i = \{(p, q) \mid p \to_u q\}$. For a DOI graph G and any string u such that $G = G_u$ we say that u corresponds to G.

Example 2. The DOI graph corresponding to string $u = 255m\bar{2}343m\bar{4}$ is shown in Fig. 1(b).

Definition 2. Let G be a directed, vertex- and edge-labeled graph. We say that G is *forbidden* if there is no string u such that G is isomorphic to the *DOI* graph corresponding to u.

The following results have been proved in [1].

Theorem 1 (*[1]*)**.** *Let G be a directed labeled graph with $\{+, -\}$ as vertex labels and with two edge colors as edge labels. If G is a 3-vertex, acyclic graph, then G is* forbidden *if and only if it is isomorphic to one of the graphs in Fig. 2.*

Lemma 2 (*[1]*)**.** *Let u be a legal string and G the DOI graph corresponding to $||u||$. Assume that G contains the path $s_1 \Rightarrow_G s_2 \Rightarrow_G \cdots \Rightarrow_G s_n$, $s_i \neq s_j$ for $i \neq j$. Then the first occurrences of the pointers in $||u||$ appear in the order $s_1 s_2 \cdots s_n$; the same holds also for the sequence of their second occurrences. Moreover, if $s_1 \Rightarrow_G s_n$, then $s_i \Rightarrow_G s_j$ for all $1 \leq i < j \leq n$, and $||u|| = s_1 s_2 \cdots s_n s_1 s_2 \cdots s_n$.*

Theorem 3 (*[1]*)**.** *Any DOI graph G is a directed acyclic graph.*

Example 3. We notice that there can exist more than one string corresponding to a DOI graph. For example, the strings 245566324377, 246655324377, 772455663243 and 772466553243 have the same DOI graph as shown in Fig. 3.

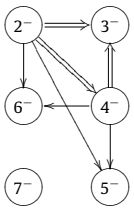


Fig. 3. A graph with more than one string corresponding to it.

3. The strings-to-DOI graphs mapping

We discuss in this section the injectivity of the strings-to-DOI graphs mapping. While in general the same DOI graph can correspond to several strings, we show that this is not true for the realistic strings and for the overlap-only connected DOI graphs.

Lemma 4. *Let G be a connected DOI graph consisting of negative vertices and overlap edges only. Then there is a unique string u_G corresponding to G.*

Proof. Consider the non-trivial case of a graph with at least two vertices. It follows from Theorem 3 that in any connected component G with at least two vertices there is a vertex with indegree 0. Since G is connected, there is only one such vertex; we denote it by p. Similarly, there exists a unique vertex, say q with outdegree 0.

We prove now that there is a directed path from p to q visiting all vertices of G. Consider the longest path P from p to q, say $p = p_1 \Rightarrow p_2 \Rightarrow \cdots \Rightarrow p_n = q$, $n \geq 2$; since G is connected, such a path exists. Assume that there is a vertex r not visited by this path. Then there is $1 \leq i < n$ and an edge $p_i \Rightarrow r$ with r not visited by P. Without loss of generality, assume that i is the largest with this property. Note now that there must be an overlap edge between p_{i+1} and r since otherwise the subgraph induced by p_i, p_{i+1} and r would be forbidden, being isomorphic with the graph in Fig. 2(2). Based on the maximality of i, there cannot be an edge from p_{i+1} to r. Consequently, $r \Rightarrow p_{i+1}$ is an edge in G and so, $p = p_1 \Rightarrow p_1 \Rightarrow \cdots \Rightarrow p_i \Rightarrow r \Rightarrow p_{i+1} \Rightarrow \cdots \Rightarrow p_n = q$ is a path from p to q in G longer than P, a contradiction.

Consider now a string u such that $G_u = G$. It follows from Lemma 2 that the first occurrences of all letters of u appear in u in the order $p_1 p_2 \cdots p_n$; the same holds also for their second occurrences. To prove the uniqueness of u it is enough to fix the position of the second occurrence of all letters p_i, $1 \leq i < n$. Let l_i be the largest j with $i < j \leq n$ such that $p_i \Rightarrow p_j$. It follows from Lemma 2 that $p_i p_{i+1} \cdots p_{l_i} p_i p_{i+1} \cdots p_{l_i}$ is a scattered substring of u, i.e., the second occurrence of p_i appears in u after the first occurrence of p_{l_j}. Moreover, the second occurrence of p_i must appear before the first occurrence of p_{l_i+1} since otherwise $p_i \Rightarrow p_{l_i+1}$, contradicting the definition of l_i. □

Corollary 5. *Let G be a connected DOI graph consisting of overlap edges only. Then there is a unique string u_G corresponding to G, modulo switching the sign of both occurrences of any letter of u_G.*

Proof. The result follows from Lemma 4 by noting that if a string v is obtained from u by switching the signs of both occurrences of a letter i occurring in u, then $G_u = G_v$. □

Note that the result analogous to Corollary 5 for inclusion-only graphs does not hold. For example, *pqqrrp* and *prrqqp* have the same DOI graph.

We discuss next the case of the so-called realistic strings and we show that the string-to-DOI graphs mapping is injective for this type of strings. Realistic legal strings correspond to the genome structure of ciliates, see [7] for details.

In the following, for clarity, we identify the marker m with the pointer 1, and we consider the pointers modulo k. In this case $k + 1 \equiv 1 \pmod{k}$, i.e., m follows k. For this reason, let $\Gamma_k = \{1, 2, \ldots, k\}$, and let $\Gamma_k^{\circledast}$ denote the strings over the alphabet $\Gamma \cup \overline{\Gamma}_k$ in the natural manner.

A legal string $u \in \Gamma_k^{\circledast}$ is called *realistic*, if $u = \varphi(w)$, where

- $w \in \Gamma_k^{\circledast}$ is a signed permutation, i.e., each symbol $p \in \Gamma_k$ occurs in w exactly once - either in the form p or $\overline{p}$;
- $\varphi\colon \Gamma_k^{\circledast} \to \Gamma_k^{\circledast}$ is a homomorphism that satisfies the substitution laws

$$\varphi(p) = p(p+1), \quad \varphi(\overline{p}) = (\overline{p+1})\overline{p}.$$

By our convention, we have $\varphi(k) = k\,1$ and $\varphi(\overline{k}) = \overline{1}\,\overline{k}$. For example, if $k = 4$ then $w = 3\overline{1}\overline{4}2$ is a signed permutation that gives the realistic string $\varphi(w) = 3412\overline{1}\overline{4}23$.

Theorem 6. *For any realistic strings u and v we have that $G_u = G_v$ if and only if $u = v$.*

Proof. Let u and v be two different realistic strings corresponding to a common DOI graph G. Therefore there are permutations w and w' such that $u = \varphi(w)$ and $v = \varphi(w')$. If $w = qs$ and $w' = qs'$ for a common prefix symbol $q \in \Gamma_k \cup \overline{\Gamma_k}$ and strings s and s', then also $G_{\varphi(sq)} = G_{\varphi(s'q)}$. Hence we can assume that the permutations w and w' do not have any

nonempty common prefix, i.e., their first symbols are different. It follows, by the definition of φ, that also the first pointers in u and v are different. Let then

$$u = pp_2 \dots p_n \quad \text{and} \quad v = qq_2 \dots q_n;$$

where $n = 2k$, $p, q, p_i, q_i \in \Gamma_k \cup \overline{\Gamma_k}$ for each i, and $p \neq q$.

Since $G_u = G_v$, the p-intervals $I_u(p)$ of u and $I_v(p)$ of v contain the same number of each pointer from Γ_k.

Suppose first that $q = \overline{p}$. By symmetry, we may assume that $p \in \Gamma_k$. It follows that $u = p(p+1)\alpha$ and $v = \overline{p}\overline{(p-1)}\beta$ for some strings α and β. Since the sign of p is the same in G_u and G_v, we have either

$$\begin{aligned} u &= p(p+1)\cdots(p-1)p\sigma_u, \\ v &= \overline{p}\overline{(p-1)}\cdots\overline{(p+1)}\overline{p}\sigma_v, \end{aligned} \tag{1}$$

or

$$\begin{aligned} u &= p(p+1)\cdots\overline{p}\overline{(p-1)}\sigma_u, \\ v &= \overline{p}\overline{(p-1)}\cdots p(p+1)\sigma_v, \end{aligned} \tag{2}$$

because u and v are realistic.

In Case (1), the pointer $p+1$ occurs once in the p-interval $I_u(p)$ of u, and hence it occurs also once in the p-interval $I_v(p)$ of v, i.e., $\varphi(p+1) = (p+1)(p+2)$ is in $I_v(p)$. Moreover, $p+2$ occurs only once in $I_v(p)$, since $p+1$ does so in the interval $I_u(p)$ of u. We proceed by induction to obtain that the intervals $I_u(p)$ and $I_v(p)$ contain a unique occurrence of every pointer $\Gamma_k \setminus \{p\}$.

Now, the second occurrence of $p+1$ in u and v lies in the suffix σ_u and σ_v, respectively.

Let then t with $t \neq p, p+1$ be a pointer. From the form of the string u, we deduce that $p+1 \Rightarrow t$ or $p+1 \to t$, since one occurrence of t belongs to the p-interval. The form of the string v yields that the second possibility cannot hold. Thus $p+1 \Rightarrow t$ for all pointers $t \neq p, p+1$. But clearly, in the string v this does not hold for $t = p-1$, because the first occurrence of $p-1$ comes before than the first occurrence of $p+1$ in v. This contradiction shows that Case (1) does not hold.

In Case (2), the pointer $p-1$ occurs once in the p-interval $I_u(p)$, and an analogous reasoning to the previous gives that each pointer different from p has a unique occurrence in the p-intervals of u and v. As in the case for (1), we derive a contradiction.

We now assume that $q \notin \{p, \overline{p}\}$.

Note that we do not have any overlap edge in G between the pointers p and q: Indeed, if $p \Rightarrow q$, then $p \Rightarrow q$ in the string u and $q \Rightarrow p$ in the string v. Note also that the intervals $I_u(p)$ and $I_v(q)$ cannot be included in the other one.

We can now write

$$\begin{aligned} u &= p^1\delta_1 p^2\delta_2 q^1\delta_3 q^2\delta_4 \\ v &= q^1\delta_1' q^2\delta_2' p^1\delta_3' p^2\delta_4'. \end{aligned}$$

We define $u_1 = p^1\delta_1 p^2\delta_2$, $u_2 = q^1\delta_3 q^2\delta_4$, $v_1 = q^1\delta_1' q^2\delta_2'$, $p^1\delta_3' p^2\delta_4'$. We claim that u_1 and u_2 do not contain common pointers, and neither do v_1 and v_2.

Let t be a pointer such that $t^1 \in \mathrm{dom}(u_1)$ and $t^2 \in \mathrm{dom}(u_2)$. If $t^2 \in \mathrm{dom}(\delta_3)$ then, $t \Rightarrow q$, which is a contradiction since in v the pointer q starts the word. Similarly, if $t^2 \in \mathrm{dom}(\delta_4)$ then $t \to q$, which is again a contradiction. A similar argument gives that v_1 an v_2 are disjoint of pointers.

By our assumptions, we can again assume that u is of one of the following forms:

$$\begin{aligned} u &= p(p+1)\cdots(p-1)p\delta_2 q^1\delta_3 q^2\delta_4, \\ u &= p(p+1)\cdots\overline{p}\overline{(p-1)}\delta_2 q^1\delta_3 q^2\delta_4. \end{aligned}$$

Since u_1 and u_2 are disjoint of pointers. If a pointer t occurs in u_1, then both occurrences of t are in u_1, and either $t = q-1$ or also $t+1$ (modulo k) is in u_1, since $\varphi(t) = t(t+1)$, i.e., $t+1$ leans on t inside u_1. Hence, we obtain inductively that all pointers $p, p+1, \dots, q-1$ (modulo k) are in u_1. Similarly, by going in the other direction down from p, we have that the pointers $p, p-1, p-2, \dots, q+1$ (modulo k) are all in u_1. However, $u_2 \neq q^1q^2$ by the form of $u = \varphi(w_u)$. This contradiction proves the claim. □

4. Simple gene assembly in ciliates

We introduce in this section our DOI-graph-based model for simple gene assembly in ciliates. We first recall the formulation of the model in terms of legal strings.

4.1. Simple gene assembly on legal strings

A gene can be represented as a legal string by simply denoting it as a sequence of pointers and markers. The three molecular operations are defined on legal strings as follows.

Definition 3. The string pointer reduction system is formalized as follows. In each case $p, q \in \Delta_k$ are distinct pointers and $u_1, u_2, u_3 \in \Sigma^{*}$.

i. The *simple string negative* rule ssn_p is defined as follows:

$$\mathsf{ssn}_p(u_1\tilde{p}\tilde{p}u_2) = u_1u_2, \qquad \mathsf{ssn}_p(\tilde{p}u_3\tilde{p}) = u_3, ,$$

where $\tilde{p} \in \{p, \overline{p}\}$ and u_3 contains only markers (boundary case). We denote $\mathsf{Ssn} = \{\mathsf{ssn}_p \mid p \in \Delta_k, k \geq 2\}$.

ii. The *simple string positive* rule ssp_p for a pointer p is defined as follows:

$$\mathsf{ssp}_p(u_1\tilde{p}q\tilde{\overline{p}}u_3) = u_1\overline{q}u_3,$$

where $\tilde{p} \in \{p, \overline{p}\}$ and $\tilde{q} \in \{q, \overline{q}\}$. We denote $\mathsf{Ssp} = \{\mathsf{ssp}_p \mid p \in \Delta_k, k \geq 2\}$.

iii. The *simple string double* rule $\mathsf{ssd}_{p,q}$ for pointers p and q is defined as follows:

$$\mathsf{ssd}_{p,q}(u_1\tilde{p}\tilde{q}u_3\tilde{p}\tilde{q}u_5) = u_1u_3u_5,$$

where $\tilde{p}\tilde{q} \in \{pq, \overline{pq}\}$. We denote $\mathsf{Ssd} = \{\mathsf{ssd}_{p,q} \mid p, q \in \Delta_k, k \geq 2\}$.

The string-based versions of the simple operations are based on "locality": only pointers that are at a minimal distance from each other can be removed by such an operation. This concept of locality is however lost when going from a legal string to its associated overlap graph since overlap graphs do not contain the information about the order of pointers in their corresponding strings. By adding the concept of "inclusion" to overlap graphs it is possible to capture the information about the domains of p – intervals for each pointer p.

4.2. Simple gene assembly on DOI graphs

Definition 4. Let G be a DOI graph and p an arbitrary vertex of G. We introduce the following terms:

i. *Incoming inclusion edges*: we denote by $inSet_i(p)$ the set of all vertices q such that $q \to p$ is an (inclusion) edge in G. Also, $inDeg_i(p)$ is the number of vertices in $inSet_i(p)$.
ii. *Outgoing inclusion edges*: we denote by $outSet_i(p)$ the set of all vertices q such that $p \to q$ is an (inclusion) edge in G. Also, $outDeg_i(p)$ is the number of vertices in $outSet_i(p)$.
iii. *Incoming overlap edges*: we denote by $inSet_o(p)$ the set of all vertices q such that $q \Rightarrow p$ is an (overlap) edge in G. Moreover, $inDeg_o(p)$ is the number of vertices in $inSet_o(p)$.
iv. *Outgoing overlap edges*: we denote by $outSet_o(p)$ the set of all vertices q such that $p \Rightarrow q$ is an (overlap) edge in G. Also, $outDeg_o(p)$ is the number of vertices in $outSet_o(p)$.

We define now the DOI graphs-based version of our gene assembly operations: *simple graph negative rule* sgn, *simple graph positive rule* sgp, and *simple graph double rule* sgd.

Definition 5. Let $G = (V, \sigma, E_o, E_i)$ be a DOI graph be a directed, vertex- and edge-labeled graph. For any distinct vertices $p, q \in V \setminus \{m\}$, the graph operations sgn_p, sgp_p and $\mathsf{sgd}_{p,q}$ are defined on G as follows:

(i) *The simple graph negative rule* sgn for p, denoted sgn_p, is applicable to G if $\sigma(p) = -$ and $inDeg_o(p) = outDeg_o(p) = outDeg_i(p) = 0$. In this case, $\mathsf{sgn}_p(G) = G \setminus \{p\}$.

We denote $\mathsf{Sgn} = \{\mathsf{sgn}_p \mid p \in \Delta_k, k \geq 2\}$. We say that sgn_p *corresponds* to a string-rewriting rule ssn_p.

(ii) *The simple graph positive rule* sgp for p, denoted sgp_p, is applicable to G if $\sigma(p) = +$, $inDeg_o(p) + outDeg_o(p) = 1$, and $outDeg_i(p) = 0$. Let q be the vertex with the property $inSet_o(p) \cup outSet_o(p) = \{q\}$. In this case, $\mathsf{sgp}_p(G)$ is the graph obtained from $G \setminus \{p\}$ by switching the label of q: q is negative in $\mathsf{sgp}_p(G)$ if and only if it is positive in G.

We denote $\mathsf{Sgp} = \{\mathsf{sgp}_p \mid p \in \Delta_k, k \geq 2\}$. We say that sgp_p *corresponds* to a string-rewriting rule ssp_p.

(iii) *The simple graph double rule* sgd for p, q, denoted $\mathsf{sgd}_{p,q}$, is applicable to G if:

- $\sigma(p) = \sigma(q) = -$,
- $q \in outSet_o(p)$,
- $inSet_o(p) \cup p = inSet_o(q)$,
- $outSet_o(p) = outSet_o(q) \cup q$,
- $inSet_i(p) = inSet_i(q)$ and
- $outSet_i(p) = outSet_i(q)$.

In this case, $\mathsf{sgd}_{p,q}(G) = G \setminus \{p, q\}$.

We denote $\mathsf{Sgd} = \{\mathsf{sgd}_{p,q} \mid p, q \in \Delta_k, k \geq 2\}$. We say that $\mathsf{sgd}_{p,q}$ corresponds to a string-rewriting rule $\mathsf{ssd}_{p,q}$.

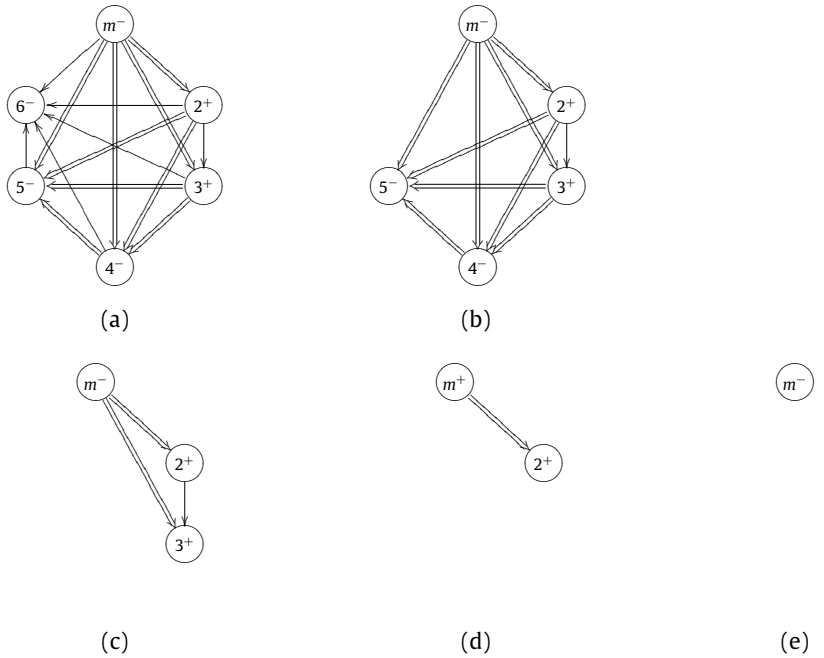


Fig. 4. Reducing overlap-inclusion graph: (a) G is the DOI graph corresponding to $u = m234566m\overline{32}45$, (b) $G' = \mathsf{sgn}_6(G)$ corresponds to $u' = \mathsf{sgn}_6(u) = m2345m\overline{32}45$, (c) $G'' = \mathsf{sgd}_{4,5}(G')$ corresponds to $u'' = \mathsf{sgd}_{4,5}(u') = m23m\overline{32}$, (d) $G''' = \mathsf{sgp}_3(G'')$ corresponds to $u''' = \mathsf{sgp}_3(u'') = m2\overline{m2}$, (e) $G^{iv} = \mathsf{sgp}_2(G''')$ corresponds to $u^{iv} = \mathsf{sgp}_2(u''') = mm$.

Example 4. Consider the DOI graph G corresponding to $u = m234566m\overline{32}45$, see Fig. 4(a). Clearly, sgn_6 is applicable to G since vertex 6 is negative and $inDeg_o(6) = outDeg_o(6) = outDeg_i(6) = 0$. The result is the graph G' illustrated in Fig. 4(b): vertex 6 is removed with all its incident edges. We can apply to G' operation $\mathsf{sgd}_{4,5}$, since both 4 and 5 are negative, edge $4 \Rightarrow 5$ is present in G' and there are no other paths from 4 to 5, sets of incoming overlap edges into 4 and 5 are the same with the exception of edge $4 \Rightarrow 5$, sets of outgoing overlap edges from 4 and 5 are the same with the exception of $4 \Rightarrow 5$, and sets of incoming and outgoing include edges of 4 and 5 are equal (empty). As a result of the application of $\mathsf{sgd}_{4,5}$ we obtain graph G'' depicted in Fig. 4(c): vertices 4 and 5 are removed together with all their incident edges. Then, we can apply sgp_3 to G'', since 3 is positive in G'', it has one incoming overlap edge and no outgoing edges (both overlap and inclusion). Consequently we obtain graph G''' (see Fig. 4(d)) with vertex 3 removed together with its incident edges and vertex m changed its sign from *negative* to *positive*. Then, we can apply sgp_2 to G''' and finally obtain single isolated negative vertex m.

Next we prove that the string-based model for simple gene assembly and the DOI graph-based one are equivalent.

Theorem 7. *Let u be a legal string, G_u its DOI graph, and $\psi \in \mathsf{Ssn} \cup \mathsf{Ssp} \cup \mathsf{Ssd}$. Let $\phi \in \mathsf{Sgn} \cup \mathsf{Sgp} \cup \mathsf{Sgd}$ be the DOI graph operation corresponding to ψ. Then ϕ is applicable to G_u and $G_{\psi(u)} = \phi(G_u)$.*

Proof. We prove the claim inductively.

Case 1: $\psi = \mathsf{ssn}_p$. In this case $u = u_1 p p u_2$, for some strings u_1, u_2. Thus there are no overlap edges incident to p in G_u and no inclusion edges starting from p. Thus, $\mathsf{sgn}_p(u)$ is applicable to G_u; $\mathsf{sgn}_p(u)$ removes the pointer p and any possible incoming inclusion relations, yielding the graph corresponding to $\mathsf{sgn}_p(G_u)$.

Case 2: $\psi = \mathsf{ssp}_p$. In this case $u = u_1 p u_2 p u_3$, for some strings u_1, u_2, u_3, where $|u_2| = 1$. Thus p must have a single (outgoing or incoming) overlap edge, and cannot have any outgoing inclusion edges. Operation $\mathsf{ssp}_p(u)$ removes the pointer p and all inclusion and overlap relationships connected to it, in addition to reversing the sign of the pointer found in u_2. The result follows from the definition of $\mathsf{sgp}_p(G_u)$.

Case 3: $\psi = \mathsf{ssd}_{p,q}$. In this case $u = u_1 p q u_2 p q u_3$. Thus, p, q overlap in u and as they are adjacent to each other in the string, the corresponding vertices in G_u have the same overlap and inclusion neighbors, except for each other. Operation $\mathsf{ssd}_{p,q}(u)$ removes both pointers, thus yielding the DOI graph corresponding to $\mathsf{sgd}_{p,q}(G_u)$. □

Theorem 8. *Let G be a DOI graph corresponding to a legal string u. Let $G' = \phi(G)$, where $\phi \in \mathsf{Sgn} \cup \mathsf{Sgp} \cup \mathsf{Sgd}$ and ψ is the string-based operation corresponding to ϕ. Then ψ is applicable to u and $\psi(u)$ is a string corresponding to $\phi(G)$.*

Proof. We discuss the following three cases.

Case 1: $\phi = \mathsf{sgn}_p(G)$. Vertex p has no overlap and no outgoing inclusion edge, therefore, any corresponding string must have the form $u = u_1 p p u_2$. The claim follows since $\mathsf{ssn}_p(u) = u_1 u_2$.

Case 2: $\phi = \mathrm{sgp}_p(G)$. Vertex p has only one overlap edge, either outgoing or incoming and no outgoing inclusion edge, therefore, any corresponding string must have the form $u = u_1pq\overline{p}u_2$, or $u = u_1\overline{p}qpu_2$. To obtain G' from G, vertex p is removed and the signing of the vertex which is overlap-adjacent to p will be reversed. The claim follows since $\mathrm{ssp}_p(u) = u_1\overline{q}u_2$.

Case 3: $\phi = \mathrm{sgd}_{p,q}(G)$. By the definition of $\mathrm{sgd}_{p,q}$, we have that $\sigma(p) = \sigma(q) = -$, $q \in outSet_o(p)$ and $inSet_o(p) \cup p = inSet_o(q)$, $outSet_o(p) = outSet_o(q) \cup q$, $inSet_i(p) = inSet_i(q)$ and $outSet_i(p) = outSet_i(q)$. It follows then that $u = u_1p\alpha qu_2p\beta qu_3$, for some string $u_1, u_2, u_3, \alpha, \beta$. We claim that α and β are both empty. If α were not empty, then let r be a pointer occurring in α. Depending on where the other occurrence from the pointer set $\{r, \overline{r}\}$, we get a contradiction with the relationships above on the neighborhoods of p and$q$. A similar argument holds for β. Thus, $u = u_1pqu_2pqu_3$. If $G' = \mathrm{sgd}_{p,q}(G)$, then in G', p, q are removed, as well as all the edges incident to them, yielding the DOI graph corresponding to $\mathrm{ssd}_{p,q}(u) = u_1u_2u_3$. □

Corollary 9. *Let G be a DOI graph and $\phi \in \mathsf{Sgn} \cup \mathsf{Sgp} \cup \mathsf{Sgd}$ applicable to G. Then $\phi(G)$ is also a DOI graph.*

Example 5. Consider string $u = b234566e\overline{3}\overline{2}45$ from Example 4. Its overlap-inclusion graph G is presented in Fig. 4(a). If we apply ssn_6 to u, we obtain string $u' = b2345e\overline{3}\overline{2}45$ to which graph $G' = \mathrm{sgn}_6(G)$ from Fig. 4(b) corresponds. If we apply to u' operation $\mathrm{ssd}_{4,5}$, then we obtain string $u'' = b23e\overline{3}\overline{2}$ to which graph $G'' = \mathrm{sgd}_{4,5}(G')$ from Fig. 4(c) corresponds. By applying ssp_3 to u'' we obtain string $u''' = b2\overline{e}\overline{2}$ to which graph $G''' = \mathrm{sgp}_3(G'')$ from Fig. 4(d) corresponds. Finally, by applying ssp_2 to u''' we obtain string be which corresponds to graph $\mathrm{sgp}_2(G''')$ consisting of a single negative vertex m, see Fig. 4(e).

5. Discussion

In this paper we introduced a graph-based formalization of the simple gene assembly operations. The formalization is based on a new type of graphs – DOI graphs – as a model for the pointer structure of ciliate genes. The DOI graphs extend a closely related type of graph, overlap-inclusion graphs, introduced in [3] for the same purpose. While the overlap graphs were used successfully in [3] to represent two of the three simple gene assembly operations, they could not be used for defining the third operation, the so-called simple double rule, called also the simple double-loop alternating direct repeat rule in [7]. We proved here that our DOI graph-based rules are equivalent with the string-based rewriting system for simple gene assembly, thus capturing successfully all three operations of the model, answering to the open problem formulated in [3].

A number of interesting problems remain open in connection to the DOI graphs and the graph-based simple operations; we mention only one here, concerned with the characterization of the reduction power of the simple operations: which graphs can be reduced by using only negative rules, only positive rules, only double rules, only negative and positive rules, etc.? The power of the negative and the positive rules (both by themselves and in combination with each other) was characterized in [3] in terms of overlap-inclusion graphs and the results can be extended to DOI graphs as well; the counterparts of these results where double rules are involved remain open.

References

[1] S. Azimi, T. Harju, M. Langille, I. Petre, V. Rogojin, Directed overlap-inclusion graphs as representations of ciliate genes, Fundamenta Informaticae 110 (1) (2011) 29–44.
[2] R. Brijder, T. Harju, N. Jonoska, I. Petre, G. Rozenberg, Gene assembly in ciliates, in: J.N. Kok, G. Rozenberg, T.H.W. Back (Eds.), Handbook of Natural Computing, in: Molecular Computation, vol. 2, Springer, 2012.
[3] R. Brijder, H.J. Hoogeboom, Combining overlap and containment for gene assembly in ciliates, Theoretical Computer Science 411 (6) (2010) 897–905.
[4] R. Brijder, M. Langille, I. Petre, A string-based model for simple gene assembly, in: Z. Esik, E. Csuhaj-Varju (Eds.), Proceedings of FCT 2007, in: Lecture Notes in Computer Science, vol. 4639, Springer, 2007, pp. 161–172.
[5] R. Brijder, M. Langille, I. Petre, Extended strings and graphs for simple gene assembly, Theoretical Computer Science 411 (4–5) (2010) 730–738.
[6] A.R.O. Cavalcanti, T.H. Clarke, L.F. Landweber, Mds_ies_db: a database of macronuclear and micronuclear genes in spirotrichous ciliates, Nucleic Acids Research 33 (1) (2005) D396.
[7] A. Ehrenfeucht, T. Harju, I. Petre, D.M. Prescott, G. Rozenberg, Computation in Living Cells: Gene Assembly in Ciliates, Springer, 2004.
[8] A. Ehrenfeucht, I. Petre, D.M. Prescott, G. Rozenberg, Universal and simple operations for gene assembly in ciliates, in: C. Martin-Vide, V. Mitrana (Eds.), Words, Sequences, Languages: Where computer science, biology and linguistics come across, Kluwer, 2000, pp. 329–342.
[9] A. Ehrenfeucht, D.M. Prescott, G. Rozenberg, Computational aspects of gene (un) scrambling in ciliates, in: E. Winfree, L.F. Landweber (Eds.), Evolution as Computation, Springer, 2001, pp. 216–256.
[10] T. Harju, I. Petre, V. Rogojin, G. Rozenberg, Patterns of simple gene assembly in ciliates, Discrete Applied Mathematics 156 (14) (2008) 2581–2597.
[11] T. Harju, G. Rozenberg, Computational processes in living cells: Gene assembly in ciliates, Lecture Notes in Computer Science 2450 (2003) 1–20.
[12] I. Petre, G. Rozenberg, Gene assembly in ciliates, Scholarpedia 5 (1) (2010) 9269.
[13] D.M. Prescott, A. Ehrenfeucht, G. Rozenberg, Molecular operations for dna processing in hypotrichous ciliates, European Journal of Protistology 37 (3) (2001) 241–260.

Paper III

Sepinoud Azimi, Ion Petre, The reduction power of simple operations for gene assembly in ciliates. Discrete Mathematics and Computer Science 1, 23-36, 2014.

The Reduction Power Of Simple Operations For Gene Assembly In Ciliates

Sepinoud Azimi[1,2,3] and Ion Petre[1,2,3]
{sazimi,ipetre}@abo.fi

[1] Computational Biomodeling Laboratory
[2] Turku Centre for Computer Science
[3] Department of IT, Åbo Akademi University
Joukahainengatan 3-5, FIN-20520 Åbo

Abstract. The simple intramolecular model for gene assembly in ciliates predicts correctly the assembly of all currently known ciliate gene patterns. The model consists of three molecular operations: the ld (*loop, direct-repeat excision*), the simple hi (*hairpin, inverted-repeat recombination*), and the simple dlad (*double-loop alternating direct-repeat recombination*) operations. The gene transformations conjectured by the simple intramolecular model for gene assembly can be studied as operations on the so-called directed-overlap inclusion (in short, DOI) graphs introduced in [2]. In this paper we focus on characterizing the DOI graphs that are reducible using only some combinations of the three simple operations. We also show that the DOI graph model is confluent.

Keywords: Directed overlap-inclusion graphs, gene assembly in ciliates, simple operations, confluent.

1 Introduction

Ciliates are an ancient group of unicellular eukaryotes which possess two types of nuclei: macronuclei (*MAC*) and micronuclei (*MIC*) [8, 3]. Both types of nuclei are sequences of building blocks called *macronuclear destined sequences*, or *MDS*s. In macronuclear genes the MDSs are presented in an orthodox order, whereas in micronuclear genes they are shuffled, as well as separated by non-coding sequences called *internally eliminated sequences*, or *IES*s. During the process of sexual conjugation, the old macronucleus disintegrates and a new one is developed from the micronucleus. In this process, the micronuclear genes get transformed to their macronuclear form by having their IESs removed and their MDSs sorted in the orthodox order; this is facilitated by some short, specific nucleotide sequences called *pointers* that are repeated at the end of an MDS and at the beginning of the following MDS in macronuclear gene. This process is called *gene assembly*, see [8, 13].

We focus in this paper on the intramolecular model for gene assembly in ciliates, introduced in [9, 14]. The model consists of three molecular operations,

ld, hi, and dlad, all conjecturing the folding of the gene in a specific pattern (a loop, a hairpin, or a double loop) so that a pair of pointers (two pairs in the case of dlad) are aligned. We illustrate the folding and recombinations involved in each operations in Figure 1 and refer for details on these operations to [8, 3]. The simple version of the intramolecular model, introduced in [10], assumes that the folds involved in each of the three operations are as simple as possible: in-between the aligned pointers there is a minimal number of other pointers (zero in the case of simple ld and simple dlad, one in the case of simple hi). The resulting model was shown in [10] to be capable of explaining the assembly of all currently known ciliate genes in [7]; we refer to [8] and [3] for more details. Various modeling frameworks have been proposed to represent the gene assembly in ciliates, ranging from signed permutations [11] to a string rewriting system [5, 6] to graph-based models [4, 1]. In this study, we investigate the reduction power of simple operations on the latest of such representations called *directed overlap inclusion (DOI) graphs* introduced in [1].

2 Preliminaries

We introduce here some of the notions and notations we use throughout the paper. For more details we refer to [8].

2.1 Legal Strings

Let $\Delta_k = \{2, 3, \ldots, k\}$, for some $k \geq 1$, be an alphabet whose letters are called *pointers*, and let $\Sigma_k = \Delta_k \cup \{m\}$, where letter m refers to the (begining and ending) marker. We also denote the signed copy of Σ_k by $\overline{\Sigma}_k = \overline{\Delta}_k \cup \{\overline{m}\}$, where $\Sigma_k \cap \overline{\Sigma}_k = \emptyset$. We make the convention that $\bar{\bar{p}} = p$ for all $p \in \Sigma_k \cup \{m\}$. Let $\Sigma_k^{\maltese} = (\Sigma_k \cup \overline{\Sigma}_k)^*$.

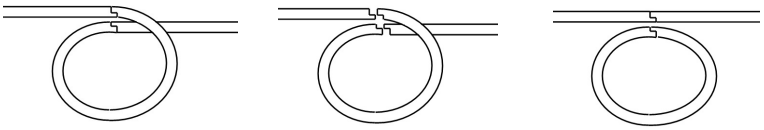
String $u \in \Sigma_k^{\maltese}$ is called a *legal string* if u contains two occurrences from the marker set $\{m, \bar{m}\}$ and for any $p \in \Delta_k$, u contains either 0 or 2 occurrences from the set $\{p, \bar{p}\}$. We define the *domain* of u as $\mathsf{dom}(u) = \{p \in \Sigma_k \mid$ either p or $\bar{p}$ occurs in $u\}$. We say that u is *sorted* if $\mathsf{dom}(u) = \{m\}$.

Let $p \in \Sigma_k \cup \bar{\Sigma}_k$ and let $u \in \Sigma_k^{\maltese}$ be a legal string, we say that p is *positive* if p and $\bar{p}$ occurs in u, and we say that it is *negative* otherwise. Let $u = u_1 p' u_2 p'' u_3$, where u_1, u_2 and u_3 are strings over $\Sigma_k \cup \overline{\Sigma}_k$ and $p', p'' \in \{p, \overline{p}\}$; substring u_2 is called the *p-interval* of u.

For any distinct $p, q \in \mathsf{dom}(u)$, p and q have one of the following relations:

- p and q *overlap* in u if exactly one occurrence from $\{p, \overline{p}\}$ can be found in the q-interval of u. We denote the overlap relation by $p \Rightarrow_u q$, if the first occurrence of p occurs in u before the first occurrence of q, and we denote it by $q \Rightarrow_u p$ otherwise;
- p *includes* q if the two occurrences from $\{q, \overline{q}\}$ are found within the p-interval. This relation is denoted by $p \rightarrow_u q$;
- p and q are *disjoint* in u if they do not overlap and neither is included in the other in u.

ld

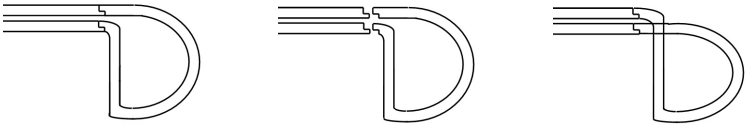


(a) The molecule folds as a loop to align the pointers.

(b) Enzymes cut on the pointer sites.

(c) As a result, one circular molecule gets excised from the original DNA molecule.

hi

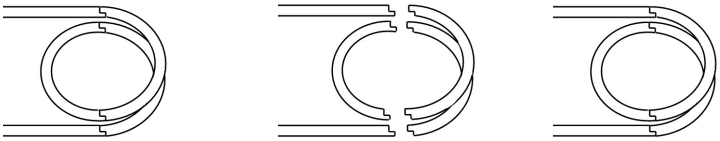


(d) The molecule folds as a hairpin to align the pointers.

(e) Enzymes cut on the pointer sites.

(f) As a result, a segment of the molecule gets inverted.

dlad



(g) The molecule folds as a double loop to align the pointers.

(h) Enzymes cut on the pointer sites.

(i) As a result, two segments of the molecule get inter-changed.

Fig. 1. The three operations of the intramolecular model for gene assembly in ciliates.

A gene can be represented as a legal string through its sequence of pointers and markers, for example, the legal string corresponding to actin I gene in Sterkiella nova is $34456756789m\bar{3}\,\bar{2}m289$, see [8].

The three molecular operations are formulated as rewriting rules on legal strings as follows.

Definition 1 ([5]). *The* string pointer reduction system *is formalized as follows. In each case* $p, q \in \Delta_k$ *are distinct pointers and* $u_1, u_2, u_3 \in \Sigma^{\maltese}$.

1. *The* simple string negative *rule* ssn_p *is defined as follows:*

$$\mathsf{ssn}_p(u_1\tilde{p}\tilde{p}u_2) = u_1u_2,$$

where $\tilde{p} \in \{p, \bar{p}\}$. *We denote* $\mathsf{Ssn} = \{\mathsf{ssn}_p \mid p \geq 2\}$.

2. *The* simple string positive *rule* ssp_p *is defined as follows:*

$$\mathsf{ssp}_p(u_1\tilde{p}\tilde{q}\bar{\tilde{p}}u_3) = u_1\bar{\tilde{q}}u_3,$$

where $\tilde{p} \in \{p, \bar{p}\}$ *and* $\tilde{q} \in \{q, \bar{q}\}$. *We denote* $\mathsf{Ssp} = \{\mathsf{ssp}_p \mid p \geq 2\}$.

3. *The* simple string double *rule* $\mathsf{ssd}_{p,q}$ *is defined as follows:*

$$\mathsf{ssd}_{p,q}(u_1\tilde{p}\tilde{q}u_3\tilde{p}\tilde{q}u_5) = u_1u_3u_5,$$

where $\tilde{p} \in \{p, \bar{p}\}$ *and* $\tilde{q} \in \{q, \bar{q}\}$. *We denote* $\mathsf{Ssd} = \{\mathsf{ssd}_{p,q} \mid p, q \geq 2\}$.

Let $\phi = \phi_r \circ \phi_{r-1} \circ \ldots \circ \phi_1$ *be a composition of rules* $\phi_i \in \mathsf{Ssn} \cup \mathsf{Ssp} \cup \mathsf{Ssd}$, *for all* $1 \leq i \leq r$; *we call any such composition a* reduction strategy. *We say that* ϕ *is* successful *for legal string* u *if* $\phi(u) = mm$ *or* $\phi(u) = \bar{m}\bar{m}$.

Example 1. Let $u = 34456756789m\bar{3}\,\bar{2}m289$ be the legal string corresponding to actin I gene in Sterkiella nova. Then $\mathsf{ssp}_3 \circ \mathsf{ssn}_5 \circ \mathsf{ssn}_4 \circ \mathsf{ssp}_2 \circ \mathsf{ssd}_{8,9} \circ \mathsf{ssd}_{6,7}$ is a reduction strategy for u:

$$\begin{aligned}
u_1 &= \mathsf{ssd}_{6,7}(u) = 3445589m\bar{3}\,\bar{2}m289,\\
u_2 &= \mathsf{ssd}_{8,9}(u_1) = 34455m\bar{3}\,\bar{2}m2,\\
u_3 &= \mathsf{ssp}_2(u_2) = 34455m\bar{3}\,\bar{m},\\
u_4 &= \mathsf{ssn}_4(u_3) = 355m\bar{3}\,\bar{m},\\
u_5 &= \mathsf{ssn}_5(u_4) = 3m\bar{3}\,\bar{m},\\
u_6 &= \mathsf{ssp}_3(u_5) = \bar{m}\bar{m}.
\end{aligned}$$

2.2 Overlap-Inclusion Graphs

Overlap-inclusion graphs (in short, OI graphs) have been introduced in [4]. Using our notation (which is slightly different than that of [4]), for a legal string u its overlap-inclusion graph $G_u = (V, \sigma, E)$ is defined as follows:

- $V = \mathsf{dom}(u)$;
- $\sigma : V \to \{+, -\}$ is the signing of vertices: for each $p \in V$, $\sigma(p) = +$ if p is a positive pointer in u and $\sigma(p) = -$ otherwise;
- $E = \{\{p, q\} \mid p \Rightarrow_u q \text{ or } q \Rightarrow_u p)\} \bigcup \{(p, q) \mid p \to_u q\}$.

In other words, for any pair of overlapping pointers $\{p, q\}$ in u there is an undirected (overlap) edge in G between p and q, and for any pointer q whose interval is included in the interval of some pointer p, G has the directed (inclusion) edge $p \to_G q$.

The simple negative rule and the simple positive rule can be defined on OI graphs so that they correspond to the string pointer reduction system; for details we refer to [4]. The simple double rule could not be defined for such graphs. The rule could however be introduced, see [1], on an extension of these graphs, the DOI graphs, that we discuss in the next section.

Example 2. Let u be the legal string corresponding to actin I gene in Sterkiella nova. Its corresponding OI graph is given in Figure 2(a).

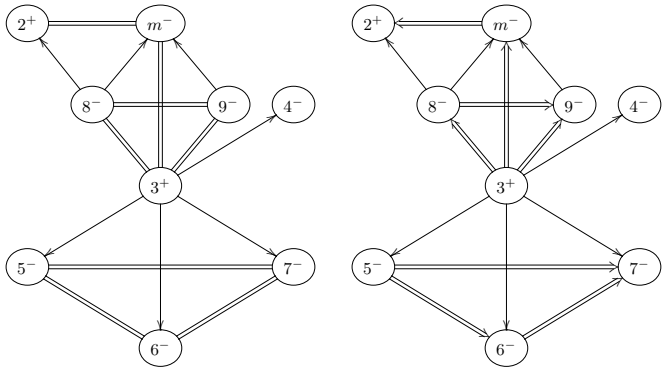


Fig. 2. (a) The OI graph corresponding to actin I gene in Sterkiella nova, (b) the DOI graph corresponding to actin I gene in Sterkiella nova.

2.3 Directed Overlap-Inclusion Graphs

We recall here the notion of *directed overlap-inclusion (DOI) graphs* introduced in [1] and recall some of their basic properties that will be needed in this paper. We first recall that a directed graph G is called *connected* if for any distinct vertices u and v of G, there is either a (directed) path from u to v, or a (directed) path from v to u. We also recall that for a set of vertices U of G we denote by $G \setminus U$ the graph obtained from G by removing all vertices in U and all edges incident to them.

Definition 2 ([1]). *Let u be a legal string over some Σ_k. The* directed overlap-inclusion (DOI) graph $G_u = (V, \sigma, E_o, E_i)$ corresponding to u *is defined as follows:*

- $V = \mathsf{dom}(u)$ *is the set of vertices;*
- $\sigma : V \to \{+, -\}$ *is the signing of vertices such that for each $p \in V$, $\sigma(p) = +$ if p is a positive pointer in u and $\sigma(p) = -$ otherwise;*
- E_o *and* E_i *are sets of its directed edges,* $E_o = \{(p, q) \mid p \Rightarrow_u q\}$ *and* $E_i = \{(p, q) \mid p \to_u q\}$.

For a DOI graph G and any string u such that $G = G_u$ we say that G corresponds to u.

Two DOI graphs G and H have the same *structure*, denoted by $G \equiv H$, if there is a graph isomorphism between G and H.

Example 3. Let u be the legal string corresponding to actin I gene in Sterkiella nova; its corresponding directed overlap-inclusion graph is given in Figure 2(b).

Theorem 1 ([1]). *Any DOI graph G is a directed acyclic graph.*

Definition 3 ([2]). *Let G be a DOI graph and p an arbitrary vertex of G. We introduce the following terms:*

1. Incoming inclusion edges*: we denote by* $\mathsf{inSet}_i(p)$ *the set of all vertices q such that $q \to p$ is an (inclusion) edge in G. Also,* $\mathsf{inDeg}_i(p)$ *is the number of vertices in* $\mathsf{inSet}_i(p)$.
2. Outgoing inclusion edges*: we denote by* $\mathsf{outSet}_i(p)$ *the set of all vertices q such that $p \to q$ is an (inclusion) edge in G. Also,* $\mathsf{outDeg}_i(p)$ *is the number of vertices in* $\mathsf{outSet}_i(p)$.
3. Incoming overlap edges*: we denote by* $\mathsf{inSet}_o(p)$ *the set of all vertices q such that $q \Rightarrow p$ is an (overlap) edge in G. Moreover,* $\mathsf{inDeg}_o(p)$ *is the number of vertices in* $\mathsf{inSet}_o(p)$.
4. Outgoing overlap edges*: we denote by* $\mathsf{outSet}_o(p)$ *the set of all vertices q such that $p \Rightarrow q$ is an (overlap) edge in G. Also,* $\mathsf{outDeg}_o(p)$ *is the number of vertices in* $\mathsf{outSet}_o(p)$.

We recall now the definition of simple operations for DOI graphs introduced in [2]. The operations are defined in general for any directed, vertex- and edge-labeled graph, in particular for DOI graphs.

Definition 4 ([2]). *Let $G = (V, \sigma, E_o, E_i)$ be a directed, vertex- and edge-labeled graph. For any distinct vertices $p, q \in V \setminus \{m\}$, the graph operations* sgn_p, sgp_p *and* $\mathsf{sgd}_{p,q}$ *are defined on G as follows:*

(i) The simple graph negative rule sgn *for p, denoted* sgn_p*, is applicable to G if:*

- $\sigma(p) = -$ *and*
- $\mathsf{inDeg}_o(p) = \mathsf{outDeg}_o(p) = \mathsf{outDeg}_i(p) = 0$.

In this case, $\mathsf{sgn}_p(G) = G \setminus \{p\}$. *We denote* $\mathsf{Sgn} = \{\mathsf{sgn}_p \mid p \in \Delta_k, p \geq 2\}$. *We say that* sgn_p corresponds *to the string-rewriting rule* ssn_p.

(ii) The simple graph positive rule sgp *for p, denoted* sgp_p*, is applicable to G if:*

- $\sigma(p) = +$,
- $\mathsf{inDeg}_o(p) + \mathsf{outDeg}_o(p) = 1$, *and*
- $\mathsf{outDeg}_i(p) = 0$.

Let q be the vertex with the property $\mathsf{inSet}_o(p) \cup \mathsf{outSet}_o(p) = \{q\}$. *In this case,* $\mathsf{sgp}_p(G)$ *is the graph obtained from $G \setminus \{p\}$ by switching the label of q: q is negative in* $\mathsf{sgp}_p(G)$ *if and only if it is positive in G. We denote* $\mathsf{Sgp} = \{\mathsf{sgp}_p \mid p \in \Delta_k, p \geq 2\}$. *We say that* sgp_p corresponds *to the string-rewriting rule* ssp_p.

(iii) The simple graph double rule sgd *for p, q, denoted* $\mathsf{sgd}_{p,q}$*, is applicable to G if:*

- $\sigma(p) = \sigma(q) = -$,
- $q \in \mathsf{outSet}_o(p)$,
- $\mathsf{inSet}_o(p) \cup \{p\} = \mathsf{inSet}_o(q)$,
- $\mathsf{outSet}_o(p) = \mathsf{outSet}_o(q) \cup \{q\}$,

– $\mathsf{inSet}_i(p) = \mathsf{inSet}_i(q)$ *and*
– $\mathsf{outSet}_i(p) = \mathsf{outSet}_i(q)$.

In this case, $\mathsf{sgd}_{p,q}(G) = G \setminus \{p, q\}$. *We denote* $\mathsf{Sgd} = \{\mathsf{sgd}_{p,q} \mid p, q \in \Delta_k, p, q \geq 2, p \neq q\}$. *We say that* $\mathsf{sgd}_{p,q}$ *corresponds to the string-rewriting rule* $\mathsf{ssd}_{p,q}$.

A reduction strategy *is a composition of simple graph rules* $\phi = \phi_{p_n} \circ \ldots \circ \phi_{p_2} \circ \phi_{p_1}$. *We say that* ϕ *is* successful *on* G *if* $\phi(G)$ *is the graph having only vertex* m *where* m *is negative. Let* $\Omega \subseteq \{\mathsf{Sgn}, \mathsf{Sgp}, \mathsf{Sgd}\}$ *be a set of types of graph rules. If all rules in* ϕ *are from the rule sets indicated by* Ω*, then we say that* ϕ *is an* Ω-strategy *and that* ϕ *is* Ω-successful*, resp. We also say that a directed, vertex- and edge-labeled graph is* Ω-reducible *if it has an* Ω*-successful reduction strategy.*

Example 4. Let G be the DOI graph corresponding to actin I gene in Sterkiella nova. A successful reduction strategy of G is presented in Figure 3.

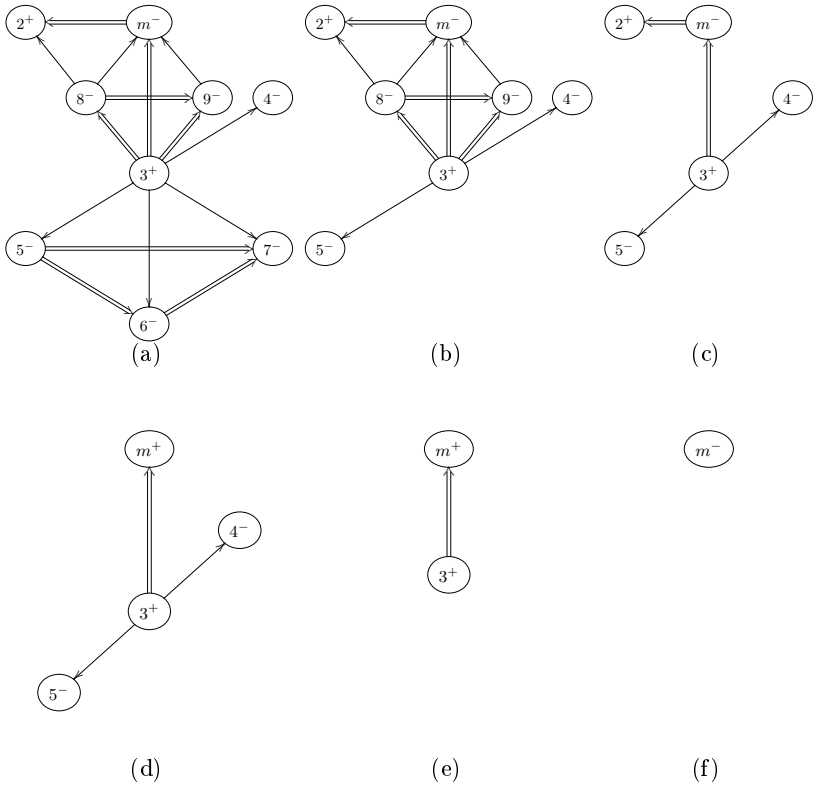


Fig. 3. (a) The DOI graph G corresponding to actin I gene in Sterkiella nova; (b) $G' = \mathsf{sgd}_{6,7}(G)$; (c) $G'' = \mathsf{sgd}_{8,9}(G')$; (d) $G''' = \mathsf{sgp}_2(G'')$; (e) $G^4 = \mathsf{sgn}_4\,\mathsf{sgn}_5(G''')$; (f) $G^5 = \mathsf{sgp}_3(G^4)$.

We recall the equivalence between the string-based model for simple gene assembly and the DOI graph-based model.

Theorem 2 ([2]). *Let u be a legal string, G_u its corresponding DOI graph. Let also $\phi \in \mathsf{Ssn} \cup \mathsf{Ssp} \cup \mathsf{Ssd}$ and $\psi \in \mathsf{Sgn} \cup \mathsf{Sgp} \cup \mathsf{Sgd}$ be the DOI graph rule corresponding to ϕ. Then ϕ is applicable to u if and only if ψ is applicable to G_u. In this case, $G_{\phi(u)} = \psi(G_u)$.*

The following result is straightforward from the definition of our operations and that of the DOI graph equivalence.

Lemma 1. *Let G, H be two DOI graphs such that $G \equiv H$; let $\mu : G \to H$ be the graph isomorphism between them.*

- *If sgn_p is applicable to G, then $\mathsf{sgn}_{\mu(p)}$ is applicable to H and $\mathsf{sgn}_p(G) \equiv \mathsf{sgn}_{\mu(p)}(H)$.*
- *If sgp_p is applicable to G, then $\mathsf{sgp}_{\mu(p)}$ is applicable to H and $\mathsf{sgp}_p(G) \equiv \mathsf{sgp}_{\mu(p)}(H)$.*
- *If $\mathsf{sgd}_{p,q}$ is applicable to G, then $\mathsf{sgd}_{\mu(p),\mu(q)}$ is applicable to H and $\mathsf{sgd}_{p,q}(G) \equiv \mathsf{sgd}_{\mu(p),\mu(q)}(H)$.*

3 The Reduction Power Of The Simple Operations

In this section we focus on characterizing the DOI graphs that are reducible using simple operations of different types.

3.1 {Sgn}-Reducible Graphs

Theorem 3. *Let $G = (V, \sigma, E_o, E_i)$ be a DOI graph. G is Sgn-reducible if and only if $E_o = \emptyset$ and $\sigma(v) = -$, for all $v \in V$.*

Proof. We first prove the reverse implication. Let G be a DOI graph with a single negative node; clearly G is reducible through Sgn operations. Let $n \geq 2$ and assume that all DOI graphs of size $n-1$ with only inclusion edges and negative vertices are reducible using only sgn. Let G be such a DOI graph of size n. By Theorem 1, G is acyclic and so there is a node p such that $\mathsf{inDeg}_i(p) = 0$. Thus, $\mathsf{sgn}_p(G)$ is applicable to G; let $G' = \mathsf{sgn}_p(G) = G \setminus \{p\}$. The conclusion follows by applying the induction hypothesis to G'.

The proof of the direct implication is straightforward. Let G be a DOI graph reducible using only Sgn. By Definition 4, applying an Sgn operation neither removes any overlap edge, nor changes any signing of the vertices. Therefore all edges are inclusion edges and all vertices are negative.

3.2 {Sgn, Sgp}-Reducible Graphs

The next theorem characterizes the reduction power of $\mathsf{Sgn} \cup \mathsf{Sgp}$-operations. The formulation and the proof of the result follows closely the corresponding result of [4] for OI graphs. We first introduce some notations.

For a composition $\phi = \phi_{p_n} \circ \ldots \circ \phi_{p_2} \circ \phi_{p_1}$ of $\mathsf{Sgn} \cup \mathsf{Sgp}$-operations ϕ_{p_i}, $1 \leq i \leq n$, we denote by $\mathsf{dom}_-(\phi) = \{p \in \Delta_k \mid \exists 1 \leq i \leq n$ such that $\phi_{p_i} = \mathsf{sgn}_p\}$. We also denote $\mathsf{ord}(\phi) = (p_1, p_2, \ldots, p_n)$ the order in which the vertices are reduced by ϕ.

For a directed graph G we say that an ordering $P = (p_1, p_2, \ldots, p_n)$ of its vertices is *anti-topological* if there is no edge from p_i to p_j for all $i < j$. In particular, in the case of a DOI graph, an anti-topological ordering of its vertices takes into account both its inclusion and its overlap edges.

For a DOI graph $G = (V, \sigma, E_o, E_i)$, we define $G_o = (V, \sigma, E'_o)$ to be the *undirected* subgraph of G induced by its overlap edges: $E'_o = \{\{i, j\} \mid (i, j) \in E_o$ or $(j, i) \in E_o\}$. For a vertex p, $\deg_{G_o}(p)$ denotes its degree in the (undirected) graph G_o.

Theorem 4. *Let $G = (V, \sigma_G, E_o, E_i)$ be a DOI graph. Let $N \subseteq V \setminus \{m\}$ and $P = (p_1, p_2, \ldots, p_n, p_{n+1})$ a linear ordering of V with $p_{n+1} = m$. There is an $\{\mathsf{Sgn}, \mathsf{Sgp}\}$-successful reduction ϕ of G with $N = \mathsf{dom}_-(\phi)$ and $P = (\mathsf{ord}(\phi), m)$ if and only if the following conditions are satisfied:*

1. *G_o is a forest;*
2. *For each vertex $q \in G$, $\deg_{G_o}(q)$ is even if and only if $\sigma_G(q) = -$;*
3. *Each tree in the forest G_o has exactly one vertex in $N \cup \{m\}$;*
4. *Consider G_o as a rooted forest with its roots in $N \cup \{m\}$ and denote by G_N the graph obtained from G by changing the orientation of all its edges $s \Rightarrow t$ where s is a child of t in the rooted forest G_o. Then G_N is acyclic and P is an anti-topological ordering of G_N.*

Proof. We prove the result by showing the equivalence of both sides of the statement with the following set of conditions:

a. For all $1 \leq i < j \leq n+1$, there is no inclusion edge $p_i \to p_j$ in G;
b. For all $1 \leq i \leq n$, if $p_i \in N$, then there is no overlap edge between p_i and p_j in G, in either direction, for any $1 \leq i < j \leq n$. If $p_i \notin N$, then there is exactly one $j > i$ such that there exists an overlap edge between p_i and p_j in G.
c. For each vertex $q \in G$, $\deg_{G_o}(q)$ is even if and only if $\sigma_G(q) = -$.

Claim. The left-hand side of the theorem's statement holds if and only if conditions a-c are satisfied.

Proof of Claim: Let $\phi = \phi_{p_n} \circ \ldots \circ \phi_{p_1}$ be an $\{\mathsf{Sgn}, \mathsf{Sgp}\}$-successful strategy, $N = \mathsf{dom}_-(\phi)$ and $P = (\mathsf{ord}(\phi), m) = (p_1, p_2, \ldots, p_n, p_{n+1})$, with $p_{n+1} = m$. Denote $G_i = \phi_{p_i} \circ \ldots \circ \phi_{p_1}(G)$, for all $1 \leq i \leq n$.

Let $1 \leq i < j \leq n+1$, i.e., p_i is reduced before p_j either by an sgn or an sgp operation. By Definition 4 we can conclude that there is no outgoing inclusion edge from p_i to p_j in G_{i-1}. Since G_{i-1} is obtained from G by successive node removals and possible node label switches, it follows that the same is true in G, thus proving condition **a**.

Let $1 \leq i \leq n$. If $p_i \in N = \mathsf{dom}_-(\phi)$, then by the definition of sgn operation it is easy to see that p_i has no overlap edge incident to any vertex p_j with $j > i$. If $p_i \notin N$, then p_i is reduced in ϕ through sgp_{p_i}. Therefore, by Definition 4, there is only one vertex p_j adjacent to p_i in G_{i-1} with an overlap edge where $i < j$. The same clearly holds also in G, thus proving condition **b**.

Condition **c** can easily be obtained through induction on the number of vertices of G.

To prove the reverse implication we use induction on the number of vertices of graph G.

If $n = 0$, then $p_1 = m$ and $\deg_{G_o}(p_1) = 0$. By (c) we have $\sigma(p_1) = -$ and so, G is trivially reducible through the empty reduction strategy. Assume now that the claim holds for all graphs with at most k vertices, for some $k \geq 1$. Let G be a DOI graph with $k+1$ vertices, that satisfies conditions **a-c**.

If $p_1 \in N$, then by **a-b** it is either isolated or it has some incoming inclusion edges; therefore $\deg_{G_o}(p_1) = 0$, $\sigma(p_1) = -$ and so, sgn_{p_1} is applicable to G. Consequently, based on the definition of sgn, $G_1 = \mathsf{sgn}_{p_1}(G)$ is a DOI graph with k vertices that satisfies conditions **a-c**. By the induction hypothesis G_1 is $\{\mathsf{Sgn}, \mathsf{Sgp}\}$-reducible and thus, so is G.

If $p_1 \notin N$, by condition **a** p_1 has no outgoing inclusion edges and by **b**, there is exactly one $j > 1$ such that there is an overlap edge between p_1 and p_j, and by **c**, $\sigma(p_1) = +$. Hence, p_1 is reducible using sgp. Applying sgp does not add any edges and it only changes the signing of vertex p_j. Consequently, $G_1 = \mathsf{sgp}_{p_1}(G)$ is a DOI graph with k vertices that satisfies conditions **a-c**. By induction hypothesis, G_1 is $\{\mathsf{Sgn}, \mathsf{Sgp}\}$-reducible and thus, so is G.

Claim. The right-hand side of the theorem's statement holds if and only if conditions **a-c** are satisfied.

Proof of Claim: We note immediately that condition **c** is identical to 2.

We first prove the direct implication. By 4 P is an anti-topological ordering of G_N. Therefore for every $i < j$, there is no edge from p_i to p_j in G_N. Since the inclusion edges in G_N are the inclusion edges in G, it follows that for all $i < j$ there is no inclusion edge $p_i \to p_j$ in G, proving **a**.

Let $1 \leq i \leq n$. If $p_i \in N$, then p_i is the root of a rooted tree in the forest G_o. By 4, it follows that if there is any overlap edge incident to p_i in G_N, then it is an outgoing overlap edge from p_i. Therefore, for every $j > i$ if there is an overlap edge between p_i and p_j in G, then G_N has an edge from p_i to p_j, which contradicts P being an anti-topological ordering of G_N. Hence, for $p_i \in N$, there is no overlap edge between p_i and p_j for any $j > i$. This proves the first part of **b**.

If $p_i \notin N$, then p_i is not the root of a rooted tree in G_o and so, there exists a directed edge $p_j \Rightarrow p_i$ in G_N. Since P is an anti-topological ordering on the vertices of G_N, it follows that $i < j$. Assume now that there are $j_1 > j_2 > i$ such that there is an overlap edge between p_{j_1} to p_i and an overlap edge between p_{j_2} and p_i in G. This implies that $p_{j_1} \Rightarrow_{G_N} p_i$ and $p_{j_2} \Rightarrow_{G_N} p_i$ are edges of G_N, based on the anti-topological ordering of its vertices. Observe now that p_{j_1} and p_{j_2} are vertices in the same rooted tree of G_N and so, there is a directed path ρ_1 from its root, say p_r, to p_{j_1} and a directed path ρ_2 from p_r to p_{j_2}. Note however that ρ_1, ρ_2 and the edges $p_{j_1} \Rightarrow_{G_N} p_i$, $p_{j_2} \Rightarrow_{G_N} p_i$ induce a cycle in G_o, contradicting i. This concludes the proof of the second part of **b**.

To prove the reverse implication, assume that G_o has a cycle and let p_i be the vertex with the smallest index on that cycle. This implies that there are two vertices $j_1, j_2 > i$ such that there is an edge between p_i and p_{j_1} and an edge between p_i and p_{j_2}. This contradicts **b**, thus proving 1.

To prove 3, let T be a tree in G_o and let p_i be the vertex in T with the largest index i. By **b**, if $p_i \notin N \cup \{m\}$, there is an overlap edge between p_i and a vertex p_j where $j > i$, a contradiction with our choice of i.

Assume now there are two vertices of T $p_i, p_j \in N \cup \{m\}$, where $i < j$. There is an overlap path, say ρ, in T from p_i to p_j: $\rho = (p_{k_1}, \ldots, p_{k_r})$, where $r \geq 2$, $k_1 = i$ and $k_r = j$. By **b**, p_i has no overlap edge incident to any vertex with a larger index and so, $k_2 < k_1$. It also follows by **b** that p_{k_2} can have at most one overlap edge incident to a vertex with a larger index, which is p_{k_1}; therefore, $k_3 < k_2$. Iterating this argument we conclude that $j = k_r < \ldots < k_2 < k_1 = i$, contradicting that $i < j$.

To prove 4, let p_i, p_j be two vertices in G where $i < j$ such that there is an overlap edge $p_i \Rightarrow_{G_N} p_j$. Take i to be the smallest such index. Then by **b**, $p_i \notin N$. By the construction of G_N, it follows that there is another vertex p_k such that $p_k \Rightarrow_{G_N} p_i$. If $k > i$, then p_i is overlap-adjacent to two vertices with index larger than i, which contradicts **b**. Thus, $k < i$; this however contradicts our choice of i. Hence P is an anti-topological ordering of G_N. It then follows that G_N is also acyclic.

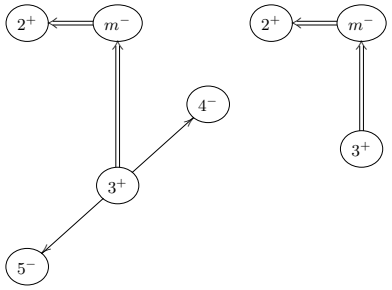
3.3 {Sgp}-Reducible Graphs

Theorem 5. *Let $G = (V, \sigma_G, E_o, E_i)$ be a DOI graph with $m \in V$. G is* {Sgp}*-reducible if and only if the following conditions are satisfied:*

1. *G_o is a tree;*
2. *For each vertex $q \in G$, $\deg_{G_o}(q)$ is even if and only if $\sigma_G(q) = -$;*
3. *Let G_N be the graph obtained from G by changing the orientation of all its directed edges $s \Rightarrow t$ where s is a child of t in the rooted forest G_o. Then G_N is acyclic and each successful reduction in G corresponds to an anti-topological ordering of G_N.*

Proof. Note that Theorem 4 implies an ordering in which the graph is reduced and the vertices in N are the last ones to be reduced in every connected component of the graph. Therefore $N = \emptyset$ in the case where a DOI graph is reduced using only Sgp. Then the claim follows by Theorem 4.

Example 5. Let G be the DOI graph corresponding to actin I gene in Sterkiella nova given in Figure 2(b). Since G_o is not a forest, it follows by Theorem 4 that G is not $\{\mathsf{Sgn}, \mathsf{Sgp}\}$-reducible. Let G' be the induced subgraph of G given in Figure 4(a); let $N = \{4, 5\}$ and $P = (2, 5, 4, 3)$. It follows by Theorem 4 that G' is $\{\mathsf{Sgn}, \mathsf{Sgp}\}$-reducible with the successful strategy $\phi = \mathsf{sgp}_3\, \mathsf{sgn}_4\, \mathsf{sgn}_5\, \mathsf{sgp}_2(G')$. Note that G' is not $\{\mathsf{Sgp}\}$-reducible, since G'_o is not a tree and therefore, G' does



(a) (b)

Fig. 4. (a) An $\{\mathsf{Sgn}, \mathsf{Sgp}\}$-reducible subgraph of the DOI graph corresponding to actin I gene in Sterkiella nova. (b) An $\{\mathsf{Sgp}\}$-reducible subgraph of the DOI graph corresponding to actin I gene in Sterkiella nova.

not satisfy the conditions in Theorem 5. On the other hand, but the subgraph G'' given in Figure 4(b) is $\{\mathsf{Sgp}\}$-reducible.

The following problems concerning the reduction power of the simple operations remain open:

1. characterize $\{\mathsf{Sgd}\}$-reducible DOI graphs;
2. characterize $\{\mathsf{Sgn}, \mathsf{Sgd}\}$-reducible DOI graphs;
3. characterize $\{\mathsf{Sgp}, \mathsf{Sgd}\}$-reducible DOI graphs;
4. characterize $\{\mathsf{Sgn}, \mathsf{Sgp}, \mathsf{Sgd}\}$-reducible DOI graphs.

4 Confluent Strategies On DOI Graphs

It has been shown in [12] and [5] that the strategies using simple operations are confluent for signed permutations and legal strings. In this section we show that this result holds also for DOI graphs. The results here are similar to those of [12] and [5] in the case of permutations and strings.

Definition 5. *Let $G = (V, \sigma_G, E_o, E_i)$ be a DOI graph. We say that the reduction strategy ϕ for G is* maximal *if either ϕ is successful for G, or no operation is applicable to $\phi(G)$. We say that two reduction strategies ϕ, ψ for G are* confluent *if $\phi(G) \equiv \psi(G)$.*

Lemma 2. *Let G be a DOI graph and $\phi \in \mathsf{Sgn}$, $\psi \in \mathsf{Sgn} \cup \mathsf{Sgp} \cup \mathsf{Sgd}$ be two operations applicable to G. Then both $\phi \circ \psi$ and $\psi \circ \phi$ to G are applicable and $\phi \circ \psi(G) = \psi \circ \phi(G)$.*

Proof. The result follows easily since applying sgn on one vertex neither has any effect on the signing of the other vertices nor adds any edges to influence the applicability of any other operations.

Lemma 3. *Let G be a DOI graph and $\phi, \psi \in \mathsf{Sgp}$ be two distinct operations applicable to G. Then either $\phi \circ \psi(G) = \psi \circ \phi(G)$ or $\phi(G) \equiv \psi(G)$.*

Proof. Let $\phi = \mathsf{sgp}_p$ and $\psi = \mathsf{sgp}_q$, $p \neq q$. We consider the following two cases:

Case 1 If vertices p and q do not overlap, then clearly, by definition, $\phi \circ \psi(G) = \psi \circ \phi(G)$.

Case 2 If vertices p and q overlap, say $p \Rightarrow q$, then the only other edges incident to p and q are incoming inclusion edges. It follows then that $\mathsf{sgn}_q \circ \mathsf{sgp}_p(G) = \mathsf{sgn}_p \circ \mathsf{sgp}_q(G)$, i.e., $\phi(G) \equiv \psi(G)$.

Lemma 4. *Let G be a DOI graph and $\phi, \psi \in \mathsf{Sgd}$ be two distinct operations applicable to G. Then either $\phi \circ \psi(G) = \psi \circ \phi(G)$ or $\phi(G) \equiv \psi(G)$.*

Proof. Let $\phi = \mathsf{sgd}_{p,q}$ and $\psi = \mathsf{sgd}_{r,s}$. If $\{p, q\} \neq \{r, s\}$, clearly $\phi \circ \psi(G) = \psi \circ \phi(G)$. Consider now the case where $|\{p, q\} \cap \{r, s\}| = 1$. Note that if $p = r$ or $q = s$, then $\mathsf{sgd}_{p,q}$ and $\mathsf{sgd}_{r,s}$ are not applicable simultaneously to G. The followng two cases are then possible:

Case 1 $q = r$, $p \Rightarrow q$ and $q \Rightarrow s$,
Case 2 $p = s$, $p \Rightarrow q$ and $r \Rightarrow p$.

It is enough to discuss here only Case 1, as the other is symmetric. Denote $G' = \mathsf{sgd}_{p,q}(G)$ and $G'' = \mathsf{sgd}_{q,s}(G)$. Then $\mathsf{outSet}_o(p) \setminus \{q\} = \mathsf{outSet}_o(q)$ and $\mathsf{outSet}_o(q) \setminus \{s\} = \mathsf{outSet}_o(s)$. As a result $\mathsf{outSet}_o(s)$ in G' is equal to $\mathsf{outSet}_o(p)$ in G''. Similar results hold true for the sets of outgoing inclusion edges and incoming overlap and inclusion edges. Hence, $\phi(G) \equiv \psi(G)$.

Lemma 5. *Let G be a DOI graph and $\phi \in \mathsf{Sgp}$, $\psi \in \mathsf{Sgd}$ be two operations applicable to G. Then $\phi \circ \psi(G) = \psi \circ \phi(G)$.*

Proof. Let $\phi = \mathsf{sgp}_p$ and $\psi = \mathsf{sgp}_{q,r}$. If $p \notin \{q, r\}$, then clearly, by definition, $\phi \circ \psi(G) = \psi \circ \phi(G)$.

Let $p \in \{q, r\}$. Since sgp_p is applicable to G, p is a positive vertex in G. But $\mathsf{sgd}_{q,r}$ is also applicable to G and so, both q and r should be negative in G, a contradiction.

Theorem 6. *Let G be a DOI graph and ϕ, ψ be two maximal strategies for G. Then ϕ and ψ are confluent.*

Proof. We prove by induction on the number of vertices of G.

The case when G has only one vertex is trivial. Suppose now the claim holds for all DOI graphs with $|V| \leq k$.

Let G be a DOI graph with $|V| = k+1$ and $\phi = \phi_m \circ \ldots \circ \phi_2 \circ \phi_1$, $\psi = \psi_n \circ \ldots \circ \psi_2 \circ \psi_1$. If $\phi_1 = \psi_1$, the the claim follows by the induction hypothesis. Assume that they are distinct, but both applicable to G. By Lemmas 2-5, either $\phi_1 \circ \psi_1(G) = \psi_1 \circ \phi_1(G)$ or $\phi_1(G) \equiv \psi_1(G)$. The latter case is concluded based on Lemma 1.

If $\phi_1 \circ \psi_1(G) = \psi_1 \circ \phi_1(G)$, then by induction hypothesis all maximal strategies on $\phi_1(G)$ are confluent; consequently, all maximal strategies for $\psi_1 \circ \phi_1(G)$ are also confluent. Similarly, all strategies on $\psi_1(G)$ are confluent; consequently, all maximal strategies on $\phi_1 \circ \psi_1(G)$ are also confluent. Since $\phi_1 \circ \psi_1(G) = \psi_1 \circ \phi_1(G)$, it follows that all maximal strategies on $\phi_1(G)$ are confluent with all maximal strategies on $\psi_1(G)$, concluding the proof.

Corollary 1. *Let G be a DOI graph. Then either all its maximal strategies are successful, or they are all unsuccessful and in this case, the resulting graphs have the same structure.*

References

1. Azimi, S., Harju, T., Langille, M., Petre, I., Rogojin, V.: Directed overlap-inclusion graphs as representations of ciliate genes. Fundamenta Informaticae 110(1), 29–44 (2011)
2. Azimi, S., Harju, T., Langille, M., Petre, I.: Simple gene assembly as a rewriting of directed overlap-inclusion graphs. Theoretical Computer Science 454, 30–37 (2012)
3. Brijder, R., Harju, T., Jonoska, N., Petre, I., Rozenberg, G.: Gene assembly in ciliates. In: Rozenberg, G., Bäck, T., Kok, J. (eds.) Handbook of Natural Computing, vol. 2 (Molecular Computation). Springer (2012)
4. Brijder, R., Hoogeboom, H.: Combining overlap and containment for gene assembly in ciliates. Theoretical Computer Science 411(6), 897–905 (2010)
5. Brijder, R., Langille, M., Petre, I.: A string-based model for simple gene assembly. In: Csuhaj-Varju, E., Esik, Z. (eds.) Proceedings of FCT 2007. Lecture Notes in Computer Science, vol. 4639, pp. 161–172. Springer (2007)
6. Brijder, R., Langille, M., Petre, I.: Extended strings and graphs for simple gene assembly. Theoretical Computer Science 411(4-5), 730–738 (2010)
7. Cavalcanti, A., Clarke, T., Landweber, L.: Mds_ies_db: a database of macronuclear and micronuclear genes in spirotrichous ciliates. Nucleic acids research 33(1), D396 (2005)
8. Ehrenfeucht, A., Harju, T., Petre, I., Prescott, D., Rozenberg, G.: Computation in living cells: gene assembly in ciliates. Springer (2004)
9. Ehrenfeucht, A., Prescott, D., Rozenberg, G.: Computational aspects of gene (un) scrambling in ciliates. In: Landweber, L., Winfree, E. (eds.) Evolution as computation, pp. 216–256. Springer (2001)
10. Harju, T., Petre, I., Rogojin, V., Rozenberg, G.: Patterns of simple gene assembly in ciliates. Discrete Applied Mathematics 156(14), 2581–2597 (2008)
11. Harju, T., Rozenberg, G.: Computational processes in living cells: gene assembly in ciliates. Lecture Notes in Computer Science 2450, 1–20 (2003)

12. Langille, M., Petre, I.: Simple gene assembly is deterministic. Fundamenta Informaticae 73(1), 179–190 (2006)
13. Petre, I., Rozenberg, G.: Gene assembly in ciliates. Scholarpedia 5(1), 9269 (2010)
14. Prescott, D., Ehrenfeucht, A., Rozenberg, G.: Molecular operations for dna processing in hypotrichous ciliates. European Journal of Protistology 37(3), 241–260 (2001)

Paper IV

Sepinoud Azimi, Bogdan Iancu, Ion Petre, Reaction system models for the heat shock response, Fundamenta Informaticae, IOS Press, 131, 1–14, 2014.

Fundamenta Informaticae 131 (2014) 299–312
DOI 10.3233/FI-2014-1016
IOS Press

Reaction System Models for the Heat Shock Response

Sepinoud Azimi*, Bogdan Iancu, Ion Petre
Computational Biomodeling Laboratory
Turku Centre for Computer Science
Åbo Akademi University
20520 Turku, Finland
{*Sepinoud.Azimi, Bogdan.Iancu, Ion.Petre*}*@abo.fi*

Abstract. Reaction systems are a formal framework for modeling processes driven by biochemical reactions. They are based on the mechanisms of facilitation and inhibition. A main assumption is that if a resource is available, then it is present in sufficient amounts and as such, several reactions using the same resource will not compete concurrently against each other; this makes reaction systems very different as a modeling framework than traditional frameworks such as ODEs or continuous time Markov chains. We demonstrate in this paper that reaction systems are rich enough to capture the essential characteristics of ODE-based models. We construct a reaction system model for the heat shock response in such a way that its qualitative behavior correlates well with the quantitative behavior of the corresponding ODE model. We construct our reaction system model based on a novel concept of dominance graph that captures the competition on resources in the ODE model. We conclude with a discussion on the expressivity of reaction systems as compared to that of ODE-based models.

Keywords: Reaction systems; heat shock response; quantitative model; qualitative model; model comparison.

1. Introduction

Reaction systems (RS in short) are a formal framework for modeling processes driven by biochemical reactions. They were introduced in [2], see also [1] and references therein. The fundamental idea in this

*Address for correspondence: Department of Information Technologies, Åbo Akademi University, Joukahaisenkatu 3-5, 20520 Turku, Finland

framework is that biochemical reactions are based on the mechanisms of *facilitation* and *inhibition*. A reaction is modeled as a triplet: a set of reactants, a set of inhibitors, and a set of products. A reaction can take place in a given state if all its reactants are present in that state and none of its inhibitors; when triggered, the reaction creates its products. Two major assumptions in reaction systems set them apart from standard methods for biomodelling (such as ordinary differential equations, stochastic processes, Petri nets, and process algebras):

- *The threshold assumption:* if a resource is present, then it is present in a "sufficient amount" and it will not cause any conflict between several reactions needing that resource. In other words, several reactions needing the same reactant will not be in conflict.
- *No permanency assumption:* an entity will vanish from the current state unless it is produced by one of the reactions enabled in that state.

The goal of this paper is to demonstrate that reaction systems are capable of capturing the essential characteristics of complex ODE models. We construct an RS model for the molecular heat shock response introduced in [6]. Our focus is on building the model in such a way that a number of properties of the ODE-based model of [6] for the heat shock response are preserved: mass-conservation, steady state configuration with and without stress, behavior under continuous stress. The challenge here is that these properties are essentially numerical, correlating well to numerical experimental data and knowledge, whereas the RS framework is qualitative, as shown by the threshold assumption. Moreover, special attention has to be given to overcoming the no permanency assumption to make sure that, e.g., a gene is not removed from the system, even when no gene activity has occurred. We first take the straightforward approach of building the RS model through translating the reactants/products from the molecular model to an RS with the same reactants/products and no inhibitors. It turns out however that the resulting RS model leads to a behavior that is very different than that of the ODE-based model. We show however that an RS model can be built in a different way, qualitatively replicating the numerical behavior of the ODE model.

The paper is structured as follows. In Section 2 we introduce some basics of modeling with ODEs and some basic notions of reaction systems. In Section 3 we introduce our molecular model for the heat shock response and discuss some of the numerical properties of its corresponding mass-action-based ODE model. In Section 4 we build a direct translation of the molecular model to an RS model with no inhibitors and look at some of its interactive processes. In Section 5 we build a different RS model whose interactive processes correlate well with the numerical behavior of the ODE model. We conclude with some discussion in Section 6.

2. Preliminaries

2.1. Reaction-based molecular models and their ODE-based representation

Biochemical networks can be represented as reaction-based molecular models. Such models consist of sets of coupled chemical reactions describing the system of interest; the reactions can be reversible or irreversible. Formally, the fundamental constituents of a model M are represented by a set of species $\Sigma = \{S_i \mid 1 \leq i \leq m\}$ and a set of reactions $\{R_j \mid 1 \leq j \leq n\}$, where n and m are nonnegative integers.

An irreversible reaction R_j, $1 \leq j \leq n$, is formalized as a rewriting rule as follows:

$$R_j \ : \ c_{1,j}S_1 + c_{2,j}S_2 + \ldots + c_{m,j}S_m \xrightarrow{k_j} c'_{1,j}S_1 + c'_{2,j}S_2 + \ldots + c'_{m,j}S_m.$$

A reversible reaction can be written in the following form:

$$R_j \ : \ c_{1,j}S_1 + c_{2,j}S_2 + \ldots + c_{m,j}S_m \underset{k_j^-}{\overset{k_j^+}{\rightleftarrows}} c'_{1,j}S_1 + c'_{2,j}S_2 + \ldots + c'_{m,j}S_m.$$

The nonnegative integers k_j and k_j^-, k_j^+ represent the *kinetic rate constants* of an irreversible, respectively reversible reaction R_j. The coefficients $c_{1,j}, \ldots, c_{m,j}, c'_{1,j}, \ldots, c'_{m,j}$ are positive integers characterizing the stoichiometry of the reaction. The stoichiometric coefficient of species S_i in reaction R_j is defined by: $n_{i,j} = c'_{i,j} - c_{i,j}$.

The reactant species, found on the left-hand side of the reaction, are called *substrates*, while the species produced, occurring on the right-hand side, are referred to as *products*. A species S_i with $c_{i,j} = 0$ ($c'_{i,j} = 0$, resp.) can be omitted from the left-hand (right-hand, resp.) side of reaction R_j corresponding to the null coefficient. A reversible reaction can always be regarded as a pair of two irreversible reactions, see [3].

The *molecularity* of a reaction R_j is defined by the following sum: $\sum_{i=1}^{m} c_{i,j}$. One generally considers systems comprising reactions of a molecularity of at most two. Reactions exhibiting a molecularity of three are very infrequent due to the high improbability of simultaneous collisions between three molecules leading to the formation of a complex. Reactions with a molecularity greater than three are completely disregarded due to the impossibility of collision between more than three molecules synchronously, see [5].

A molecular model M can be represented as a mathematical model in various manners, following a continuous or discrete time evolution, based on continuous or discrete species variables. A common representation is based on ordinary differential equations (ODEs) and the principle of mass-action, see [4]. To each species S_i, $1 \leq i \leq m$, one associates a time-dependent function: $[S_i] : \mathbb{R}_+ \to \mathbb{R}_+$, representing the concentration of the species over time. Therefore, the evolution of the system is described by a system of differential equations of the following form:

$$\frac{d[S_i]}{dt} = -\sum_{j=1}^{n} \left(k_j^+ c_{i,j} \prod_{l=1}^{m} [S_l]^{c_{l,j}}(t) \right) + \sum_{j=1}^{n} \left(k_j^- c'_{i,j} \prod_{l=1}^{m} [S_l]^{c'_{l,j}}(t) \right), \ \ 1 \leq i \leq m.$$

Such systems of ODEs can be rarely solved analytically, but many numerical methods for analyzing them exist. In particular, a numerical integration of the ODE system is interpreted as a numerical simulation of the corresponding molecular model.

2.2. Reaction systems

Reaction systems have been introduced in [2] as a formal framework for the analysis of biochemical networks. A biochemical reaction in this framework is based on a finite set of reactants and it is triggered provided that all the reactants involved in that particular reaction are *present* in a given state and all of its inhibitors are *absent*, see [8].

We recall in this section the basic definitions we need throughout the paper. For more details we refer to [2, 8].

Definition 2.1. [2] A *reaction* is a triplet of non-empty, finite sets: $a = (R_a, I_a, P_a)$, where $R_a \cap I_a = \emptyset$. The sets R_a, I_a, P_a stand for the set of *reactants*, *inhibitors*, *products* of a, respectively. Given a set S, if R_a, I_a, $P_a \subseteq S$, then a is a reaction in S. The set of reactions in S is denoted by *rac*(S).

Definition 2.2. [2] Let A be a set of reactions, T a finite set, and $a \in A$.

(i) The *result* of a on T, denoted $res_a(T)$, is

$$res_a(T) = \begin{cases} P_a, & \text{if } R_a \subseteq T \text{ and } I_a \cap T = \emptyset \\ \emptyset, & \text{otherwise.} \end{cases}$$

(ii) The *result of A on T*, denoted $res_A(T)$, is

$$res_A(T) = \bigcup_{a \in A} res_a(T).$$

Definition 2.3. [2] A *reaction system* (RS in short) is defined as an ordered pair $\mathcal{A} = (S, A)$, where S is a finite set and $A \subseteq$ *rac*(S). The set S is called the *background* (set) of A.

Definition 2.4. [2] Let $\mathcal{A}$ be a reaction system. An *interactive process* in $\mathcal{A}$ is a pair $\pi = (\gamma, \delta)$, where $\gamma = C_0, C_1, ..., C_n, \delta = D_1, D_2, ..., D_n \subseteq S$, $n \geq 1$, with $D_1 =$ *res*$_{\mathcal{A}}(C_0)$ and, for each $1 < i \leq n$, $D_i =$ *res*$_{\mathcal{A}}(C_{i-1} \cup D_{i-1})$.

The sequences γ and δ are the *context sequence* of π, *con*(π), and the *result sequence* of π, *res*(π), resp. The *state sequence* of π is $\tau = W_0, W_1, ..., W_n$, where $W_i = C_i \cup D_i$, for all $i \in \{0, ..., n\}$ and $W_0 = C_0$. W_0 is the *initial state* of π, init(π), and W_n is the *final state* of π, fst(π).

Definition 2.5. Let $\mathcal{A} = (S, A)$ be a reaction system and $C \subseteq S$. We say that $D \subseteq S$ is a *steady state* of $\mathcal{A}$ for C if *res*$_{\mathcal{A}}(C \cup D) = D$.

3. A molecular model for the heat shock response

The heat shock response in eukaryotes is a fundamental, well conserved defense mechanism, which allows the cell to react to environmental stressors such as elevated temperatures. The increase of temperature in the environment causes proteins in the cell to misfold and build up large conglomerates that ultimately result in cell death. The key elements for the heat shock response mechanism are the heat shock proteins (hsp), which operate as molecular chaperones for misfolded proteins (mfp), facilitating their refolding process. The response is regulated by the transactivation of the hsp-encoding genes. Gene transcription is mediated by heat shock factors (hsf), which, under stress, dimerize (hsf_2), subsequently trimerize (hsf_3) and then bind to a promoter-site of the hsp-encoding gene, the heat shock-element (hse). As soon as trimers are bound, protein synthesis is activated and new hsp molecules are produced. When the level of hsp is sufficiently uplifted, hsp synthesis is turned off, see [6, 7]. Heat shock proteins sequestrate hsf molecules, hence promoting DNA binding. When the temperature is elevated, proteins in the

Table 1. The molecular model for the eukaryotic heat shock response proposed in [6].

Reaction	Reaction
$2\,\mathsf{hsf} \leftrightarrows \mathsf{hsf}_2$	$\mathsf{hsp} + \mathsf{hsf}_3 \to \mathsf{hsp{:}\,hsf} + 2\,\mathsf{hsf}$
$\mathsf{hsf} + \mathsf{hsf}_2 \leftrightarrows \mathsf{hsf}_3$	$\mathsf{hsp} + \mathsf{hsf}_3{:}\,\mathsf{hse} \to \mathsf{hsp{:}\,hsf} + 2\,\mathsf{hsf} + \mathsf{hse}$
$\mathsf{hsf}_3 + \mathsf{hse} \leftrightarrows \mathsf{hsf}_3{:}\,\mathsf{hse}$	$\mathsf{hsp} \to \emptyset$
$\mathsf{hsf}_3{:}\,\mathsf{hse} \to \mathsf{hsf}_3{:}\,\mathsf{hse} + \mathsf{hsp}$	$\mathsf{prot} \to \mathsf{mfp}$
$\mathsf{hsp} + \mathsf{hsf} \leftrightarrows \mathsf{hsp{:}\,hsf}$	$\mathsf{hsp} + \mathsf{mfp} \leftrightarrows \mathsf{hsp{:}\,mfp}$
$\mathsf{hsp} + \mathsf{hsf}_2 \to \mathsf{hsp{:}\,hsf} + \mathsf{hsf}$	$\mathsf{hsp{:}\,mfp} \to \mathsf{hsp} + \mathsf{prot}$

cell (prot) tend to misfold, causing hsp: hsf complexes to break up. The heat shock response is switched on again, facilitating hsp synthesis, see [6]. We list in Table 1 the reactions of the molecular model in [6].

A mathematical model is associated with the molecular model in Table 1. The mathematical model consists in a mass-action-based system of ODEs, see [4] for a brief discussion on the principle of mass-action. We refer to [6] for the system of ODEs and the numerical setup of the ODE model.

For a constant temperature of $37°C$ the model reaches a steady state where most hsf's are bound in a complex hsp: hsf, most hse's are available for binding to trimers and there are very few misfolded proteins. For a constant temperature of $42°C$ the model reaches a steady state different from the one at $37°C$ in that the level of misfolded proteins, in both forms mfp and hsp: mfp are relatively high. Upon removal of the stress and return to $37°C$, the model returns to the basal values attained under a constant temperature of $37°C$. For a more detailed discussion about the steady states of the model and numerical simulations we refer to [6].

4. From the molecular model to a reaction system model: a direct translation

In this section, we formulate a reaction system model for the heat shock response, based on the mechanisms of facilitation and inhibition. We use a direct translation approach, i.e. we translate each reaction in the molecular model to a reaction in the corresponding reaction system. We disregard in the current model the reaction $\mathsf{hsp} \to \emptyset$ in Table 1 due to its very low reaction rate in the ODE model; similarly, we ignore reactions $\mathsf{hsf}_3{:}\,\mathsf{hse} \to \mathsf{hsf}_3 + \mathsf{hse}$ and $\mathsf{hsp{:}\,mfp} \to \mathsf{hsp} + \mathsf{mfp}$ (both the reverse directions of some irreversible reactions in Table 1). We also disregard the dimer form hsf_2 of hsf; indeed, the dimer is only a transient state from hsf to hsf_3 and their levels remain insignificant regardless of the stress. The simplified molecular model for the heat shock response is in Table 2.

Table 2. The simplified molecular model for the eukaryotic heat shock response.

Reaction		Reaction	
$3\,\mathsf{hsf} \leftrightarrows \mathsf{hsf}_3$	(1)	$\mathsf{hsp} + \mathsf{hsf}_3{:}\,\mathsf{hse} \to \mathsf{hsp{:}\,hsf} + 2\,\mathsf{hsf} + \mathsf{hse}$	(6)
$\mathsf{hsf}_3 + \mathsf{hse} \to \mathsf{hsf}_3{:}\,\mathsf{hse}$	(2)	$\mathsf{prot} \to \mathsf{mfp}$	(7)
$\mathsf{hsf}_3{:}\,\mathsf{hse} \to \mathsf{hsf}_3{:}\,\mathsf{hse} + \mathsf{hsp}$	(3)	$\mathsf{hsp} + \mathsf{mfp} \to \mathsf{hsp{:}\,mfp}$	(8)
$\mathsf{hsp} + \mathsf{hsf} \leftrightarrows \mathsf{hsp{:}\,hsf}$	(4)	$\mathsf{hsp{:}\,mfp} \to \mathsf{hsp} + \mathsf{prot}$	(9)
$\mathsf{hsp} + \mathsf{hsf}_3 \to \mathsf{hsp{:}\,hsf} + 2\,\mathsf{hsf}$	(5)		

4.1. The first reaction system model

We describe first a simple method for translating a set of molecular reactions to a reaction system. A unary molecular reaction $A \rightarrow C$ is translated into the RS reaction $\{\{A\}, \{\mathsf{d_I}\}, \{C\}\}$, where $\mathsf{d_I}$ is a "dummy" variable. A binary molecular reaction $A + B \rightarrow C$ is translated into the RS reaction $\{\{A, B\}, \{\mathsf{d_I}\}, \{C\}\}$. The dummy variable is only used here to comply with the constraint that the set of inhibitors of all RS reactions should be non-empty; none of the molecular reactions specify any explicit inhibitor. The ternary reaction (1) is treated as a unary reaction $\mathsf{hsf} \leftrightarrow \mathsf{hsf}_3$. The case of reversible molecular reactions is handled analogously, provided that we first replace it with two irreversible molecular reactions, standing for the two directions of the original reaction, and then define their correspondents in a reaction system as above. The full RS model obtained as a result is given in Table 3.

Table 3. The direct translation of the biochemical reactions of the simplified model of the heat shock response to a reaction system.

Reaction in the chemical network	Reaction in the reaction system	
$3\,\mathsf{hsf} \leftrightarrow \mathsf{hsf}_3$	$(\{\mathsf{hsf}\}, \{\mathsf{d_I}\}, \{\mathsf{hsf}_3\})$	(i)
	$(\{\mathsf{hsf}_3\}, \{\mathsf{d_I}\}, \{\mathsf{hsf}\})$	(ii)
$\mathsf{hsf}_3 + \mathsf{hse} \rightarrow \mathsf{hsf}_3\colon \mathsf{hse}$	$(\{\mathsf{hsf}_3, \mathsf{hse}\}, \{\mathsf{d_I}\}, \{\mathsf{hsf}_3\colon \mathsf{hse}\})$	(iii)
$\mathsf{hsf}_3\colon \mathsf{hse} \rightarrow \mathsf{hsf}_3\colon \mathsf{hse} + \mathsf{hsp}$	$(\{\mathsf{hsf}_3\colon \mathsf{hse}\}, \{\mathsf{d_I}\}, \{\mathsf{hsf}_3\colon \mathsf{hse}, \mathsf{hsp}\})$	(iv)
$\mathsf{hsp} + \mathsf{hsf} \leftrightarrow \mathsf{hsp}\colon \mathsf{hsf}$	$(\{\mathsf{hsp}, \mathsf{hsf}\}, \{\mathsf{d_I}\}, \{\mathsf{hsp}\colon \mathsf{hsf}\})$	(v)
	$(\{\mathsf{hsp}\colon \mathsf{hsf}\}, \{\mathsf{d_I}\}, \{\mathsf{hsp}, \mathsf{hsf}\})$	(vi)
$\mathsf{hsp} + \mathsf{hsf}_3 \rightarrow \mathsf{hsp}\colon \mathsf{hsf} + 2\,\mathsf{hsf}$	$(\{\mathsf{hsp}, \mathsf{hsf}_3\}, \{\mathsf{d_I}\}, \{\mathsf{hsp}\colon \mathsf{hsf}, \mathsf{hsf}\})$	(vii)
$\mathsf{hsp} + \mathsf{hsf}_3\colon \mathsf{hse} \rightarrow \mathsf{hsp}\colon \mathsf{hsf} + \mathsf{hse} + 2\,\mathsf{hsf}$	$(\{\mathsf{hsp}, \mathsf{hsf}_3\colon \mathsf{hse}\}, \{\mathsf{d_I}\}, \{\mathsf{hsp}\colon \mathsf{hsf}, \mathsf{hsf}, \mathsf{hse}\})$	(viii)
$\mathsf{prot} \rightarrow \mathsf{mfp}$	$(\{\mathsf{prot}\}, \{\mathsf{d_I}\}, \{\mathsf{mfp}\})$	(ix)
$\mathsf{hsp} + \mathsf{mfp} \rightarrow \mathsf{hsp}\colon \mathsf{mfp}$	$(\{\mathsf{hsp}, \mathsf{mfp}\}, \{\mathsf{d_I}\}, \{\mathsf{hsp}\colon \mathsf{mfp}\})$	(x)
$\mathsf{hsp}\colon \mathsf{mfp} \rightarrow \mathsf{hsp} + \mathsf{prot}$	$(\{\mathsf{hsp}\colon \mathsf{mfp}\}, \{\mathsf{d_I}\}, \{\mathsf{hsp}, \mathsf{prot}\})$	(xi)

4.2. Interactive processes in the first model

We analyze in this subsection the dynamics of the reaction system in terms of interactive processes and compare it with that of the corresponding ODE model.

Note that in the ODE model the temperature was taken into account through the temperature dependent protein misfolding rate constant. Since our construction only translates the reactions and not their quantitative values, the effect of the temperature on the model is lost in the translation. Consequently, the RS model can only at best capture some of the behaviour of the model in the absence of stress; we show below that the model in fact even fails to do that.

In our first interactive process we start from an initial state consisting of a minimal set of species needed in the heat shock response model: hsf, hse, and prot; all other species and complexes can be

obtained in the molecular model (as well as in the ODE model) starting from these main ingredients. Thus, let the initial context of our reaction system be $C_0 = \{\mathsf{hsf}, \mathsf{prot}, \mathsf{hse}\}$. Throughout this interactive process all subsequent contexts are empty: $C_i = \emptyset$, for all $i \geq 1$. This interactive process is represented in Table 4 and it shows that for every $k \geq 1$, $D_{2k} = \{\mathsf{hsf}\}$ and $D_{2k+1} = \{\mathsf{hsf}_3\}$. The prediction of the RS model is thus that the model will enter into a loop of length two when starting from $\{\mathsf{hsf}, \mathsf{prot}, \mathsf{hse}\}$, for an empty context. This is in contradiction with the prediction of the ODE model, which shows the system converging to a steady state at $37°C$.

Table 4. An interactive process for the direct translation of the simplified model of the heat shock reponse for the first setting.

State	C_i	D_i	W_i	r_i
0	$\{\mathsf{hsf}, \mathsf{prot}, \mathsf{hse}\}$	$\emptyset$	$\{\mathsf{hsf}, \mathsf{prot}, \mathsf{hse}\}$	{(i), (ix)}
1	$\emptyset$	$\{\mathsf{hsf}_3, \mathsf{mfp}\}$	$\{\mathsf{hsf}_3, \mathsf{mfp}\}$	{ (ii) }
2	$\emptyset$	$\{\mathsf{hsf}\}$	$\{\mathsf{hsf}\}$	{ (i) }
3	$\emptyset$	$\{\mathsf{hsf}_3\}$	$\{\mathsf{hsf}_3\}$	{ (ii) }
4	$\emptyset$	$\{\mathsf{hsf}\}$	$\{\mathsf{hsf}\}$	{ (i) }

For our second interactive process, the initial state consists of all species included in the $37°C$ steady-state of the ODE model: $C_0 = \{\mathsf{hse}, \mathsf{hsp}\colon \mathsf{hsf}, \mathsf{prot}\}$. The interactive process starting from this state and using an empty context is represented in Table 5. Thus, the prediction of the RS model is that $D_{2k-1} = D_5$ and $D_{2k} = D_6$, for all $k \geq 3$. This again shows a contradiction with the ODE model. Indeed, starting from an initial state corresponding to the $37°C$ steady state of the ODE model, the RS model eventually enters into a loop of length 2.

Table 5. An interactive process for the direct translation of the simplified model of the heat shock response for the second setting.

State	C_i	D_i	W_i	r_i
0	$\{\mathsf{hse}, \mathsf{hsp}\colon \mathsf{hsf}, \mathsf{prot}\}$	$\emptyset$	$\{\mathsf{hse}, \mathsf{hsp}\colon \mathsf{hsf}, \mathsf{prot}\}$	(vi) , (ix)
1	$\emptyset$	$\{\mathsf{hsp}, \mathsf{hsf}, \mathsf{mfp}\}$	$\{\mathsf{hsp}, \mathsf{hsf}, \mathsf{mfp}\}$	(i), (v), (x)
2	$\emptyset$	$\{\mathsf{hsf}_3, \mathsf{hsp}\colon \mathsf{hsf}, \mathsf{hsp}\colon \mathsf{mfp}\}$	$\{\mathsf{hsf}_3, \mathsf{hsp}\colon \mathsf{hsf}, \mathsf{hsp}\colon \mathsf{mfp}\}$	(ii), (vi), (xi)
3	$\emptyset$	$\{\mathsf{hsp}, \mathsf{hsf}, \mathsf{prot}\}$	$\{\mathsf{hsf}, \mathsf{hsp}\}$	(i), (v), (ix)
4	$\emptyset$	$\{\mathsf{hsf}_3, \mathsf{hsp}\colon \mathsf{hsf}, \mathsf{mfp}\}$	$\{\mathsf{hsf}_3, \mathsf{hsp}\colon \mathsf{hsf}, \mathsf{mfp}\}$	(ii), (vi)
5	$\emptyset$	$\{\mathsf{hsf}, \mathsf{hsp}\}$	$\{\mathsf{hsf}, \mathsf{hsp}\}$	(i), (v)
6	$\emptyset$	$\{\mathsf{hsf}_3, \mathsf{hsp}\colon \mathsf{hsf}\}$	$\{\mathsf{hsf}_3, \mathsf{hsp}\colon \mathsf{hsf}\}$	(ii), (vi)
7	$\emptyset$	$\{\mathsf{hsp}, \mathsf{hsf}\}$	$\{\mathsf{hsp}, \mathsf{hsf}\}$	(i), (v)

5. A second reaction system model for heat shock response

We introduce in this section a second RS model for the heat shock response. Our strategy this time is completely different than in the last section: we will formulate a number of essential properties of the heat shock response model and build the RS model to satisfy them. These properties will be achieved in the RS model *only through the mechanism of inhibition*, whereas they are emerging in the ODE model through a 'game of numbers', i.e., through the numerical values of the kinetic rate constants.

We introduce two new resources, nostress and stress, to model the system in the absence and the presence of the heat shock, resp., mirroring the behavior of the ODE model for temperature values of $37°C$ and $42°C$, resp.

We build the model so that the following properties hold in any state W of the reaction system $\mathcal{A}$, where either $\mathsf{stress} \in W$, or $\mathsf{nostress} \in W$, but not both:

P1. *mass-conservation of* hse: if $\{\mathsf{hse}, \mathsf{hsf_3}\colon \mathsf{hse}\} \cap W \neq \emptyset$, then $\{\mathsf{hse}, \mathsf{hsf_3}\colon \mathsf{hse}\} \cap \mathit{res}_{\mathcal{A}}(W) \neq \emptyset$;

P2. *a single form of* hse: if $\{\mathsf{hse}, \mathsf{hsf_3}\colon \mathsf{hse}\} \not\subseteq W$, then $\{\mathsf{hse}, \mathsf{hsf_3}\colon \mathsf{hse}\} \not\subseteq \mathit{res}_{\mathcal{A}}(W)$;

P3. *mass-conservation of* prot: if $\mathsf{prot} \in W$, then $\mathsf{prot} \in \mathit{res}_{\mathcal{A}}(W)$;

P4. *misfolded proteins must be addressed*: if $\mathsf{mfp} \in W$, then $\{\mathsf{mfp}, \mathsf{hsp}\colon \mathsf{mfp}\} \cap \mathit{res}_{\mathcal{A}}(W) \neq \emptyset$;

P5. *a single form of* hsf: let $\mathsf{HSF} = \{\mathsf{hsf}, \mathsf{hsf_3}, \mathsf{hsf_3}\colon \mathsf{hse}, \mathsf{hsp}\colon \mathsf{hsf}\}$; if $|W \cap \mathsf{HSF}| \leq 1$, then $|\mathit{res}_{\mathcal{A}}(W) \cap \mathsf{HSF}| \leq 1$;

P6. *stability of* hsp: hsf *in the absence of stress*: if $\{\mathsf{nostress}, \mathsf{hsp}\colon \mathsf{hsf}\} \subseteq W$, then $\mathsf{hsp}\colon \mathsf{hsf} \in \mathit{res}_{\mathcal{A}}(W)$.

Our main challenge is in building an RS model that captures qualitatively a behavior driven by numerical competition on resources, *in the absence* of an explicit mechanism for concurrency. Since our model consists of only unary and binary reactions, our main observation is that we can capture the competition between two binary molecular reactions using resources $\{A, B_1\}$ and $\{A, B_2\}$, resp., in terms of a preference (*binding affinity*) of A over, say, B_1, rather than B_2. We can formulate these relationships in the form of a *dominance graph*, where the graph nodes represent the molecular reactions (we indicate by '+' the left-to-right direction of a reversible reaction and by '-' the reverse direction) and a directed edge $u \to v$ indicates that u, v compete on a common resource and u is favoured over v.

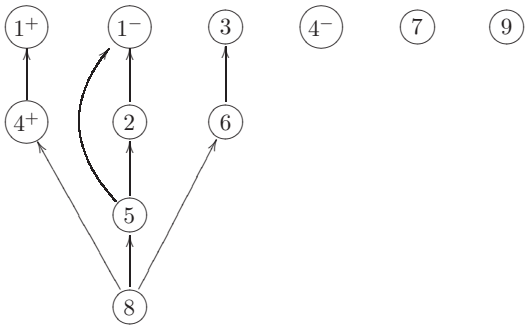


Figure 1. The reaction dominance graph of the simplified heat shock response model. The node labels refer to the molecular reactions in Table 2. We indicate with a directed edge $u \to v$ the property that u is favoured over v.

We build the dominance graph in Figure 1 for the molecular model in Table 2 based on the following assumptions:

- hsf has a higher affinity for hsp than for another hsf (edge $4^+ \to 1^+$);
- hsf_3 has a higher affinity to bind to hsp, than to break into hsf monomers or to interact with hse (edges $5 \to 1^-$ and $5 \to 2$);
- hsf_3 has a higher affinity to bind to hse, than to break into hsf monomers (edge $2 \to 1^-$);
- hsf_3: hse has a higher affinity to interact with hsp than to promote gene transcription (edge $6 \to 3$);
- hsp has a higher affinity for mfp than for hsf, hsf_3, hsf_3: hse (edges $8 \to 4^+$, $8 \to 5$, $8 \to 6$).

All these assumptions correspond to numerical observations regarding the reaction rates of the ODE model in [6].

5.1. Building the second model

We discuss now the construction of the RS model going through the reactions of the simplified molecular model in Table 2 one by one. The corresponding RS reactions are in Table 6. In the following we extend the terminology of 'enabled reactions' to the molecular model in Table 2; we will say that a molecular reaction is enabled in the current state W of our RS system, if all its reactants are in W.

Molecular reaction (1^+) is first modelled through RS reaction (10), where hsp is indicated as an inhibitor, as suggested by the edge $4^+ \to 1^+$. Note however based on Figure 1 that reaction (1^+) is enabled also when (8) is enabled, since in this case (4^+) is disabled. In other words, (1^+) is enabled in the presence of hsp, mfp, with no inhibitor. This leads to formulating RS reaction (11).

Molecular reaction (1^-) is first modelled through RS reaction (12), where hse, hsp are indicated as inhibitors, as suggested by edges $5 \to 1^-$ and $2 \to 1^-$. Similarly, as above, we note based on Figure 1 that if (8) is enabled, i.e., if hsp and mfp are in the current state, then (5) is disabled and so, in this case only reaction (2) supersedes (1^-). This leads to formulating RS reaction (13).

Molecular reaction (2) is modelled through RS reactions (14) and (15). The reasoning in this case is similar to that corresponding to (1^+). Additionally however, we need to introduce several other RS reactions to make sure that, in case (14) and (15) are disabled in the current state, hse is not lost, as required by property **P1**. In other words, this is analogous to building the RS correspondent of a molecular reaction hse $\to$ hse that in the dominance graph would have an incoming edge from node 2. With a similar reasoning as above, this leads to adding RS reaction (16): hse is preserved unless hsf_3 is present, when (2) is enabled. Moreover, based on the graph in Figure 1, we note that if (5) is enabled, i.e., if hsf_3 and hsp were in the current state, but not mfp, then (2) is disabled, i.e., the molecular reaction hse $\to$ hse would be enabled. This leads to introducing RS reaction (17).

Molecular reaction (3) is modelled through RS reactions (18) and (19). The reasoning in this case is similar to that corresponding to (1^+).

Modeling molecular reactions (4^+), (5) and (6) are all similar. They are modelled through RS reactions (20), (23), and (24), resp. We only note that when writing (23) we took into account **P5** and excluded hsf from the result of the RS reaction; this is in slight disagreement with the molecular reaction (5) where hsf is a product of the reaction. Applying **P5** we decided to keep in the set of products only one form of hsf and we favoured hsp: hsf over hsf, since it is the more common form of hsf in the ODE model.

To model the molecular reaction (4^-) we took into account **P6** and formulated it through RS reactions (21) and (22), depending on whether stress or nostress are in the current state.

Molecular reaction (7) is modelled through RS reactions (25) and (26), where we also took into account **P3**.

Molecular reaction (8) is modelled through RS reaction (27) and (28), where we also took into account **P4**.

Finally, molecular reaction (9) is modelled in a straightforward way through RS reaction (29). The complete list of reactions of the RS model for heat shock response is given in Table 6.

Table 6. The list of reactions of the second reaction system model for heat shock response.

Reaction		Reaction	
$(\{\mathsf{hsf}\}, \{\mathsf{hsp}\}, \{\mathsf{hsf}_3\})$	(10)	$(\{\mathsf{hsp}, \mathsf{hsf}\}, \{\mathsf{mfp}\}, \{\mathsf{hsp}\colon \mathsf{hsf}\})$	(20)
$(\{\mathsf{hsf}, \mathsf{hsp}, \mathsf{mfp}\}, \{\mathsf{d_I}\}, \{\mathsf{hsf}_3\})$	(11)	$(\{\mathsf{hsp}\colon \mathsf{hsf}, \mathsf{stress}\}, \{\mathsf{nostress}\}, \{\mathsf{hsp}, \mathsf{hsf}\})$	(21)
$(\{\mathsf{hsf}_3\}, \{\mathsf{hse}, \mathsf{hsp}\}, \{\mathsf{hsf}\})$	(12)	$(\{\mathsf{hsp}\colon \mathsf{hsf}, \mathsf{nostress}\}, \{\mathsf{stress}\}, \{\mathsf{hsp}\colon \mathsf{hsf}\})$	(22)
$(\{\mathsf{hsf}_3, \mathsf{hsp}, \mathsf{mfp}\}, \{\mathsf{hse}\}, \{\mathsf{hsf}\})$	(13)	$(\{\mathsf{hsp}, \mathsf{hsf}_3\}, \{\mathsf{mfp}\}, \{\mathsf{hsp}\colon \mathsf{hsf}\})$	(23)
$(\{\mathsf{hsf}_3, \mathsf{hse}\}, \{\mathsf{hsp}\}, \{\mathsf{hsf}_3\colon \mathsf{hse}\})$	(14)	$(\{\mathsf{hsp}, \mathsf{hsf}_3\colon \mathsf{hse}\}, \{\mathsf{mfp}\}, \{\mathsf{hsp}\colon \mathsf{hsf}, \mathsf{hse}\})$	(24)
$(\{\mathsf{hsf}_3, \mathsf{hse}, \mathsf{hsp}, \mathsf{mfp}\}, \{\mathsf{d_I}\}, \{\mathsf{hsf}_3\colon \mathsf{hse}\})$	(15)	$(\{\mathsf{prot}, \mathsf{stress}\}, \{\mathsf{nostress}\}, \{\mathsf{prot}, \mathsf{mfp}\})$	(25)
$(\{\mathsf{hse}\}, \{\mathsf{hsf}_3\}, \{\mathsf{hse}\})$	(16)	$(\{\mathsf{prot}, \mathsf{nostress}\}, \{\mathsf{stress}\}, \{\mathsf{prot}\})$	(26)
$(\{\mathsf{hse}, \mathsf{hsf}_3, \mathsf{hsp}\}, \{\mathsf{mfp}\}, \{\mathsf{hse}\})$	(17)	$(\{\mathsf{hsp}, \mathsf{mfp}\}, \{\mathsf{d_I}\}, \{\mathsf{hsp}\colon \mathsf{mfp}\})$	(27)
$(\{\mathsf{hsf}_3\colon \mathsf{hse}\}, \{\mathsf{hsp}\}, \{\mathsf{hsf}_3\colon \mathsf{hse}, \mathsf{hsp}\})$	(18)	$(\{\mathsf{mfp}\}, \{\mathsf{hsp}\}, \{\mathsf{mfp}\})$	(28)
$(\{\mathsf{hsf}_3\colon \mathsf{hse}, \mathsf{hsp}, \mathsf{mfp}\}, \{\mathsf{d_I}\}, \{\mathsf{hsf}_3\colon \mathsf{hse}, \mathsf{hsp}\})$	(19)	$(\{\mathsf{hsp}\colon \mathsf{mfp}\}, \{\mathsf{d_I}\}, \{\mathsf{hsp}, \mathsf{prot}\})$	(29)

Our following result shows that the resulting model satisfies properties **P1-P6**.

Theorem 5.1. The reaction system in Table 6 satisfies properties **P1-P6**.

Proof:
To prove **P1**, note that if $\mathsf{hse} \in W$, then either (14), (15), (16), or (17) are enabled, all leading to a state containing hse or $\mathsf{hsf}_3\colon \mathsf{hse}$. If $\mathsf{hsf}_3\colon \mathsf{hse} \in W$, then the same argument holds, noting that either (18), or (19), or (24) are enabled.

To prove **P2** we first observe that there is no reaction with both hse and $\mathsf{hsf}_3\colon \mathsf{hse}$ in its list of products. To prove that the model satisfies the property it is enough to show that no two reactions, one having hse, the other having $\mathsf{hsf}_3\colon \mathsf{hse}$ in its list of products, are enabled simultaneously. The reactions having hse in their list of products are (16), (17) and (24); those having $\mathsf{hsf}_3\colon \mathsf{hse}$ in their list of products are (14), (15), (18), and (19). Reaction (16) cannot be enabled simultaneously with either (14) or (15) because hsf_3 is an inhibitor for (16) and it is a reactant for the others. Also, if (16) were enabled simultaneously with (18) or (19), then $\mathsf{hse}, \mathsf{hsf}_3\colon \mathsf{hse} \in W$, a contradiction with the hypothesis of **P2**. Similar arguments show that (17) and (24) cannot be enabled simultaneously with (14), (15), (18), and (19).

To prove **P3** note that the only reactions involving prot are (25) and (26), that exactly one of them is triggered if stress $\in W$ or nostress $\in W$ but not both, and that both have prot in their list of products.

To prove **P4**, it is enough to observe that if $\mathsf{mfp}, \mathsf{hsp} \in W$, then (27) is enabled, while if $\mathsf{mfp} \in W$ and $\mathsf{hsp} \notin W$, then (28) is enabled.

We prove now **P5**. First, it is easy to see that if $W \cap \mathsf{HSF} = \emptyset$, then $res_{\mathcal{A}}(W) \cap \mathsf{HSF} = \emptyset$. Second, note that there is no reaction with more than one element of HSF in its set of products. If $\mathsf{hsf} \in W$, then the reactions that may be enabled are (10), (11), (20); no two of them can be enabled simultaneously. If $\mathsf{hsf}_3 \in W$, then the reactions that may be enabled are (12), (13), (14), (15), (17), and (23); of these, only (17) and (23) can be enabled simultaneously, but the products of (17) do not include any element from HSF. If $\mathsf{hsf}_3\colon \mathsf{hse} \in W$, then the reactions that may be enabled are (18), (19), and (24); no two of them can be enabled simultaneously. If $\mathsf{hsp}\colon \mathsf{hsf} \in W$, then the reactions that may be enabled are (21) and (22), which cannot be enabled simultaneously.

Property **P6** follows from reaction (22). □

5.2. Interactive processes in the second model

We analyze here the interactive processes of the second RS model of the heat shock response and compare it with the results previously attained in the ODE model of [6]. Taking into account the qualitative nature of our model, by the presence of a resource in the environment, except for stress and nostress, conforming to the *threshold assumption*, we assume that there is a sufficient amount of the respective resource in the environment.

Similarly as in the case of our first reaction system model, we start our first interactive process from an initial state consisting of a minimal set of species needed in the heat shock response model: hsf, hse, and prot. To draw a parallel with the numerical simulations of the ODE model at $37°C$, the subsequent contexts of our reaction system consists of the resource nostress. The result is shown in Table 7.

Table 7. An interactive process of the second RS model for heat shock response, in the absence of stress. The first column of the table represents the state of the system, C_i is the context given to the system in state i, $D_i = res_{\mathcal{A}}(C_{i-1} \cup D_{i-1})$ and $W_i = C_i \cup D_i$. The last column provides the list of the reactions triggered in each state.

State	C_i	D_i	W_i	r_i
0	{hsf, prot, hse, nostress}	$\emptyset$	{hsf, prot, hse, nostress}	(10), (16), (26)
1	{nostress}	{hsf_3, prot, hse}	{hsf_3, prot, hse, nostress}	(14), (26)
2	{nostress}	{hsf_3: hse, prot}	{hsf_3: hse, prot, nostress}	(18), (26)
3	{nostress}	{hsp, hsf_3: hse, prot}	{hsp, hsf_3: hse, prot, nostress}	(24), (26)
4	{nostress}	{hsp: hsf, hse, prot}	{hsp: hsf, hse, prot, nostress}	(16), (22), (26)
5	{nostress}	{hsp: hsf, hse, prot}	{hsp: hsf, hse, prot, nostress}	(16), (22), (26)

The result shows that the reaction system in this case enters into a steady state, that is similar to the steady state of the ODE model in the absence of stress.

Our next interactive process follows the behavior of the RS model when the context introduces stress in every state, corresponding to the situation when the temperature is set to $42°C$ in the ODE model. As

the initial state we take the steady state achieved in the previous interactive process: $\{\mathsf{hse}, \mathsf{prot}, \mathsf{hsp}\colon \mathsf{hsf}\}$. The result is shown in Table 8. We note that also in this case the system is reaching a steady state, similar to the steady state of the ODE model for a temperature of $42°C$.

Table 8. An interactive process of the second reaction system model for heat shock response at $42°C$.

State	C_i	D_i	W_i	r_i
0	$\{\mathsf{hse}, \mathsf{prot}, \mathsf{hsp}\colon \mathsf{hsf}, \mathsf{stress}\}$	$\emptyset$	$\{\mathsf{hse}, \mathsf{prot}, \mathsf{hsp}\colon \mathsf{hsf}, \mathsf{stress}\}$	(16),(21),(25)
1	$\{\mathsf{stress}\}$	$\{\mathsf{hse}, \mathsf{hsp}, \mathsf{hsf}, \mathsf{prot}, \mathsf{mfp}\}$	$\{\mathsf{hse}, \mathsf{hsp}, \mathsf{hsf}, \mathsf{prot}, \mathsf{mfp}, \mathsf{stress}\}$	(11),(16),(25),(27)
2	$\{\mathsf{stress}\}$	$\{\mathsf{prot}, \mathsf{mfp}, \mathsf{hsp}\colon \mathsf{mfp}, \mathsf{hsf}_3, \mathsf{hse}\}$	$\{\mathsf{prot}, \mathsf{mfp}, \mathsf{hsp}\colon \mathsf{mfp}, \mathsf{hsf}_3, \mathsf{hse}, \mathsf{stress}\}$	(14),(25),(28),(29)
3	$\{\mathsf{stress}\}$	$\{\mathsf{hsp}, \mathsf{prot}, \mathsf{mfp}, \mathsf{hsf}_3\colon \mathsf{hse}\}$	$\{\mathsf{hsp}, \mathsf{prot}, \mathsf{mfp}, \mathsf{hsf}_3\colon \mathsf{hse}, \mathsf{stress}\}$	(19),(25),(27)
4	$\{\mathsf{stress}\}$	$\{\mathsf{hsp}, \mathsf{prot}, \mathsf{mfp}, \mathsf{hsf}_3\colon \mathsf{hse}, \mathsf{hsp}\colon \mathsf{mfp}\}$	$\{\mathsf{hsp}, \mathsf{prot}, \mathsf{mfp}, \mathsf{hsf}_3\colon \mathsf{hse}, \mathsf{hsp}\colon \mathsf{mfp}, \mathsf{stress}\}$	(19),(25),(27),(29)
5	$\{\mathsf{stress}\}$	$\{\mathsf{hsp}, \mathsf{prot}, \mathsf{mfp}, \mathsf{hsf}_3\colon \mathsf{hse}, \mathsf{hsp}\colon \mathsf{mfp}\}$	$\{\mathsf{hsp}, \mathsf{prot}, \mathsf{mfp}, \mathsf{hsf}_3\colon \mathsf{hse}, \mathsf{hsp}\colon \mathsf{mfp}, \mathsf{stress}\}$	(19),(25),(27),(29)

In our final interactive process we start from the steady state achieved in the previous one and consider the case when the context consists in all subsequent state of only nostress. This corresponds to a case when the ODE model is stabilized at $42°C$, followed then by a temperature of $37°C$. The result is shown in Table 9. The model reaches again a steady state, the same as that reached in the first interactive process. The situation is similar to that of the ODE model, where upon removal of the stress, the model eventually returns to its basal physiological values.

Table 9. Interactive process for the recovery (at $37°C$) of the second reaction system model after several steps of heat shock (at $42°C$). The process starts from the steady state obtained in Table 8.

State	C_i	D_i	W_i	r_i
0	$\{\mathsf{hsp}, \mathsf{prot}, \mathsf{hsf}_3\colon \mathsf{hse}, \mathsf{mfp}, \mathsf{hsp}\colon \mathsf{mfp}, \mathsf{nostress}\}$	$\emptyset$	$\{\mathsf{hsp}, \mathsf{prot}, \mathsf{hsf}_3\colon \mathsf{hse}, \mathsf{mfp}, \mathsf{hsp}\colon \mathsf{mfp}, \mathsf{nostress}\}$	(19),(26),(27), (29)
1	$\{\mathsf{nostress}\}$	$\{\mathsf{hsp}, \mathsf{prot}, \mathsf{hsp}\colon \mathsf{mfp}, \mathsf{hsf}_3\colon \mathsf{hse}\}$	$\{\mathsf{hsp}, \mathsf{prot}, \mathsf{hsp}\colon \mathsf{mfp}, \mathsf{hsf}_3\colon \mathsf{hse}, \mathsf{nostress}\}$	(24),(26),(29)
2	$\{\mathsf{nostress}\}$	$\{\mathsf{hse}, \mathsf{hsp}, \mathsf{hsp}\colon \mathsf{hsf}, \mathsf{prot}\}$	$\{\mathsf{hse}, \mathsf{hsp}, \mathsf{hsp}\colon \mathsf{hsf}, \mathsf{prot}, \mathsf{nostress}\}$	(16),(22),(26)
3	$\{\mathsf{nostress}\}$	$\{\mathsf{hse}, \mathsf{prot}, \mathsf{hsp}\colon \mathsf{hsf}\}$	$\{\mathsf{hse}, \mathsf{prot}, \mathsf{hsp}\colon \mathsf{hsf}, \mathsf{nostress}\}$	(16),(22),(26)
4	$\{\mathsf{nostress}\}$	$\{\mathsf{hse}, \mathsf{prot}, \mathsf{hsp}\colon \mathsf{hsf}\}$	$\{\mathsf{hse}, \mathsf{prot}, \mathsf{hsp}\colon \mathsf{hsf}, \mathsf{nostress}\}$	(16),(22),(26)

6. Discussion

We demonstrated in this paper the ability of the reaction system framework to capture the intricate behaviour of ODE-based models. Given the qualitative nature of reaction systems and their unusual two major assumptions, the non-permanency and the threshold assumptions, the result bridges two fundamentally different modelling paradigms. As a case-study we chose the heat shock response. We focused on emulating the behavior of the corresponding mass-action ODE model for the heat shock response. Whereas the ODE model is driven by the numerical values of its kinetic constants, the reaction system framework only allows the specification of logical dependencies in terms of facilitators and inhibitors. Moreover, its two fundamental principles, the non-permanency of resources and the uncompetitive triggering of reactions, make the reaction system framework fundamentally different than that of the ODE-based modelling.

Our first approach, where each molecular reaction would be directly translated to a reaction system with no inhibitors failed to reproduce the biological knowledge about the heat shock response and the behavior of the ODE model. The *non-permanency principle* of the reaction systems has a major role in the behavior of the reaction system being different than that of the ODE model. For example, the heat shock element hse is lost in an interactive process in any state where reaction (iii) is not triggered; this is in disagreement with the common intuition that a gene promoter is not lost when there is no gene binding activity; it is also in disagreement with the ODE model, where the total amount of heat shock element, either in the form of hse , or as $\mathsf{hsf}_3\colon\mathsf{hse}$, is conserved. The solution here, that we implement in our second reaction system model, is to have several rules making sure that hse is involved in at least one reaction regardless of the context, and thus preserved throughout the interactive process. The *threshold assumption* of the reaction systems also plays a major role in the disagreement between the models. This is seen, e.g., in the treatment of proteins in the two models. While the ODE model exhibits a *gradual* misfolding (and refolding) of prot, in the RS model *all* proteins (prot) are converted into mfp's in a single step. Another weakness of the RS model obtained through the direct translation is that it does not take into consideration the main driving factor of the heat shock response: the temperature.

Our second RS model was successful in emulating the behavior of the ODE model. The key observation here was that in building our first RS model we lost the information about all kinetic constants of the model and did not compensate for it in the RS model. In building our second model we developed an approach where we capture the affinity of a species for another species through carefully selected inhibitors rather than kinetic constants as in the ODE model. We also focused on a number of properties, including mass conservation, that the RS model should satisfy. Instrumental here was the notion of dominance graph that aims to translate numerical observations about the reaction rates of the ODE model into a binary relation on the set of RS reactions. The competition between two reactions on the same resource is captured by an ODE-based model through the reaction rates: the higher the rate of a reaction, the more it will consume the competed resource; note however that all reactions, even those with the lowest rate, will get some part of the resource and will be able to run. Going towards the qualitative realm of reaction systems, we translated the competition between two reactions in terms of a directed graph where an edge $i \to j$ represents that reactions i, j compete on a common resource and that in the ODE model, reaction i has a higher rate than reaction j. This is then translated into a reaction system where, through the use of suitable inhibitors, the RS reaction corresponding to j cannot be triggered if the RS reaction corresponding to i is enabled. This then helps to capture cascading effects: if reaction

i dominates reaction j and j dominates k, then having i enabled effectively enables k; this is similar to the behaviour of ODE-based models.

Our approach based on dominance graphs might be possible to generalize to arbitrary molecular models consisting of unary and binary molecular reactions. The main problem here is the algorithmic construction of the dominance graph starting from a molecular model, its kinetic rate constants and the initial values of all variables. How much a-priori knowledge about the behaviour of the model is needed, as well as how well the procedure scales up with the size of the model appear as interesting questions, worthy of further attention. A particularly interesting question is how, if at all possible, can one capture a dynamic situation where the dominance relations between reactions changes in time, as it may well be possible in ODE models.

The reaction system framework forces the modeler to make explicit a number of assumptions about the model, that in other framework are typically hidden in some numerical values. Moreover, the explicit list of inhibitors is shedding light into the causality relations between the various reactions in the system. This kind of insight is highly valuable and may be very difficult to obtain through other frameworks, either from the presentation of the model, or from numerical simulations.

Acknowledgments

We thank Daniela Besozzi for reading through and commenting on the manuscript. We acknowledge the detailed comments from the two referees, that have helped to improve the presentation of the paper.

References

[1] R. Brijder, A. Ehrenfeucht, M. Main, and G. Rozenberg. A tour of reaction systems. *International Journal of Foundations of Computer Science*, 22(07):1499–1517, 2011.

[2] A. Ehrenfeucht and G. Rozenberg. Reaction systems. *Fundamenta Informaticae*, 75(1):263–280, 2007.

[3] F.G. Helfferich. *Kinetics of multistep reactions*, volume 40. Elsevier Science, 2004.

[4] E. Klipp, R. Herwig, A. Kowald, C. Wierling, and H. Lehrach. *Systems biology in practice: concepts, implementation and application*. Wiley-Vch, 2005.

[5] D. L. Nelson and M. M. Cox. Lehninger principles of biochemistry. *Worth Publishers*, 2000.

[6] I. Petre, A. Mizera, C.L. Hyder, A. Meinander, A. Mikhailov, R.I. Morimoto, L. Sistonen, J.E. Eriksson, and R. Back. A simple mass-action model for the eukaryotic heat shock response and its mathematical validation. *Natural Computing*, 10(1):595–612, 2011.

[7] T.R. Rieger, R.I. Morimoto, and V. Hatzimanikatis. Mathematical modeling of the eukaryotic heat-shock response: dynamics of the hsp70 promoter. *Biophysical journal*, 88(3):1646–1658, 2005.

[8] G. Rozenberg, A. Ehrenfeucht, and M. Main. Combinatorics of life and death for reaction systems. *International Journal of Foundations of Computer Science*, 21(3):345–356, 2010.

Paper V

Sepinoud Azimi, Cristian Gratie, Sergiu Ivanov, Luca Manzoni, Ion Petre, Antonio E. Porreca, Complexity of model checking for reaction systems, Submitted, 2015.

Complexity of Model Checking for Reaction Systems

Sepinoud Azimi[a], Cristian Gratie[a], Sergiu Ivanov[c], Luca Manzoni[b], Ion Petre[a], Antonio E. Porreca[b]

[a] *Computational Biomodeling Laboratory, Turku Centre for Computer Science and Department of Computer Science, Åbo Akademi University, Finland*
[b] *Dipartimento di Informatica, Sistemistica e Comunicazione Università degli Studi di Milano-Bicocca, Italy*
[c] *Laboratoire d'Algorithmique, Complexité et Logique, Université Paris Est – Créteil Val de Marne, France*

Abstract

Reaction systems are a new mathematical formalism inspired by the living cell and driven by only two basic mechanisms: facilitation and inhibition. As a modeling framework, they differ from the traditional approaches based on ODEs and CTMCs in two fundamental aspects: their qualitative character and the non-permanency of resources. In this article we introduce to reaction systems several notions of central interest in biomodeling: mass conservation, invariants, steady states, stationary processes, elementary fluxes, and periodicity. We prove that the decision problems related to these properties span a number of complexity classes from P to NP- and coNP-complete to PSPACE-complete.

Keywords: Reaction systems; model checking; biomodeling; conserved sets; invariants; steady state; stationary process; elementary flux; periodicity; complexity classes.

1. Introduction

Reaction systems were introduced in [6], see also [3] for a recent survey, as a framework inspired by the functioning of living cells. The central focus in reaction systems is on capturing the interactions between biochemical reactions through two basic mechanisms: facilitation and inhibition. In this framework, reactions are defined in a simple way that includes explicitly their facilitating and their inhibiting components; a reaction is presented through its (finite, nonempty) sets of reactants, inhibitors and products. Such a reaction transforms its set of reactants into its sets of products, provided that the current state includes none of its inhibitors.

There are several aspects of reaction systems that make them different from traditional modeling approaches based on, e.g., ordinary differential equations or stochastic processes. First, reactions are at the forefront of this framework, while structures are secondary: reactions *create* their next state, rather than transforming the current state. A consequence of this focus is that one may follow

Email addresses: sazimi@abo.fi (Sepinoud Azimi), cgratie@abo.fi (Cristian Gratie), sergiu.ivanov@u-pec.fr (Sergiu Ivanov), luca.manzoni@disco.unimib.it (Luca Manzoni), ipetre@abo.fi (Ion Petre), porreca@disco.unimib.it (Antonio E. Porreca)

in this framework the cause-effect relationships between reactions and answer to questions such as "what events led to this reaction being enabled now", that are difficult to answer in traditional modeling frameworks. Second, resources in reaction systems are subject to two fundamental principles: the *threshold principle* and the *non-permanency principle*. The threshold principle indicates that if present in the system, a resource is available in unlimited amounts; this effectively defines reaction systems as a qualitative framework, and eliminates any mechanism of concurrency through competition on resources. As such, modeling concurrency in reaction system has to be defined explicitly through facilitators and inhibitors, rather than implicitly through a "game of numbers" of kinetic rate (or stochastic) constants — this gives a deeper insight into the principles of the phenomenon to be modeled. The non-permanency principle indicates that the next state of a reaction system consists of only the products of the reactions enabled in the current state; this means that any resource that is not sustained by an enabled reaction will disappear from the system. In other words, sustaining a resource has to be done explicitly in reaction systems, unlike in the case of traditional modeling frameworks where it is typically implicit.

Two main lines of research have been investigated for reaction systems so far. In one line of research, the focus has been on investigating their mathematical properties, e.g., functions defined by reaction systems, state sequences, effect of limited resources, cycles and attractors, connections to propositional logic, see, e.g., [5, 7, 8, 16, 17]. On the other main line of research, the focus has been on the capabilities of reaction systems as a modeling framework, see, e.g., [1, 4, 2]. Within this line of research, there is a growing interest in model checking for reaction systems. For example, a temporal logic is introduced in [12] for defining and checking temporal properties of reaction systems; the authors prove that model checking in this logic is PSPACE-complete.

Our paper continues the investigation of reaction systems as a modeling framework and considers the formalization of several notions of central interest in biomodeling: mass conservation, invariants, steady states, stationary processes, elementary fluxes, and periodicity. Our definitions of these notions for reaction systems aim to be a natural correspondent of their usual definitions in quantitative frameworks, see, e.g. [10]. Going from a quantitative framework to a qualitative one is reflected in our definitions; for example, whereas mass conservation in a quantitative framework is a linear relation, in the context of reactions systems it becomes a set-theoretic relation on the states of the system. We focus on the computational complexity of deciding these properties for a given reaction system and we establish their complexity classes. In contrast with the quantitative case, the problems are difficult in the case of reaction systems. We prove that the decision problems related to these properties span a number of complexity classes from P to NP- and coNP-complete to PSPACE-complete.

The paper is organized as follows. In Section 2 we introduce some basic definitions of reaction systems, as well as the running case study of our paper, on the eukaryotic heat shock response. In Section 3 we formulate for reaction system the biomodeling properties mentioned above, while in Section 4 we establish their complexity classes. We discuss our conclusions in Section 5.

2. Preliminaries

In this section, we recall the notion of a reaction system, as well as some related concepts capturing the static structure and the dynamic aspects of the model. For more details we refer to [6] and [5].

Definition 2.1 ([6])**.** *Let S be a finite set. A* reaction *a in S is a triplet of finite nonempty sets $a = (R_a, I_a, P_a)$, where $R_a, I_a, P_a \subseteq S$ and $R_a \cap I_a = \varnothing$. We say that R_a, I_a, and P_a are the sets of reactants, inhibitors, and products of a, respectively. The set of all reactions in S is denoted by* $\mathrm{rac}(S)$.

A reaction system (RS) *is an ordered pair $\mathcal{A} = (S, A)$, where S is a finite set of* symbols *(also called sometimes elements, species, or entities) and $A \subseteq \mathrm{rac}(S)$. The set S is called the* background *(set) of A.*

We use the following notations:

$$\mathcal{R} = \bigcup_{a \in A} R_a, \quad \mathcal{P} = \bigcup_{a \in A} P_a, \quad \textit{and} \quad \mathrm{supp}(\mathcal{A}) = \mathcal{R} \cup \mathcal{P}.$$

The set $\mathrm{supp}(\mathcal{A})$ *will be called the* support set *of $\mathcal{A}$.*

The following definition introduces the result of a reaction and of a reaction system.

Definition 2.2 ([6])**.** *Let $\mathcal{A} = (S, A)$ be a reaction system, $T \subseteq S$, and $a \in A$. We say that a is* enabled *by T, denoted by* $\mathrm{en}_a(T)$*, if $R_a \subseteq T$ and $I_a \cap T = \varnothing$.*

(1) The result of a on T *is defined as follows:*

$$\mathrm{res}_a(T) = \begin{cases} P_a, & \textit{if } \mathrm{en}_a(T), \\ \varnothing, & \textit{otherwise.} \end{cases}$$

(2) The result of $\mathcal{A}$ on T *is defined as follows:*

$$\mathrm{res}_{\mathcal{A}}(T) = \bigcup_{a \in A} \mathrm{res}_a(T).$$

(3) An interactive process *in $\mathcal{A}$ is a pair $\pi = (\gamma, \delta)$, where $\gamma = (C_0, C_1, ..., C_n)$ and $\delta = (D_1, D_2, ..., D_n)$, $n \geq 1$, are sequences of subsets of S with $D_1 = \mathrm{res}_{\mathcal{A}}(C_0)$ and, for each $1 < i \leq n$, $D_i = \mathrm{res}_{\mathcal{A}}(C_{i-1} \cup D_{i-1})$.*

Example 2.1 (Running case-study: the eukaryotic heat shock response)**.** *We introduce in this example our running case study on the eukaryotic heat shock response. For more details we refer to [18]. The heat shock response is a defense mechanism by which the cell reacts to elevated temperatures. The key players of such a mechanism are heat shock proteins,* hsp*, which are responsible for facilitating the refolding process of misfolded proteins,* mfp*, by operating as molecular chaperons. The heat shock proteins form* hsp: mfp *complexes and help* mfp *to refold. The heat shock response is regulated by the transactivation of the* hsp*-encoding genes. The heat shock factor,* hsf*, is the protein which promotes gene transcription and, in the absence of the environmental stressors, is found abundantly bound to* hsp*'s. Heat stress results in dimerization and subsequently trimerization of* hsf*, producing* hsf_2 *and* hsf_3 *respectively. The* hsf *trimers can*

Table 1: The simplified molecular model for the eukaryotic heat shock response [1].

Reaction		Reaction	
$3\,\mathsf{hsf} \leftrightarrows \mathsf{hsf}_3$	(1)	$\mathsf{hsp} + \mathsf{hsf}_3\colon \mathsf{hse} \to \mathsf{hsp}\colon \mathsf{hsf} + 2\,\mathsf{hsf} + \mathsf{hse}$	(6)
$\mathsf{hsf}_3 + \mathsf{hse} \to \mathsf{hsf}_3\colon \mathsf{hse}$	(2)	$\mathsf{prot} \to \mathsf{mfp}$	(7)
$\mathsf{hsf}_3\colon \mathsf{hse} \to \mathsf{hsf}_3\colon \mathsf{hse} + \mathsf{hsp}$	(3)	$\mathsf{hsp} + \mathsf{mfp} \to \mathsf{hsp}\colon \mathsf{mfp}$	(8)
$\mathsf{hsp} + \mathsf{hsf} \leftrightarrows \mathsf{hsp}\colon \mathsf{hsf}$	(4)	$\mathsf{hsp}\colon \mathsf{mfp} \to \mathsf{hsp} + \mathsf{prot}$	(9)
$\mathsf{hsp} + \mathsf{hsf}_3 \to \mathsf{hsp}\colon \mathsf{hsf} + 2\,\mathsf{hsf}$	(5)		

then bind to the promoter site of the hsp*-encoding gene, i.e., the heat shock element* (hse)*; the transcription of the gene is thus activated and the synthesis of* hsp *enabled. The* hsp *synthesis is turned off as soon as the heat shock protein is raised to a sufficient level by breaking the bond in* $\mathsf{hsf}_3\colon \mathsf{hse}$*, i.e., unbinding* hsf_3 *from* hse*.*

A simple molecular model for the heat shock response was introduced in [14] and a simplified version of its molecular reactions is presented in Table 1. We follow in this paper the reaction systems version of this model, introduced in [1]. Its reactions are listed in Table 2.

The transition from Table 1 to Table 2 is not a one to one translation. To migrate from the HSR reaction-based model to the corresponding reaction system model, the authors of [1] considered various properties, including mass conservation, that should be respected in the RS model. The affinity between species (in terms of the numerical values of the kinetic rate constants) was another point to be considered. To give an insight into this construction, we explain briefly how to build the RS correspondent of reaction $(1)^+$ *of Table 1 (the left-to-right direction of the reversible reaction* (1)*). Clearly, in the RS terminology,* hsf *should be the reactant (without the mutiplicity* 3*) and the* hsf_3 *should be the product. However,* hsf *has a higher affinity to bind to* hsp *than to form trimers (in terms of the kinetic rate constant of reaction* $(4)^+$ *being higher than that of reaction* $(1)^+$*); in the ODE-based formulation of the model this translates into reaction* $(1)^+$ *having a negligible flux in the presence of (high enough levels of)* hsp*. To account for this in the RS framework, we thus list* hsp *as an inhibitor of the reaction having* hsf *as the reactant and* hsf_3 *as the product, which yields reaction* (10) *in Table 2. An additional observation is that if both* hsp *and* mfp *are present in the system, then* hsp *has a higher affinity for* mfp *than for* hsf *(in terms of the kinetic rate constant of reaction* (8) *being higher than that of reaction* $(4)^+$*); this leaves* hsf *able to form* hsf_3 *without being inhibited by* hsp*. This leads to the formulation of the RS reaction* (11) *in Table 2, explaining that in the presence of* hsf*, as well as that of* hsp *and* mfp*,* hsf_3 *will be formed. For all details on obtaining the RS model for the heat shock response we refer to [1].*

We also recall the definition of the NP-complete problem set cover [13], which will be used for reductions in a later section.

Definition 2.3. *Given a universe set* U*, a* set cover $\mathcal{F}$ *of* U *is a family of subsets of* U *having union* U*. The* set cover problem *consists of deciding, given a finite universe* U*, a set cover* $\mathcal{F} = \{F_1, \ldots, F_n\}$ *of* U*, and an integer* k*, whether there exists a set cover* $\mathcal{E} \subseteq \mathcal{F}$ *of* U *of size at most* k*.*

Table 2: The reaction system model HSR for heat shock response introduced in [1].

Reaction		Reaction	
$(\{\mathsf{hsf}\}, \{\mathsf{hsp}\}, \{\mathsf{hsf_3}\})$	(10)	$(\{\mathsf{hsp}, \mathsf{hsf}\}, \{\mathsf{mfp}\}, \{\mathsf{hsp}\colon \mathsf{hsf}\})$	(20)
$(\{\mathsf{hsf}, \mathsf{hsp}, \mathsf{mfp}\}, \{\mathsf{d_I}\}, \{\mathsf{hsf_3}\})$	(11)	$(\{\mathsf{hsp}\colon \mathsf{hsf}, \mathsf{stress}\}, \{\mathsf{nostress}\}, \{\mathsf{hsp}, \mathsf{hsf}\})$	(21)
$(\{\mathsf{hsf_3}\}, \{\mathsf{hse}, \mathsf{hsp}\}, \{\mathsf{hsf}\})$	(12)	$(\{\mathsf{hsp}\colon \mathsf{hsf}, \mathsf{nostress}\}, \{\mathsf{stress}\}, \{\mathsf{hsp}\colon \mathsf{hsf}\})$	(22)
$(\{\mathsf{hsf_3}, \mathsf{hsp}, \mathsf{mfp}\}, \{\mathsf{hse}\}, \{\mathsf{hsf}\})$	(13)	$(\{\mathsf{hsp}, \mathsf{hsf_3}\}, \{\mathsf{mfp}\}, \{\mathsf{hsp}\colon \mathsf{hsf}\})$	(23)
$(\{\mathsf{hsf_3}, \mathsf{hse}\}, \{\mathsf{hsp}\}, \{\mathsf{hsf_3}\colon \mathsf{hse}\})$	(14)	$(\{\mathsf{hsp}, \mathsf{hsf_3}\colon \mathsf{hse}\}, \{\mathsf{mfp}\}, \{\mathsf{hsp}\colon \mathsf{hsf}, \mathsf{hse}\})$	(24)
$(\{\mathsf{hsf_3}, \mathsf{hse}, \mathsf{hsp}, \mathsf{mfp}\}, \{\mathsf{d_I}\}, \{\mathsf{hsf_3}\colon \mathsf{hse}\})$	(15)	$(\{\mathsf{prot}, \mathsf{stress}\}, \{\mathsf{nostress}\}, \{\mathsf{prot}, \mathsf{mfp}\})$	(25)
$(\{\mathsf{hse}\}, \{\mathsf{hsf_3}\}, \{\mathsf{hse}\})$	(16)	$(\{\mathsf{prot}, \mathsf{nostress}\}, \{\mathsf{stress}\}, \{\mathsf{prot}\})$	(26)
$(\{\mathsf{hse}, \mathsf{hsf_3}, \mathsf{hsp}\}, \{\mathsf{mfp}\}, \{\mathsf{hse}\})$	(17)	$(\{\mathsf{hsp}, \mathsf{mfp}\}, \{\mathsf{d_I}\}, \{\mathsf{hsp}\colon \mathsf{mfp}\})$	(27)
$(\{\mathsf{hsf_3}\colon \mathsf{hse}\}, \{\mathsf{hsp}\}, \{\mathsf{hsf_3}\colon \mathsf{hse}, \mathsf{hsp}\})$	(18)	$(\{\mathsf{mfp}\}, \{\mathsf{hsp}\}, \{\mathsf{mfp}\})$	(28)
$(\{\mathsf{hsf_3}\colon \mathsf{hse}, \mathsf{hsp}, \mathsf{mfp}\}, \{\mathsf{d_I}\}, \{\mathsf{hsf_3}\colon \mathsf{hse}, \mathsf{hsp}\})$	(19)	$(\{\mathsf{hsp}\colon \mathsf{mfp}\}, \{\mathsf{d_I}\}, \{\mathsf{hsp}, \mathsf{prot}\})$	(29)

3. Biological Properties in Reaction Systems Models

In this section we formulate for reaction systems several properties that are commonly looked at in biological modeling: mass conservation, steady states, elementary fluxes, periodicity [10]. We also propose a few additional properties that generalize or are strongly related to these.

We start with the principle of mass conservation.

Definition 3.1 (Conserved sets)**.** *Let $\mathcal{A} = (S, A)$ be a reaction system. We say that a set $M \subseteq \mathrm{supp}(\mathcal{A})$ is* conserved *if for any $W \subseteq \mathrm{supp}(\mathcal{A})$, $M \cap W \neq \varnothing$ if and only if $M \cap \mathrm{res}_{\mathcal{A}}(W) \neq \varnothing$.*

Note in this definition that it is crucial that $\mathrm{supp}(\mathcal{A}) \subsetneq S$. Indeed, if $\mathrm{supp}(\mathcal{A}) = S$, then the only conserved set of $\mathcal{A}$ is the empty set; this follows from the definition by taking $W = S$ and recalling that each reaction has a non-empty set of inhibitors.

Example 3.1. *For our running example, the reaction system* HSR *in Example 2.1, a conserved set is $M = \{\mathsf{hse}, \mathsf{hsf_3}\colon \mathsf{hse}\}$. Indeed, consider an arbitrary set $W \subseteq \mathrm{supp}(\mathcal{A})$, and suppose $\mathsf{hse} \in W \cap M$. We claim that at least one of the reactions* (14)*,* (15)*,* (16)*, or* (17) *is enabled by W:*

- *if $\mathsf{hsf_3} \notin W$, then* (16) *is enabled,*
- *if $\mathsf{hsf_3} \in W$ and $\mathsf{hsp} \notin W$, then* (14) *is enabled,*
- *if $\{\mathsf{hsf_3}, \mathsf{hsp}\} \subseteq W$ and $\mathsf{mfp} \notin W$, then* (17) *is enabled, and*
- *if $\{\mathsf{hsf_3}, \mathsf{hsp}, \mathsf{mfp}\} \subseteq W$, then* (15) *is enabled.*

The product sets of reactions (14) *through* (17) *contain all either* hse *or* $\mathsf{hsf_3}\colon \mathsf{hse}$*, so $\mathrm{res}_{\mathcal{A}}(W) \cap M \neq \varnothing$. A similar reasoning can be conducted supposing that*

$\mathsf{hsf_3}\colon \mathsf{hse} \in W \cap M$ *and considering reactions* (18), (19), *or* (24) *to conclude that, in this case,* $\mathrm{res}_{\mathcal{A}}(W) \cap M \neq \varnothing$ *as well.*

Notice now that all the reactions which produce either of the species hse *or* $\mathsf{hsf_3}\colon \mathsf{hse}$ — (14) *through* (19) *and* (24) — *include either* hse *or* $\mathsf{hsf_3}\colon \mathsf{hse}$ *in their sets of reactants. This means that, whenever* $\mathrm{res}_{\mathcal{A}}(W) \cap M \neq \varnothing$*, it also holds that* $W \cap M \neq \varnothing$*. This fact, together with the argument from the previous paragraph, proves that* $W \cap M \neq \varnothing$ *if and only if* $\mathrm{res}_{\mathcal{A}}(W) \neq \varnothing$*, which means that* M *is a conserved set.*

The following definition introduces a stronger form of conservation, the invariant set.

Definition 3.2 (Invariant sets)**.** *Let* $\mathcal{A} = (S, A)$ *be a reaction system. We say that a set* $M \subseteq \mathrm{supp}(\mathcal{A})$ *is* invariant *if for any* $W \subseteq \mathrm{supp}(\mathcal{A})$*,* $M \subseteq W$ *if and only if* $M \subseteq \mathrm{res}_{\mathcal{A}}(W)$*.*

Similarly as for conserved sets, note that if $\mathrm{supp}(\mathcal{A}) = S$, then the only invariant set of $\mathcal{A}$ is the empty set. Moreover, if $M \neq \varnothing$ is an invariant set, then for any $x \in M$, there exists a reaction $a = (R, I, P)$ with $x \in P$ and $I \subseteq S - \mathrm{supp}(\mathcal{A})$. Indeed, if this is not the case, then every reaction that produces x has at least one inhibitor in $\mathrm{supp}(\mathcal{A})$ and, consequently, we can augment the set M to a set $W \subseteq \mathrm{supp}(\mathcal{A})$ that completely inhibits the production of x, thus violating the invariance of M.

Example 3.2. *Consider again the reaction system* HSR*. Based on the observation above, the only elements that can be part of some invariant set for our example are* $I_0 = \{\mathsf{hsf_3}, \mathsf{hsf_3}\colon \mathsf{hse}, \mathsf{hsp}, \mathsf{hsp}\colon \mathsf{mfp}, \mathsf{prot}\}$ *– these are the only elements produced in reactions with inhibitor set* $\{d_I\}$*.*

Assume that there is an invariant set M *such that* $\mathsf{prot} \in M$ *and consider a set* $W \subseteq \mathrm{supp}(\mathcal{A})$ *such that* $M \subseteq W$*. Then, by invariance of* M*, it must be that* $\mathsf{prot} \in M \subseteq \mathrm{res}_{\mathcal{A}}(W)$*. The only reaction producing* prot *with* d_I *as its sole inhibitor is* (29)*, so* M *must include* $\mathsf{hsp}\colon \mathsf{mfp}$ — *the unique reactant of* (29) — *to ensure that* $\mathsf{hsp}\colon \mathsf{mfp} \in W$ *and that* $\mathsf{prot} \in \mathrm{res}_{\mathcal{A}}(W)$*. Requiring* M *to also include* $\mathsf{hsp}\colon \mathsf{mfp}$ *satisfies the necessity for* M *to contain* hsp *and* mfp *(reaction* (27)*). But, by the reasoning in the previous paragraph,* mfp *cannot be part of any invariant set, because the only reaction containing* mfp *in its product set is* (25)*, and it is disabled by all sets including* nostress*, which belongs to* $\mathrm{supp}(\mathcal{A})$*. We therefore conclude that our initial supposition that* prot *may be part of an invariant set* M *is wrong.*

Similar arguments can be carried out for the other elements of I_0*, which means that the only invariant set of the reaction system* HSR *is the empty set.*

Definitions 3.1 and 3.2 of conserved and invariant sets of a reaction system $\mathcal{A} = (S, A)$ present a notable common feature: both require that certain conditions hold for $W \subseteq \mathrm{supp}(\mathcal{A})$ and $\mathrm{res}_{\mathcal{A}}(W)$. We formulate now a property that generalizes the conserved and the invariant sets.

We recall first that a *Boolean formula* φ is said to be over an alphabet S if all its variables names are from S. In the following we assume all Boolean formulae to be given in a disjunctive normal form. A subset $W \subseteq S$ is said to *satisfy* the Boolean formula φ over S if the expression for φ contains a conjunction $x_1 \wedge \ldots \wedge x_n \wedge \bar{y}_1 \wedge \ldots \wedge \bar{y}_m$ such that

(1) $\{x_i \mid 1 \leq i \leq n\} \subseteq W$, and

(2) $\{y_j \mid 1 \leq j \leq m\} \cap W = \varnothing$.

By convention, we write $\varphi(W) = 1$, or simply $\varphi(W)$, if the subset W satisfies φ, and $\varphi(W) = 0$ otherwise. For more details about the relationship between reaction systems and Boolean functions we refer to [3] and [6].

Definition 3.3 (The formula correspondence problems)**.** *Consider a reaction system $\mathcal{A} = (S, A)$. The* formula correspondence problems *for $\mathcal{A}$ are defined as follows: given two Boolean formulae φ_1 and φ_2 over S, does the following hold for every $W \subseteq \operatorname{supp}(\mathcal{A})$:*

$$\varphi_1(W) \Rightarrow \varphi_2(\operatorname{res}_{\mathcal{A}}(W)), \qquad \textit{respectively } \varphi_1(W) \Leftrightarrow \varphi_2(\operatorname{res}_{\mathcal{A}}(W)).$$

Here we write $\phi \Rightarrow \psi$ to denote implication and $\phi \Leftrightarrow \psi$ to denote equivalence.

It is easy to see that the conservation of a subset $M \subseteq S$ in $\mathcal{A}$ can be expressed as a particular case of the formula correspondence problems: M is conserved if and only if for every $W \subseteq \operatorname{supp}(\mathcal{A})$,

$$\varphi_1(W) \Leftrightarrow \varphi_2(\operatorname{res}_{\mathcal{A}}(W)), \text{ where}$$
$$\varphi_1 = \varphi_2 = \bigvee_{x \in M} x.$$

Stating that the set M is invariant can be expressed in a similar way:

$$\varphi_1(W) \Leftrightarrow \varphi_2(\operatorname{res}_{\mathcal{A}}(W)), \text{ where}$$
$$\varphi_1 = \varphi_2 = \bigwedge_{x \in M} x.$$

Since we can freely choose φ_1 and φ_2, this general approach allows for the formulation of additional, more complex properties. In particular we can make use of negative literals to specify the conservation of some set whenever particular species are not present in the current state.

Besides mass conservation, a number of other properties inspired from biology can be formulated for reaction systems. We define next the notions of steady state, stationary process, elementary flux, and periodic interactive process.

Definition 3.4 (Steady state)**.** *Let $\mathcal{A} = (S, A)$ be a reaction system. We say that $W \subseteq S$ is a* steady state *of $\mathcal{A}$ if $\operatorname{res}_{\mathcal{A}}(W) = W$.*

Example 3.3. *For our running example* HSR*, $W = \{\mathsf{hsf}, \mathsf{hsf}_3\}$ is a steady state of $\mathcal{A}$, because only reactions* (10) *and* (12) *are enabled on W, and therefore $\operatorname{res}_{\mathcal{A}}(W) = \{\mathsf{hsf}, \mathsf{hsf}_3\} = W$.*

Definition 3.5 (Stationary process)**.** *Let $\mathcal{A} = (S, A)$ be a reaction system and let the pair $\pi = ((C_n)_{n \geq 0}, (D_n)_{n \geq 1})$ be an interactive process in $\mathcal{A}$. We say that π is a* stationary process *of $\mathcal{A}$ if there exists an $n_0 \geq 1$ such that, for every $n \geq n_0$, the following holds: $C_{n+1} \cup D_{n+1} = C_n \cup D_n$.*

Example 3.4. *Consider the interactive process* $\pi = ((C_n)_{n\geq 0}, (D_n)_{n\geq 1})$ *in the reaction system* HSR *from our case study. The first element of the context sequence is defined as* $C_0 = \{\mathsf{hse}, \mathsf{hsp{:}\,hsf}, \mathsf{prot}, \mathsf{nostress}\}$*; the rest of the elements are given by* $C_n = \{\mathsf{nostress}\}$*, for* $n \geq 1$*.*

The first element D_1 *in the result sequence of* HSR *is* $D_1 = \mathrm{res}_{\mathcal{A}}(C_0) = \{\mathsf{hse}, \mathsf{hsp{:}\,hsf}, \mathsf{prot}\}$*, because only reactions* (16)*,* (22)*, and* (26) *are enabled by* C_0*. Remark that* $C_1 \cup D_1 = C_0$*, so* $D_2 = \mathrm{res}_{\mathcal{A}}(C_1 \cup D_1) = \mathrm{res}_{\mathcal{A}}(C_0) = D_1$*, and since* $C_2 = C_1$*, it holds that* $C_2 \cup D_2 = C_1 \cup D_1$*. By iterating this observation, one can show that* $C_n \cup D_n = C_1 \cup D_1$*, for* $n \geq 1$*, and hence* π *is a stationary process.*

Definition 3.6 (Elementary flux)**.** *Let* $\mathcal{A} = (S, A)$ *be a reaction system and* $W \subseteq S$ *a steady state of* $\mathcal{A}$*. We say that a subset of reactions* $E \subseteq A$ *is an* elementary flux *for* W*, if* $\mathrm{res}_{\mathcal{A}_E}(W) = W$*, where* $\mathcal{A}_E = (S, E)$*.*

Example 3.5. *Consider the steady state* $W = \{\mathsf{hsf}, \mathsf{hsf_3}\}$ *of the reaction system* HSR *modeling the heat shock response. Then the set*

$$E = \big\{ (\{\mathsf{hsf}\}, \{\mathsf{hsp}\}, \{\mathsf{hsf_3}\}),\ (\{\mathsf{hsf_3}\}, \{\mathsf{hse}, \mathsf{hsp}\}, \{\mathsf{hsf}\}) \big\},$$

comprising reactions (10) *and* (12)*, forms an elementary flux for the steady state* W*, since* $\mathrm{res}_{\mathcal{A}_E}(W) = W$*.*

Definition 3.7 (Periodic interactive processes)**.** *Let* $\mathcal{A} = (S, A)$ *be a reaction system and let the pair* $\pi = ((C_n)_{n\geq 0}, (D_n)_{n\geq 1})$ *be an interactive process in* $\mathcal{A}$*. We say that* π *is* periodic *if there exists* $n_0 \geq 1$ *and* $p \geq 1$ *(the period) such that, for every* $n \geq n_0$*, it holds that* $C_{n+p} \cup D_{n+p} = C_n \cup D_n$*.*

Example 3.6. *Again, consider the reaction system* HSR *modeling the heat shock response and an interactive process* $\pi = ((C_n)_{n\geq 0}, (D_n)_{n\geq 1})$ *of it, where* $C_0 = \{\mathsf{hsp{:}\,hsf}, \mathsf{stress}\}$*,* $C_{2n+1} = \varnothing$*, and* $C_{2n+2} = \{\mathsf{stress}\}$*, for every* $n \geq 0$*.*

The first element of the result sequence of π *is* $D_1 = \mathrm{res}_{\mathcal{A}}(C_0) = \{\mathsf{hsp}, \mathsf{hsf}\}$*, because the only reaction enabled by* C_0 *is* (21)*. Since* $C_1 = C_{2\cdot 0+1} = \varnothing$*, we have that* $D_2 = \mathrm{res}_{\mathcal{A}}(C_1 \cup D_1) = \{\mathsf{hsp{:}\,hsf}\}$*, as the only reaction enabled by* $D_1 = \{\mathsf{hsp}, \mathsf{hsf}\}$ *is* (20)*. Now,* $C_2 = C_{2\cdot 0+2} = \{\mathsf{stress}\}$*, so* $C_2 \cup D_2 = \{\mathsf{hsp{:}\,hsf}, \mathsf{stress}\} = C_0$*. We conclude therefore that all odd-numbered states of* π *have the form* $C_{2n+1} \cup D_{2n+1} = \{\mathsf{hsp}, \mathsf{hsf}\}$*, while all even-numbered states have the form* $C_{2n+2} = D_{2n+2} = \{\mathsf{hsp{:}\,hsf}, \mathsf{stress}\}$*, for* $n \geq 0$*. The interactive process* π *is therefore periodic with period* $p = 2$*.*

4. Computational Complexity of Checking the Properties

In this section we describe the computational complexity of deciding the properties defined above for a set (sequence of sets), as well as those of deciding whether sets (sequences of sets) satisfying these properties exist at all. We will use the standard notations for complexity classes. FO is the class of problems which can be described with a first-order logic formula. NP is the set of all those decision problems for which a certificate of polynomial size exists and can be in polynomial time used to verify that the instance of the problem has a positive answer. Dually, coNP comprises the problems for which the negative instances have a polynomial size certificate. The class $\Sigma_2^{\mathsf{P}} = \mathsf{NP}^{\mathsf{coNP}}$ contains all those decision problems which can be solved in polynomial time by a non-deterministic Turing machine equipped with an oracle for some coNP-complete

problem. PSPACE is the class of decision problems which can be solved by a Turing machine using a polynomial amount of space (relative to the size of its input). Finally, #P is the class of *function* problems of the form "compute $f(x)$", where f is the number of accepting paths of a non-deterministic Turing machine running in polynomial time. Note that, unlike the classes we have mentioned above, #P is a class of *counting* problems rather than decision problems.

For further details on complexity classes the reader is referred to [13] and [9].

4.1. Mass Conservation, Invariant Sets, and Formula Correspondence

The following proposition is an adapted version of [7, Theorem 7].

Theorem 4.1. *Given a reaction system $\mathcal{A} = (S, A)$ and a subset $M \subseteq \mathrm{supp}(S)$, deciding if M is conserved in $\mathcal{A}$ is a* coNP*-complete problem.*

Proof. First we prove that the problem is contained in coNP. If M is not conserved in $\mathcal{A}$, a non-deterministic Turing machine working in polynomial time can guess a set $W \subseteq \mathrm{supp}(\mathcal{A})$ such that $\neg\big(M \cap W \neq \varnothing \Leftrightarrow M \cap \mathrm{res}_{\mathcal{A}}(W) \cap M \neq \varnothing\big)$. Finding the set W requires only a polynomial number of non-deterministic choices; verifying that M is not conserved in W is a task that can be performed in polynomial time. Hence the problem is in coNP.

We now show that the property is hard for coNP. Let $\varphi = \varphi_1 \vee \ldots \vee \varphi_m$ be a DNF formula using n variables $V = \{x_1, \ldots, x_n\}$. We show that it is possible to construct a reaction system $\mathcal{A}$ and a set M such that M is conserved in $\mathcal{A}$ iff φ is a valid formula. Denote by $\mathrm{pos}(\varphi_i)$ the set of variables appearing as positive literals (i.e., non-negated) in φ_i and by $\mathrm{neg}(\varphi_i)$ the set of variables appearing negated in φ_i.

The reaction system $\mathcal{A} = (S, A)$ has as background set $S = V \cup \{\heartsuit, \spadesuit\}$. Given a set $W \subseteq S$, we associate an assignment of φ to W in the following way: a variable x_i is assigned the value *true* if $x_i \in W$, and it is assigned the value *false* otherwise. We also assign to set $W \subseteq S$ a number $k \in \{0, \ldots, 2^{n-1}\}$ defined as $\sum_{i=0}^{n-1} [x_{i+1} \in W] 2^i$, where $[P]$ has value 1 if P is a true predicate and 0 otherwise. That is, we consider the entities $x_1, \ldots, x_n$ as the digits of an n-bit number, where the $(i-1)$-th digit is 1 if $x_i \in W$ and 0 otherwise. We will denote by $B_k \subseteq \{x_1, \ldots, x_n\} \subseteq S$ the set of entities representing the number $k \in \{0, \ldots, 2^n - 1\}$.

The following reactions of $\mathcal{A}$ implement an n-bit counter incremented at every step and overflowing at $2^n - 1$:

$$(\{x_{i-1}, \ldots, x_1, \heartsuit\}, \{x_i, \spadesuit\}, \{x_i\}) \qquad \text{for } 1 \leq i \leq n, \tag{30}$$

$$(\{x_i, \heartsuit\}, \{x_j, \spadesuit\}, \{x_i\}) \qquad \text{for } 1 \leq j < i \leq n. \tag{31}$$

A reaction of type (30) sets a previously zero bit to one when all the bits with lower indices are one. A reaction of type (31) preserves a bit with value one if there is at least one zero bit with a lower index.

The following reactions of $\mathcal{A}$ preserve $\heartsuit$ if $\varphi(W)$:

$$(\mathrm{pos}(\varphi_i) \cup \{\heartsuit\}, \mathrm{neg}(\varphi_i) \cup \{\spadesuit\}, \{\heartsuit\}) \qquad \text{for } 1 \leq i \leq m. \tag{32}$$

Reactions of type (32) produce $\heartsuit$ if $\spadesuit$ is not present and if there exists a term φ_i that is satisfied by W (and hence, $\varphi(W)$). Notice that $\spadesuit \notin \mathrm{supp}(\mathcal{A}) = V \cup \{\heartsuit\}$;

its existence is due to the necessity of having non-empty inhibitors on reactions of type (32) if some $\mathrm{neg}(\varphi_i)$ is empty.

The result function defined by reactions of types (30)–(32) is the following one:

$$\mathrm{res}_{\mathcal{A}}(W) = \begin{cases} \varnothing & \text{if } \heartsuit \notin W \text{ or } \spadesuit \in W, \\ B_{(k+1) \bmod 2^n} \cup \{\heartsuit\} & \text{if } W \subseteq V \cup \{\heartsuit\},\ W \cap V = B_k,\ \varphi(W) = 1, \\ B_{(k+1) \bmod 2^n} & \text{if } W \subseteq V \cup \{\heartsuit\},\ W \cap V = B_k,\ \varphi(W) = 0. \end{cases}$$

We now show that, for each $W \subseteq \mathrm{supp}(\mathcal{A})$, it is true that $\{\heartsuit\} \cap W \neq \varnothing \Leftrightarrow \{\heartsuit\} \cap \mathrm{res}_{\mathcal{A}}(W) \neq \varnothing$ (i.e., the set $\{\heartsuit\} \subseteq \mathrm{supp}(\mathcal{A})$ is conserved in $\mathcal{A}$) if and only if φ is a valid formula. Given a set $W \subseteq \mathrm{supp}(\mathcal{A}) = V \cup \{\heartsuit\}$, either $\heartsuit \notin W$ (that is, $\{\heartsuit\} \cap W = \varnothing$), or $\heartsuit \in W$ (and then $W = B_k \cup \{\heartsuit\}$ for some k).

1. Assuming $\{\heartsuit\} \cap W = \varnothing$, no reaction $a \in A$ is enabled, since they all have $\heartsuit$ as a reactant; hence $\mathrm{res}_{\mathcal{A}}(W) = \bigcup_{a \in A} \mathrm{res}_a(W) = \varnothing$.
2. On the other hand, if $W = B_k \cup \{\heartsuit\}$ for some k, and furthermore B_k satisfies φ, the next state will be $B_{(k+1) \bmod 2^n} \cup \{\heartsuit\}$, $\mathcal{A}$ preserves $\heartsuit$, and we end up in this case again. If however B_k does not entail φ the next state will be $B_{(k+1) \bmod 2^n}$, $\heartsuit$ is not preserved, and we end up in the former case.

Since the binary counter defined by reactions of types (30) and (31) iterates across all the possible subsets of V (i.e., all the possible assignments of φ), the set $\{\heartsuit\}$ is preserved if and only if there is no W such that $\varphi(W) = 0$, that is, if and only if φ is a valid formula. The problem of establishing if a formula φ is valid is coNP-complete [13]; furthermore, the mapping $\varphi \mapsto (\mathcal{A}, \{\heartsuit\})$ is computable in polynomial time, since the number of reactions is $O(n^2 + m)$ and each of them either has a simple dependence from n (reactions of types (30) and (31)) or consists in a rewriting of the terms of φ (reactions of type (32)); as a consequence, the statement of the theorem holds. □

The observation that conserved singleton sets are also invariant sets allows to directly derive the following result.

Corollary 4.1. *Given a reaction system $\mathcal{A} = (S, A)$ and a set $M \subseteq \mathrm{supp}(\mathcal{A})$, deciding whether M is an invariant set of $\mathcal{A}$ is a* coNP*-complete problem.*

Proof. When M is a singleton, M is an invariant set of $\mathcal{A}$ if and only if M is conserved in $\mathcal{A}$. Therefore the proof of Theorem 4.1 also proves the coNP-hardness of this problem. It is only necessary to show that the problem is in coNP. Notice that it is possible to non-deterministically guess a state of $\mathcal{A}$ and, by computing the next state function, to verify in polynomial time that a given set M is not an invariant set. This shows that the problem lies in coNP. □

The previous two results admit the following straightforward generalization to the formula correspondence problems.

Corollary 4.2. *Given a reaction system $\mathcal{A} = (S, A)$ and two Boolean formulae φ_1 and φ_2, deciding whether*

$$\forall W \subseteq \mathrm{supp}(\mathcal{A}) \,.\, \varphi_1(W) \Rightarrow \varphi_2(\mathrm{res}_{\mathcal{A}}(W))$$

is coNP*-complete. The same is true also for deciding whether*

$$\forall W \subseteq \mathrm{supp}(\mathcal{A}) \,.\, \varphi_1(W) \Leftrightarrow \varphi_2(\mathrm{res}_{\mathcal{A}}(W)).$$

Proof. Both problems are in coNP because verifying that $\varphi_1(W) \# \varphi_2(\mathrm{res}_{\mathcal{A}}(W))$ does *not* hold for a candidate set $W \subseteq S$ can be done in polynomial time, where $\# \in \{\Rightarrow, \Leftrightarrow\}$.

Let $\mathcal{A}$ be a reaction system as in the proof of Theorem 4.1 and let

$$\varphi_1 = \varphi_2 = \heartsuit$$

be Boolean formulae consisting only of the literal $\heartsuit$, thus trivially in DNF.

Then, the formula $\forall W \subseteq \mathrm{supp}(\mathcal{A}) \,.\, \varphi_1(W) \Rightarrow \varphi_2(\mathrm{res}_{\mathcal{A}}(W))$ means that, for all $W \subseteq \mathrm{supp}(\mathcal{A})$, if $\heartsuit \in W$ then $\heartsuit \in \mathrm{res}_{\mathcal{A}}(W)$, which is true if and only if the formula φ encoded by $\mathcal{A}$ is valid.

On the other hand, the formula $\forall W \subseteq \mathrm{supp}(\mathcal{A}) \,.\, \varphi_1(W) \Leftrightarrow \varphi_2(\mathrm{res}_{\mathcal{A}}(W))$ is simply the conservation of $\{\heartsuit\}$, which once again holds if and only if φ is valid.

The reaction system $\mathcal{A}$ can be constructed in polynomial time from φ as in the proof of Theorem 4.1, and the formulae φ_1 and φ_2 are constant; hence, the mapping $\varphi \mapsto (\mathcal{A}, \varphi_1, \varphi_2)$ can be computed in polynomial time, and this establishes the coNP-hardness of the problem. □

Note that, by definition, the empty set is always conserved and invariant. Thus, in the existence problems we will focus on finding non-empty sets.

Theorem 4.2. *Given a reaction system $\mathcal{A} = (S, A)$, deciding if there exists a non-empty conserved set $M \subseteq \mathrm{supp}(\mathcal{A})$ is a* coNP*-hard problem contained in Σ_2^{P}.*

Proof. To show the coNP-hardness of the problem it is only necessary to notice that in the proof of Theorem 4.1 the only possible non-empty conserved set is $\{\heartsuit\}$, since for each "candidate" non-empty conserved set M different from $\{\heartsuit\}$ there are two cases:

- $M = \{\heartsuit\} \cup M'$, with $\heartsuit \notin M'$. In this case $M \cap \mathrm{res}_{\mathcal{A}}(M') = \varnothing$ but $\mathrm{res}_{\mathcal{A}}(M') = \varnothing$, thus $M \cap \varnothing = \varnothing$ and M is not conserved.
- $\heartsuit \notin M$. In this case $M \cap M \neq \varnothing$ but $\mathrm{res}_{\mathcal{A}}(M) = \varnothing$, thus M is not conserved.

Therefore, determining if there exists a non-empty conserved set is as difficult as deciding if $\{\heartsuit\}$ is conserved.

To show that the problem is contained in Σ_2^{P} it is only necessary to notice that a non-deterministic machine with an oracle for coNP can non-deterministically guess a state and then determine if it is a conserved state using the coNP oracle (Theorem 4.1). Therefore the problem is contained in $\mathsf{NP}^{\mathsf{coNP}} = \Sigma_2^{\mathsf{P}}$. □

By following the same steps of Theorem 4.2 and observing that $\{\heartsuit\}$ is the only possible invariant set, we also obtain the following result:

Corollary 4.3. *Given a reaction system $\mathcal{A} = (S, A)$, deciding if there exists a non-empty invariant set $M \subseteq S$ is a* coNP*-hard problem contained in Σ_2^{P}.*

4.2. Steady States and Elementary Fluxes

As shown in [8, Corollary 1], deciding whether a state M is a steady state of a reaction system $\mathcal{A}$ is in FO and therefore can be done in polynomial time. The following theorem is an adaptation of [8, Theorem 2], and shows that deciding whether there exists a non-trivial steady state in a reaction system is an NP-complete problem.

Theorem 4.3. *Given a reaction system $\mathcal{A} = (S, A)$, deciding if there exists a non-empty steady state $M \subseteq S$ of $\mathcal{A}$ is an NP-complete problem.*

Proof. First of all we show that the problem lies in NP. Notice that it is possible to non-deterministically guess a set M and verify, in polynomial time, that $\mathrm{res}_{\mathcal{A}}(M) = M$, thus showing that the problem is contained in NP.

To prove the hardness of the problem, we reduce SAT [13] to it. Let $\varphi = \varphi_1 \wedge \ldots \wedge \varphi_m$ be a CNF formula over the variables $V = \{x_1, \ldots, x_n\}$. We define a reaction system $\mathcal{A} = (S, A)$ where $S = V \cup \{\heartsuit, \spadesuit\}$ and where A contains the following reactions (for the notations see the proof of Theorem 4.1):

$$(\mathrm{neg}(\varphi_i) \cup \{\heartsuit\}, \mathrm{pos}(\varphi_i) \cup \{\spadesuit\}, \{\spadesuit\}) \qquad \text{for } 1 \leq i \leq m \tag{33}$$

$$(\{x_i, \heartsuit\}, \{\spadesuit\}, \{x_i\}) \qquad \text{for } 1 \leq i \leq n \tag{34}$$

$$(\{\heartsuit\}, \{\spadesuit\}, \{\heartsuit\}). \tag{35}$$

Reactions of type (33) produce $\spadesuit$ for state T when there is a non-satisfied clause in φ when considering the assignment given by T. Reactions of type (34) and (35) preserve the current state if it does not contain $\spadesuit$. The result function represented by $\mathcal{A}$ is then:

$$\mathrm{res}_{\mathcal{A}}(T) = \begin{cases} T & \text{if } \heartsuit \in T,\ \spadesuit \notin T, \text{ and } \varphi(T) = 1 \\ T \cup \{\spadesuit\} & \text{if } \heartsuit \in T,\ \spadesuit \notin T, \text{ and } \varphi(T) = 0 \\ \varnothing & \text{otherwise.} \end{cases}$$

From $\mathrm{res}_{\mathcal{A}}$ it is possible to observe that only sets in the form $U \cup \{\heartsuit\}$ for some $U \subseteq V$ do not necessarily end up in $\varnothing$. In particular, a set T of such form is a steady state of $\mathcal{A}$ iff $\varphi(T)$, thus showing the NP-completeness of the problem of determining the existence of a non-empty steady state for $\mathcal{A}$. □

As far as elementary fluxes are concerned, note first that the decision problem can be solved in polynomial time. Indeed, consider a reaction system $\mathcal{A} = (S, A)$ and W a steady state of $\mathcal{A}$. If we are given a set of reactions E, we only need to test that W is a steady state of the reduced system (S, E). Moreover, note that, by definition, $E = A$ is an elementary flux. To find non-trivial elementary fluxes, we constrain the cardinality of E, i.e. we look for sets of at most k reactions that preserve the steady state W. The following theorem describes the complexity of deciding their existence and counting them.

Theorem 4.4. *Given a reaction system $\mathcal{A} = (S, A)$, a steady state W of $\mathcal{A}$, and an integer $k < |A|$:*

- *it is NP-complete to decide if there exists an elementary flux for W with at most k reactions;*
- *it is $\#\mathsf{P}$-complete to count such elementary fluxes.*

Proof. Let $(U, \mathcal{F}, k)$ be an instance of the set cover problem. Consider the reaction system $\mathcal{A} = (S, A)$, where $S = U \cup \{\spadesuit\}$ and A contains the reaction $a_F = (U, \{\spadesuit\}, F)$ for each $F \in \mathcal{F}$. Then, we have

$$\mathrm{res}_{\mathcal{A}}(U) = \bigcup_{F \in \mathcal{F}} \mathrm{res}_{a_F}(U) = \bigcup \mathcal{F} = U.$$

If we let $\mathcal{E} \subseteq \mathcal{F}$ and $E = \{a_F \in A : F \in \mathcal{E}\}$, then we have $\mathrm{res}_{\mathcal{A}_E}(U) = U$ with $|E| \leq k$ if and only if $\bigcup \mathcal{E} = U$ with $|\mathcal{E}| \leq k$. Furthermore, the mapping $\mathcal{E} \mapsto E$ is a bijection between the set covers $\mathcal{E} \subseteq \mathcal{F}$ of U and the elementary fluxes of U, hence reductions between $(U, \mathcal{F}, k)$ and $(\mathcal{A}, U, k)$ are computable in polynomial time. The statement of the theorem follows from the NP-completeness of set covering and the #P-completeness of its counting version [15]. □

4.3. Periodicity and Stationarity

Even if a sequence of contexts $(C_n)_{n \geq 0}$ for an RS $\mathcal{A} = (S, A)$ is computable by means of a function $n \mapsto C_n$, certain properties of the resulting interactive process, such as periodicity or stationarity, are undecidable. Indeed, given a Turing machine M, we can build the sequence of contexts $n \mapsto C_n$ defined by

$$C_n = \begin{cases} S & \text{if } n \text{ is a power of 2 or } M \text{ halts on empty input within } n \text{ steps} \\ \varnothing & \text{otherwise.} \end{cases}$$

The n-th element of the context sequence is computable by checking whether n is a power of two and by simulating, by means of a universal Turing machine, the first n computation steps of M and checking whether it has halted. The trivial reaction system with no reactions has $(C_n)_{n \geq 0}$ as its state sequence, and this sequence is stationary (constantly S) if and only if the Turing machine M eventually halts. The same state sequence is periodic (namely, with period 1) only when it is stationary; thus, periodicity is also undecidable.

In fact, a universal Turing machine is not even needed, since a formalism as weak as FO logic can simulate the behaviour of a Turing machine, as shown by the proof of Trakhtenbrot's theorem [11].

When we restrict the sequences of contexts to periodic ones, they become redundant, as the reaction systems themselves are able to generate such contexts internally.

Theorem 4.5. *Consider the reaction system $\mathcal{A} = (S, A)$ and the interactive process $\pi = \big((C_n)_{n \geq 0}, (D_n)_{n \geq 1}\big)$ with a periodic context sequence, i.e. there exists an $n_0 \geq 0$ such that, for every $n \geq n_0$, $C_{n+p} = C_n$, where p is the period of the context sequence. Deciding whether π is periodic is* PSPACE*-complete.*

Proof. Deciding periodicity is already PSPACE-hard for reaction systems which do not interact with the environment after the first step of evolution, i.e. in which $C_n = \varnothing$ for all $n > 0$ [7].

To prove that deciding the periodicity of an interactive process is in PSPACE we will show how interaction with a periodic context sequence can be simulated within the reaction system itself. Based on $\mathcal{A} = (S, A)$, we construct $\mathcal{A}' =$

(S', A') where the background set and the set of reactions are defined as follows:

$$\begin{aligned} S' &= \{c_i \mid 1 \leq i < n_0 + p\} \cup \{\spadesuit\} \cup S, \\ A' &= \big\{ (\{c_i\}, \{\spadesuit\}, C_{i+1} \cup \{c_{i+1}\}) \mid 1 \leq i < n_0 \big\} \\ &\cup \big\{ (\{c_{n_0+j}\}, \{\spadesuit\}, C_{n_0+j'} \cup \{c_{n_0+j'}\}) \mid 0 \leq j < p,\, j' = (j+1) \bmod p \big\} \\ &\cup A. \end{aligned}$$

Thus, besides the species from $\mathcal{A}$, S' includes one symbol per possible context from the periodic context sequence of π, as well as $\spadesuit$ to serve as a dummy inhibitor. The reactions from A' include all the reactions from A, but also the reactions simulating the injection of the context based on the cyclic counter implemented by the symbols c_i, $1 \leq i < n_0 + p$.

From the construction of $\mathcal{A}'$ we can deduce the following properties, for any subset $T \subseteq S$:

1. $\mathrm{res}_{\mathcal{A}'}(T \cup \{c_i\}) = \mathrm{res}_{\mathcal{A}}(T) \cup C_{i+1} \cup \{c_{i+1}\}$, for $1 \leq i < n_0$, and
2. $\mathrm{res}_{\mathcal{A}'}(T \cup \{c_{n_0+j}\}) = \mathrm{res}_{\mathcal{A}}(T) \cup C_{n_0+j'} \cup \{c_{n_0+j'}\}$, for $0 \leq j < p$ and $j' = (j+1) \bmod p$.

In other words, the symbols c_i, $1 \leq i < n_0 + p$ correctly simulate the interaction with environment of the interactive process π. Hence, in the interactive process $\pi' = ((C'_n)_{n \geq 0}, (D'_n)_{n \geq 1})$ of $\mathcal{A}'$ starting at $C'_0 = C_0 \cup \{c_0\}$ and not subsequently interacting with the environment ($C'_n = \varnothing$ for $n > 0$), the i-th element of the result sequence will be $D'_i = D_i \cup C_i \cup \{c_i\}$. Therefore π is periodic if and only if π' is, and deciding the periodicity of π' is in PSPACE [7]. Since the reduction $(\mathcal{A}, C_0, \ldots, C_{n_0}, \ldots, C_{n_0+p-1}) \mapsto \mathcal{A}'$ can be performed in polynomial time, and since PSPACE is closed under such reductions, we conclude that deciding periodicity of π is in PSPACE as well. □

On the other hand, the *stationarity* of the interactive process generated by a periodic (resp., ultimately periodic) sequence of contexts can be efficiently detected by simply checking whether the first n states of the system are identical, where n is the length of the period of the sequence of contexts (resp., the length of the period plus the length of the pre-period).

We summarize the results we obtained in this paper in the following general theorem.

Theorem 4.6. *The following statements are true for a reaction system $\mathcal{A}$:*

1. *deciding if $M \subseteq \mathrm{supp}(\mathcal{A})$ is conserved in $\mathcal{A}$ is* coNP*-complete;*
2. *deciding if $M \subseteq \mathrm{supp}(\mathcal{A})$ is an invariant set of $\mathcal{A}$ is* coNP*-complete;*
3. *deciding the formula correspondence problem for two Boolean formulae is* coNP*-complete;*
4. *deciding if $\mathcal{A}$ conserves a non-empty set is a* coNP*-hard problem contained in Σ_2^{P};*
5. *deciding if $\mathcal{A}$ has a non-empty invariant set is a* coNP*-hard problem contained in Σ_2^{P};*

6. *deciding if $\mathcal{A}$ has a non-empty steady state is* NP*-complete;*

7. *for a steady state W of $\mathcal{A}$, deciding the existence of an elementary flux of size at most k is* NP*-complete;*

8. *for a steady state W of $\mathcal{A}$, counting the elementary fluxes of size at most k is* #P*-complete;*

9. *deciding the periodicity of an interactive process of $\mathcal{A}$ with a periodic context sequence is* PSPACE*-complete.*

10. *deciding the stationarity of an interactive process of $\mathcal{A}$ with a periodic context sequence is in* P*.*

5. Conclusion

We considered in this paper some of the properties of central interest in biomodeling: mass conservation, invariants, steady states, stationary processes, elementary fluxes, and periodicity. We defined them in the case of reaction systems so that they extend in a natural way from the usual quantitative case to the qualitative framework of reaction systems. We then established the complexity class of checking these properties for reaction systems models.

The results of our paper, summarized in Theorem 4.6, show that checking for these properties is difficult in the sense of computational complexity. On one hand, this is surprising because checking them in quantitative frameworks such as ODE-based reaction models or Petri-net models is in general easy and reduces to problems of linear algebra. On the other hand, our results are in line with [7, 8] who show that checking properties such as fixed points, cycles, and attractors are also difficult problems for reaction systems. Moreover, our conclusions are in line with the recent results of [12] that introduce a temporal logic for reaction systems and show that model checking in this logic is PSPACE-complete. The properties we introduce in this paper can be formulated in the temporal logic of [12] and we prove that even in these special cases the problems remain intractable, albeit on lower complexity classes than PSPACE-complete. This gives an interesting insight into the complex dynamics of reaction systems, a framework in which the quantitative competition on resources is replaced by Boolean logic-based facilitation and inhibition.

Acknowledgements

Sepinoud Azimi, Cristian Gratie and Ion Petre gratefully acknowledge support from Academy of Finland through projects 267915 and 272559.

References

[1] Sepinoud Azimi, Bogdan Iancu, and Ion Petre. Reaction system models for the heat shock response. *Fundamenta Informaticae*, 131(3):299–312, 2014.

[2] Sepinoud Azimi, Charmi Panchal, Eugen Czeizler, and Ion Petre. Reaction systems models for the self-assembly of intermediate filaments. Annals of the University of Bucharest, 2015, to appear.

[3] Robert Brijder, Andrzej Ehrenfeucht, Michael G. Main, and Grzegorz Rozenberg. A tour of reaction systems. *International Journal of Foundations of Computer Science*, 22(7):1499–1517, 2011.

[4] Luca Corolli, Carlo Maj, Fabrizio Marini, Daniela Besozzi, and Giancarlo Mauri. An excursion in reaction systems: From computer science to biology. *Theoretical Computer Science*, 454:95–108, 2012.

[5] Andrzej Ehrenfeucht, Michael Main, and Grzegorz Rozenberg. Functions defined by reaction systems. *International Journal of Foundations of Computer Science*, 22(01):167–178, 2011.

[6] Andrzej Ehrenfeucht and Grzegorz Rozenberg. Reaction systems. *Fundamenta Informaticae*, 75(1):263–280, 2007.

[7] Enrico Formenti, Luca Manzoni, and Antonio E. Porreca. Cycles and global attractors of reaction systems. In Helmut Jürgensen, Juhani Karhumäki, and Alexander Okhotin, editors, *Descriptional Complexity of Formal Systems*, volume 8614 of *Lecture Notes in Computer Science*, pages 114–125. Springer, 2014.

[8] Enrico Formenti, Luca Manzoni, and Antonio E. Porreca. Fixed points and attractors of reaction systems. In Arnold Beckmann, Erzsébet Csuhaj-Varjú, and Klaus Meer, editors, *Language, Life, Limits, 10th Conference on Computability in Europe, CiE 2014*, volume 8493 of *Lecture Notes in Computer Science*, pages 194–203. Springer, 2014.

[9] Neil Immerman. *Descriptive Complexity*. Graduate texts in computer science. Springer New York, 1999.

[10] Edda Klipp, Ralf Herwig, Axel Kowald, Christoph Wierling, and Hans Lehrach. *Systems biology in practice: concepts, implementation and application*. John Wiley & Sons, 2008.

[11] Leonid Libkin. *Elements of Finite Model Theory*. Springer, 2012.

[12] Artur Męski, Wojciech Penczek, and Grzegorz Rozenberg. Model checking temporal properties of reaction systems. *Information Sciences*, 313:22–42, 2015.

[13] Christos H. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1993.

[14] Ion Petre, Andrzej Mizera, Claire L. Hyder, Annika Meinander, Andrey Mikhailov, Richard I. Morimoto, Lea Sistonen, John E. Eriksson, and Ralph-Johan Back. A simple mass-action model for the eukaryotic heat shock response and its mathematical validation. *Natural Computing*, 10(1):595–612, 2011.

[15] J Scott Provan and Michael O. Ball. The complexity of counting cuts and of computing the probability that a graph is connected. *SIAM Journal on Computing*, 12(4):777–788, 1983.

[16] Arto Salomaa. Functions and sequences generated by reaction systems. *Theoretical Computer Science*, 466:87–96, 2012.

[17] Arto Salomaa. Functional constructions between reaction systems and propositional logic. *International Journal of Foundations of Computer Science*, 24(1):147–159, 2013.

[18] Richard Voellmy and Frank Boellmann. Chaperone regulation of the heat shock protein response. In Peter Csermely and László Vígh, editors, *Molecular Aspects of the Stress Response: Chaperones, Membranes and Networks*, volume 594 of *Advances in Experimental Medicine and Biology*, pages 89–99. Springer New York, 2007.

Paper VI

Sepinoud Azimi, Cristian Gratie, Sergiu Ivanov, Ion Petre, Dependency graphs and mass conservation in reaction systems, Theoretical Computer Science 598, 23–39, 2015.

Theoretical Computer Science ••• (••••) •••–•••

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Dependency graphs and mass conservation in reaction systems

Sepinoud Azimi [a], Cristian Gratie [a,*], Sergiu Ivanov [b], Ion Petre [a]

[a] Turku Centre for Computer Science and Department of Information Technologies, Åbo Akademi University, Finland
[b] Université Paris Est – Créteil Val de Marne, France

ARTICLE INFO

Article history:
Received 14 October 2014
Received in revised form 16 January 2015
Accepted 2 February 2015
Available online xxxx
Communicated by M.P. Jimenez

Keywords:
Reaction system
Model checking
Mass conservation
Conserved set
Conservation dependency graph
Simulator

ABSTRACT

Reaction systems is a new mathematical formalism inspired by the biological cell, which focuses on an abstract set-based representation of chemical reactions via facilitation and inhibition. In this article we focus on the property of mass conservation for reaction systems. We show that conservation of sets gives rise to a relation between the species, which we capture in the concept of the conservation dependency graph. We then describe an application of this relation to the problem of listing all conserved sets. We further give a sufficient negative polynomial criterion which can be used for proving that a set is not conserved. Finally, we present a simulator of reaction systems, which also includes an implementation of the algorithm for listing the conserved sets of a given reaction system.

1. Introduction

Reaction systems is a framework inspired by the functioning of the living cells, which was originally introduced in [1]. This formalism focuses on reactions exclusively and only considers two basic ways in which they can interact: promotion and inhibition. Reaction systems are based upon two fundamental principles. The first one, referred to as the "threshold principle", states that, whenever a resource is available, it is available in unlimited amount. This implies in particular that no competition for resources takes place. The second principle, referred to as the "no-permanency principle", states that unless a resource is explicitly sustained by a process, it will vanish and thus it will not be present in the next state of the system.

One of the central features of reaction systems is that they are explicitly conceived as open-ended systems: the influence of the environment is represented as an inflow of resources (the context).

The research topics investigated in the domain of reaction systems are various [2], but they can generally be classified along two lines. The first line comprises the research focusing on the mathematical properties of reaction systems: the set functions they can implement, their state sequences, connections to Boolean functions, etc. (e.g., [3–7]). The second main line of research regards reaction systems as an instrument for biological modeling (e.g., [8–10]). Quite naturally, investigations along this second line led to the study of model checking for reaction systems. For example, in [11], the authors introduce a temporal logic to define and subsequently verify certain properties of reaction systems. They prove that the general model checking problem is PSPACE-complete. On the other hand, [8] starts with defining a series of biologically

* Corresponding author.
E-mail addresses: sazimi@abo.fi (S. Azimi), cgratie@abo.fi (C. Gratie), sergiu.ivanov@u-pec.fr (S. Ivanov), ipetre@abo.fi (I. Petre).

http://dx.doi.org/10.1016/j.tcs.2015.02.014
0304-3975/

inspired properties for reaction systems and shows that checking some of them, while still intractable, is a problem of lower computational complexity.

In this paper we conduct a detailed study of the biologically inspired property of mass conservation in reaction systems, originally introduced and shown to be coNP-complete in [8]. We get a new insight into the connection between the internal structure of the reaction system and mass conservation by revealing a relation that the latter induces between the species, and we capture this relation by defining the conservation dependency graph. We then present an application of this graph to the inherently difficult problem of listing the conserved sets and show that, in certain cases, the algorithm we devise to solve this problem is capable of performing better than the naive exponential approach. We continue by regarding mass conservation from a yet another perspective and formulate a sufficient polynomial criterion which allows one to quickly decide that a given set of species is not conserved. Finally, we present the reaction system simulator we have developed with the goal of automating the process of running reaction systems, and which is also capable of building the conservation dependency graph of a reaction system and of using it to list the conserved sets.

This paper is structured as follows. In Section 2 we remind the basic notions of reaction systems, as well as the notion of mass conservation. In Section 3 we discuss the relationship between mass conservation and the inner structure of the reaction system, and introduce the conservation dependency graph. In Section 4 we describe the algorithm for listing the conserved sets, which is based on the conservation dependency graph. In Section 5 we provide a negative polynomial heuristics for mass conservation, as well as for a generalized conservation problem. Finally, in Section 6 we give a short presentation of our reaction systems simulator. We conclude the paper in Section 7 with a discussion of our work.

2. Preliminaries

In this section we remind the notion of a reaction system as well as some related concepts capturing the static structure and the dynamic aspects of the model. For the original introduction the reader is referred to [1] and [3].

Definition 2.1. (See [1].) Let S be a finite set, whose elements will be referred to as *species* (very often in the reaction systems literature they are also called *entities*). A *reaction* a in S is a triplet of finite sets $a = (R_a, I_a, P_a)$, where $R_a, I_a, P_a \subseteq S$ and $R_a \cap I_a = \varnothing$. We say that R_a, I_a, and P_a are the sets of *reactants*, *inhibitors*, and *products* of a, respectively. The set of all reactions in S is denoted by $\mathrm{rac}(S)$.

A *reaction system (RS)* is an ordered pair $\mathcal{A} = (S, A)$, where S is a finite set of species and $A \subseteq \mathrm{rac}(S)$. The set S is called the *background* (set) of $\mathcal{A}$.

We use the following notations of [8]:

$$\mathcal{R} = \bigcup_{a \in A} R_a, \mathcal{P} = \bigcup_{a \in A} P_a, \text{ and } \mathrm{supp}(\mathcal{A}) = \mathcal{R} \cup \mathcal{P}.$$

The set $\mathrm{supp}(\mathcal{A})$ will be called the *support set* of $\mathcal{A}$.

The following definition introduces the result of a reaction and of a reaction system.

Definition 2.2. (See [1].) Let $\mathcal{A} = (S, A)$ be a reaction system, $W \subseteq S$ a set of species, and $a \in A$ a reaction. We say that a is *enabled* by W, denoted by $\mathrm{en}_a(W)$, if $R_a \subseteq W$ and $I_a \cap W = \varnothing$.

(1) The *result of a on W* is defined as follows:

$$\mathrm{res}_a(W) = \begin{cases} P_a, & \text{if } \mathrm{en}_a(W), \\ \varnothing, & \text{otherwise.} \end{cases}$$

(2) The *result of $\mathcal{A}$ on W* is defined as follows:

$$\mathrm{res}_{\mathcal{A}}(W) = \bigcup_{a \in A} \mathrm{res}_a(W).$$

Next, we introduce a running example for this section, as well as Sections 3 and 4.

Example 2.1. One of the best preserved defense mechanisms in the living cell is the *heat shock response*. Whenever the cell is exposed to environmental stress, its proteins start to misfold, which may eventually lead to cell death. The heat shock response mechanism causes an increase in the production of molecular chaperons called the heat shock response proteins (hsp). These chaperons bind to misfolded proteins and facilitate their refolding. A different group of proteins, the heat shock factors (hsf), control hsp expression by binding to the promoter site of the hsp-encoding gene (the heat-shock element hse) and thus activate the transcription of hsp. A molecular model of such a mechanism is proposed in [12] and its corresponding reaction system based model is presented in [9]. In this paper we will consider the following simplified version of the model from [9]:

Please cite this article in press as: S. Azimi et al., Dependency graphs and mass conservation in reaction systems, Theoret. Comput. Sci. (2015), http://dx.doi.org/10.1016/j.tcs.2015.02.014

$S = \{$ hsf, hsp, hse, hsp:hsf, hsf:hse, $d_I\}$

$A = \{$ ({hsf}, {hsp}, {hsf}),

({hsp, hsf}, $\{d_I\}$, {hsp:hsf}),

({hsp:hsf}, $\{d_I\}$, {hsp, hsf}),

({hse}, {hsf}, {hse}),

({hsf, hse}, {hsp}, {hsf:hse}),

({hsf:hse}, {hsp}, {hsf:hse, hsp}),

({hsp, hsf, hse}, $\{d_I\}$, {hsp:hsf, hse}),

({hsp, hsf:hse}, $\{d_I\}$, {hsp:hsf, hse})$\}$

For this particular example, the support set contains all the species except for d_I:

$$\text{supp}(\mathcal{A}) = \{\text{hsf, hsp, hse, hsp:hsf, hsf:hse}\}$$

We now recall the notion of mass conservation in reaction systems.

Definition 2.3. (See [8].) Let $\mathcal{A} = (S, A)$ be a reaction system. We say that a set $M \subseteq \text{supp}(\mathcal{A})$ is *conserved* if for any $W \subseteq \text{supp}(\mathcal{A})$, $M \cap W \neq \varnothing$ if and only if $M \cap \text{res}_{\mathcal{A}}(W) \neq \varnothing$.

The biological intuition behind a set M being conserved is that the entities in M represent different forms of the same species (or of several closely related species), e.g., a gene promoter region G and the same promoter region G bound to a transcription factor T. The fact that M is conserved is thus interpreted as follows: if the group of species encoded by M is present in the system (i.e. at least one entity from M is present), then it will be present in the next state as well (possibly via different entities of M). Furthermore, elements from M are not produced by the system in the next state unless some entity of M is already present in the current state.

Note that mass conservation has been defined with respect to the support set so as to exclude elements of the background set which can only be provided via the context, with the intuition that such species would inevitably hinder the satisfaction of conservation properties for reaction systems. We define here a generalization of mass conservation that allows one to consider a different set of elements that can be reasonably excluded from the sets tested for conservation.

Definition 2.4 *(Parameterized conservation).* Let $\mathcal{A} = (S, A)$ be a reaction system and $T \subseteq S$ a set of species. A set $M \subseteq T$ is *conserved with respect to* T if, for any $W \subseteq T$, it holds that $M \cap W \neq \varnothing$ if and only if $M \cap \text{res}_{\mathcal{A}}(W) \neq \varnothing$. We use $\text{cons}(\mathcal{A}, T)$ to refer to all sets that are conserved with respect to T.

Note that the original definition of mass conservation (Definition 2.3) corresponds to parameterized conservation with respect to $T = \text{supp}(\mathcal{A})$.

Furthermore, it can be shown that in order to compute the conserved sets with respect to a given T we can, instead, find conserved sets with respect to the background set in a different RS. We start by defining the projection of a reaction system.

Definition 2.5. Let $\mathcal{A} = (S, A)$ be a reaction system and $T \subseteq S$ a set of species. For a reaction $a \in A$ that satisfies $R_a \subseteq T$, we define its *projection* onto T as $\text{proj}_T(a) = (R_a, I_a \cap T, P_a \cap T)$.

We define the *projection* of $\mathcal{A}$ onto T as $\text{proj}_T(\mathcal{A}) = (T, A')$ where:

$$A' = \{\text{proj}_T(a) \mid a \in A \wedge R_a \subseteq T\}$$

Lemma 2.1. *Let $\mathcal{A} = (S, A)$ be a reaction system, $T \subseteq S$ a set of species and $\mathcal{A}' = \text{proj}_T(\mathcal{A})$ the projection of $\mathcal{A}$ onto T. Then, for any set $W \subseteq T$, we have that* $\text{res}_{\mathcal{A}'}(W) = \text{res}_{\mathcal{A}}(W) \cap T$.

Proof. Consider an arbitrary reaction $a \in A$ and a set $W \subseteq T$. If $R_a \not\subseteq T$, then a has no corresponding reaction in $\mathcal{A}'$, since $\text{proj}_T(a)$ is not defined. But in this case note that a is not enabled for W in $\mathcal{A}$, since $W \subseteq T$ and $R_a \not\subseteq T$.

If $R_a \subseteq T$, then consider $a' = \text{proj}_T(a)$ and note that $en_a(W) = en_{a'}(W)$. Indeed, since $W \subseteq T$, we have that $W \cap I_a = \varnothing \Leftrightarrow W \cap (I_a \cap T) = \varnothing$. Since the reaction a' only produces $P_a \cap T$ rather than P_a, we obtain the desired result $\text{res}_{\mathcal{A}'}(W) = \text{res}_{\mathcal{A}}(W) \cap T$. □

Please cite this article in press as: S. Azimi et al., Dependency graphs and mass conservation in reaction systems, Theoret. Comput. Sci. (2015), http://dx.doi.org/10.1016/j.tcs.2015.02.014

Note that the meaning of Lemma 2.1 is that the projection of a reaction system onto a set T preserves the behavior of the result function with respect to T. This property enables us to reduce the problem of parameterized conservation to computing conserved sets with respect to the full background set.

Theorem 2.2. *For any reaction system $\mathcal{A} = (S, A)$ and any set of species $T \subseteq S$, we have that* $\text{cons}(\mathcal{A}, T) = \text{cons}(\text{proj}_T(\mathcal{A}), T)$.

Proof. Let $\mathcal{A}' = \text{proj}_T(\mathcal{A})$. Based on Lemma 2.1, we have that $M \cap \text{res}_{\mathcal{A}'}(W) = M \cap \text{res}_{\mathcal{A}}(W)$, for any $M \subseteq T$, so the conserved sets with respect to T are the same in the two reaction systems, i.e. $\text{cons}(\mathcal{A}, T) = \text{cons}(\mathcal{A}', T)$. □

Note that, for any reaction system $\mathcal{A} = (S, A)$, its projection onto a set $T \subseteq S$ uses T as background set. Based on this, we will consider, throughout the rest of this paper, only the problem of finding sets that are conserved with respect to the background set (denoted by $\text{cons}(\mathcal{A})$ instead of $\text{cons}(\mathcal{A}, S)$), but having all results implicitly applicable both for the parameterized conservation with respect to arbitrary sets T (by relying on Theorem 2.2) as well as for the original definition of mass conservation (by taking $T = \text{supp}(\mathcal{A})$ in Theorem 2.2).

It should be noted, though, that the projection $\mathcal{A}' = \text{proj}_T(\mathcal{A})$ may include reactions with empty inhibitor or product sets, even if $\mathcal{A}$ does not have them. Therefore, in this paper we remove the usual requirement that all the three sets defining reactions need to be nonempty [9,2]. This is in line with the observation that reactions that cannot be inhibited by the sets taken into consideration (e.g. subsets of the support set) are crucial for mass conservation [8].

Going back to Example 2.1, let us consider the projection of the reaction system onto its support set. This translates in this case to the removal of d_I from the inhibitor sets. We obtain $\text{proj}_{\text{supp}(\mathcal{A})}(\mathcal{A}) = (S', A')$, where:

$S' = \{$ hsf, hsp, hse, hsp:hsf, hsf:hse$\}$

$A' = \{$ ({hsf}, {hsp}, {hsf}),
({hsp, hsf}, {}, {hsp:hsf}),
({hsp:hsf}, {}, {hsp, hsf}),
({hse}, {hsf}, {hse}),
({hsf, hse}, {hsp}, {hsf:hse}),
({hsf:hse}, {hsp}, {hsf:hse, hsp}),
({hsp, hsf, hse}, {}, {hsp:hsf, hse}),
({hsp, hsf:hse}, {}, {hsp:hsf, hse})$\}$

Notice that indeed we now have empty inhibitor sets for some of the reactions. This is consistent with the use of d_I in [2] as a so called "dummy inhibitor" meant only to ensure compliance with the more restrictive version of the definition of reaction systems.

3. From mass conservation relations to dependency graphs

In this section we aim to gain a better understanding of mass conservation in reaction systems by relating it to the inner structure induced by reactions. We start by first translating the reactions to a graph that completely characterizes the behavior of the system.

Definition 3.1. Let $\mathcal{A} = (S, A)$ be a reaction system. The *behavior graph* of $\mathcal{A}$ is defined as $G_b = (V_b, E_b)$, with $V_b = 2^S$ and $E_b = \{(W, \text{res}_{\mathcal{A}}(W)) \mid W \subseteq S\}$.

Note that the behavior graph is in fact the *phase space* of the reaction system, i.e. it contains all the possible states that the system can be in, with edges denoting the transition from one state to another. Moreover, the behavior graph only encodes the result function $\text{res}_{\mathcal{A}}$. In particular, it is possible to have different reaction systems that translate to the same behavior graph (such systems are said to be functionally equivalent [1]). For the reaction system from Example 2.1 the behavior graph is depicted in Fig. 1.

Consider now a conserved set M. For any state W we have that M either intersects both W and $\text{res}_{\mathcal{A}}(W)$ or is disjoint from both of them. A similar property can be formulated for M with respect to the connected components of the behavior graph. Before showing how this can be achieved, we give several graph-theoretic definitions.

Definition 3.2. Let $G = (V, E)$ be a directed graph. We say that a node v is connected to a node u if there is a (possibly degenerate) undirected path from u to v in G. Connectedness defined in this way is an equivalence relation. We refer to its equivalence classes as *connected components* and we use $CC_G(u)$ to denote the connected component that contains u. Furthermore, we denote the set of all connected components of G by $CCS_G = \{CC_G(u) \mid u \in V\}$.

Please cite this article in press as: S. Azimi et al., Dependency graphs and mass conservation in reaction systems, Theoret. Comput. Sci. (2015), http://dx.doi.org/10.1016/j.tcs.2015.02.014

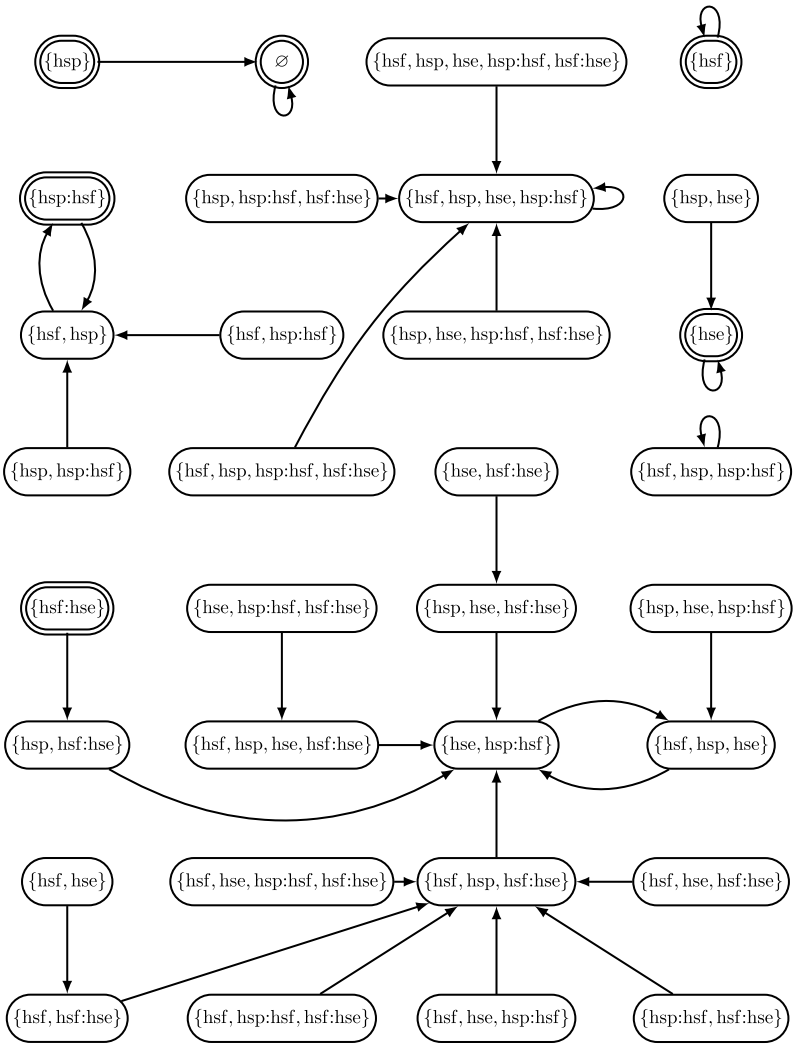


Fig. 1. Behavior graph for the HSR model (Example 2.1). Double line borders correspond to singleton sets and to the empty set.

Note that our notion of connected components differs slightly from its standard use in the sense that we only refer to sets of nodes instead of induced subgraphs. For a further introduction to graph theory, we refer the reader to [13].

In the following definition we introduce a formal notation for saying that a given set M intersects all (or none) of the sets from a collection of sets $\mathcal{C}$.

Definition 3.3. Let S be an arbitrary finite set and consider a set $M \subseteq S$ and a collection of sets $\mathcal{C} \subseteq 2^S$. We use the notation $M \sqcap \mathcal{C} = \{T \in \mathcal{C} \mid M \cap T \neq \varnothing\}$ to refer to the collection of those sets in $\mathcal{C}$ which intersect M.

We say that M *intersects* $\mathcal{C}$ if M intersects every element of $\mathcal{C}$, i.e. $M \sqcap \mathcal{C} = \mathcal{C}$. We say that M is *disjoint* from $\mathcal{C}$ if M is disjoint from every element of $\mathcal{C}$, i.e. $M \sqcap \mathcal{C} = \varnothing$. We say that M is *consistent* with $\mathcal{C}$ if M intersects $\mathcal{C}$ or is disjoint from $\mathcal{C}$.

Proposition 3.1. *Let $\mathcal{A} = (S, A)$ be a reaction system and $G_b = (V_b, E_b)$ its behavior graph. For any set $M \subseteq S$, the following two statements are equivalent:*

(1) *M is conserved,*
(2) *M is consistent with every connected component $\mathcal{C} \in CCS_{G_b}$.*

Proof. The implication (2) $\Rightarrow$ (1) follows from the fact that, for every set $W \subseteq S$, we have $\mathrm{res}_{\mathcal{A}}(W) \in CC_{G_b}(W)$.

We now prove that (1) $\Rightarrow$ (2). Let $\mathcal{C}$ be a connected component of the behavior graph G_b. Assume that M is a conserved set for which there exist $W_1, W_2 \in \mathcal{C}$ such that $M \cap W_1 = \varnothing$ and $M \cap W_2 \neq \varnothing$. Since W_1 and W_2 are in the same connected component, there is a path of nodes connecting them, i.e., there exist $V_1, \ldots, V_n$ such that $V_1 = W_1$, $V_n = W_2$ and, for all

i with $1 \leq i \leq n-1$, $(V_i, V_{i+1}) \in E$ or $(V_{i+1}, V_i) \in E$. But then, in both cases, it must be that $M \cap V_i \neq \varnothing \Leftrightarrow M \cap V_{i+1} \neq \varnothing$, so we can via transitivity conclude that $M \cap W_1 \neq \varnothing \Leftrightarrow M \cap W_2 \neq \varnothing$, which contradicts our assumption and completes the proof. □

Note that, by the result presented in Proposition 3.1, the conservation of a given set M only depends on the connected components of the behavior graph and not on its edges or their direction. This means that even fairly different reaction systems may end up having the same conserved sets or, in other words, equivalence with respect to conserved sets is a lot weaker than functional equivalence.

For the heat shock response model, one conserved set is $M = \{\text{hsf:hse}, \text{hse}\}$. To see that indeed this is the case, by Proposition 3.1, note in Fig. 1 that the connected components of $\varnothing$, {hsf}, {hsp:hsf} and {hsf, hsp, hsp:hsf} intersect M, whereas the other connected components of the behavior graph are disjoint from M.

3.1. Conservation dependency graph

In what follows we aim to further investigate the properties of conserved sets in relation with the connected components of the behavior graph.

Proposition 3.2. *Let $\mathcal{A} = (S, A)$ be a reaction system and $G_b = (V_b, E_b)$ its behavior graph. Consider an arbitrary element $x \in S$ and let $\mathcal{C}_x$ be the connected component that contains the singleton set $\{x\}$, i.e. $\mathcal{C}_x = CC_{G_b}(\{x\})$. Similarly, take $\mathcal{C}_\varnothing = CC_{G_b}(\varnothing)$. We denote, for any collection $\mathcal{C}$, $\text{cover}(\mathcal{C}) = \bigcup_{T \in \mathcal{C}} T$.*

(1) *If $x \in \text{cover}(\mathcal{C}_\varnothing)$, then x is not contained in any conserved set of $\mathcal{A}$, i.e. $\{x\} \sqcap \text{cons}(\mathcal{A}) = \varnothing$.*
(2) *If $\text{cover}(\mathcal{C}_x) = S$, then x is contained in all nonempty conserved sets of $\mathcal{A}$, i.e. $\{x\} \sqcap \text{cons}(\mathcal{A}) = \text{cons}(\mathcal{A}) \setminus \{\varnothing\}$.*
(3) *For every $y \in \text{cover}(\mathcal{C}_x)$ and for every conserved set M, if $x \notin M$, then $y \notin M$, (or, equivalently, $y \in M$ implies $x \in M$), i.e. $\{y\} \sqcap \text{cons}(\mathcal{A}) \subseteq \{x\} \sqcap \text{cons}(\mathcal{A})$.*

Proof. (1) Let M be an arbitrary conserved set. Then, from Proposition 3.1, it follows that M must be consistent with $\mathcal{C}_\varnothing$. Since $\varnothing \in \mathcal{C}_\varnothing$ and $M \cap \varnothing = \varnothing$, it must be that $M \sqcap \mathcal{C}_\varnothing = \varnothing$. In particular, we must also have $M \cap \{x\} = \varnothing$, which means that $x \notin M$.

(2) Let M be a nonempty conserved set. Then, since $M \cap S \neq \varnothing$, it must be that M intersects a set from $\mathcal{C}_x$. But from Proposition 3.1 we know that M must be consistent with $\mathcal{C}_x$ and, thus, it must be that $M \sqcap \mathcal{C}_x = \mathcal{C}_x$. In particular, we must also have $M \cap \{x\} \neq \varnothing$, which is equivalent to $x \in M$.

(3) Let M be a conserved set such that $x \notin M$. Then $M \cap \{x\} = \varnothing$ and, since M must be consistent with $\mathcal{C}_x$, it must be that $M \sqcap \mathcal{C}_x = \varnothing$, which implies that $M \cap \text{cover}(\mathcal{C}_x) = \varnothing$. In particular, this means that $y \notin M$. □

As we have seen, the conserved sets of a given RS only depend on the connected components of the behavior graph, i.e. on the partition induced by the reactions on the state space. In this context, Proposition 3.2 extracts properties of conserved sets by examining particular states and their connected components.

For example, the first two statements give us sufficient conditions for an element x to be in no conserved set, respectively in all nonempty conserved sets. Note that there is also an interesting interplay between the two statements when there exists an x such that $\text{cover}(\mathcal{C}_x) = S$ and $\varnothing \in \mathcal{C}_x$. Indeed, the latter is equivalent to having $\mathcal{C}_x = \mathcal{C}_\varnothing$, which means that $\text{cover}(\mathcal{C}_\varnothing) = S$, so no element of S can be part of a conserved set. On the other hand, the former property, $\text{cover}(\mathcal{C}_x) = S$, which translates to x being part of all nonempty conserved sets, is still (trivially) true since the only conserved set in this case is the empty set.

The more important implication of the previous remark is that, for the standard definition of reaction systems, where empty inhibitor sets are not allowed in reactions, there can be no nonempty conserved set at all. Indeed, for such reaction systems it holds that $\text{res}_\mathcal{A}(S) = \varnothing$, which leads to $\text{cover}(\mathcal{C}_\varnothing) = S$.

For the reaction system of Example 2.1, we have 7 connected components. In particular, note that $\mathcal{C}_\varnothing = \{\varnothing, \{\text{hsp}\}\}$, which means that hsp cannot be part of any conserved set. Furthermore, we have that $\text{cover}(\mathcal{C}_{\text{hsf:hse}})$ is the full background set, so hsf:hse will be part of every nonempty conserved set.

The third claim of Proposition 3.2 defines a dependency relation between the elements of the reaction system with respect to mass conservation. The statement implies that, for a pair of species (x, y) such that $y \in \text{cover}(\mathcal{C}_x)$, a conserved set that does not contain x cannot contain y or, equivalently, any conserved set that contains y must contain x as well. We can capture this dependency between species in a directed graph.

Definition 3.4. Let $\mathcal{A} = (S, A)$ be a reaction system and $G_b = (V_b, E_b)$ its behavior graph. The *conservation dependency graph* $G_{cd} = (V_{cd}, E_{cd})$ of $\mathcal{A}$ is given by $V_{cd} = S$ and $E_{cd} = \{(x, y) \mid x \in S \wedge y \in \text{cover}(\mathcal{C}_x)\}$.

Consider again our running example. Based on the connected components of singleton sets, highlighted with double borders in Fig. 1, we can compute the conservation dependency graph. The result is shown in Fig. 2.

Please cite this article in press as: S. Azimi et al., Dependency graphs and mass conservation in reaction systems, Theoret. Comput. Sci. (2015), http://dx.doi.org/10.1016/j.tcs.2015.02.014

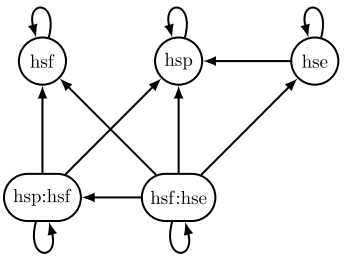


Fig. 2. Conservation dependency graph for the HSR model (Example 2.1).

Intuitively, every conserved set should satisfy all the constraints that are encoded by the conservation dependency graph. Alternatively, we can focus on the conservation dependency graph alone and consider all the sets that are consistent with the aforementioned constraints. In what follows, we capture the constraints of Proposition 3.2 (3), for arbitrary directed graphs, via the concept of source sets.

Definition 3.5. Let $G = (V, E)$ be a directed graph. A set $S \subseteq V$ is a *source set* of G if $E \cap (V \setminus S) \times S = \varnothing$, i.e. all edges of G that cross S (if any) do so from S to $V \setminus S$. We denote the set of all source sets of G by $\sigma(G)$.

It follows immediately from the definition that, for any graph $G = (V, E)$, both $\varnothing$ and V are source sets of G. The correspondence between the conserved sets of a reaction system and the source sets of its conservation dependency graph is given in Proposition 3.3.

Proposition 3.3. *Any conserved set M of a reaction system $\mathcal{A}$ is a source set in the conservation dependency graph of $\mathcal{A}$.*

Proof. The result follows from claim (3) of Proposition 3.2 and the definition of the conservation dependency graph. □

3.2. Computing the source sets of a directed graph

In this subsection we are concerned with the computation of source sets for general directed graphs. We start by investigating the interplay between source sets and the graph structure.

In what follows, we will say that a node u is an *ancestor* of a node v or, equivalently, that v is a *descendant* of u, if there exists a directed path from u to v.

Proposition 3.4. *Let $G = (V, E)$ be a directed graph and let S be an arbitrary source set of G.*

(1) *The parent of a node that is in S is also in S, i.e. for every two nodes u and v we have $v \in S \wedge (u, v) \in E \Rightarrow u \in S$.*
(2) *The child of a node that is not in S cannot be in S either, i.e. for every two nodes u and v we have $u \notin S \wedge (u, v) \in E \Rightarrow v \notin S$.*
(3) *The ancestor of a node that is in S is also in S.*
(4) *The descendant of a node that is not in S is not in S either.*

Proof. The negation of either (1) or (2) directly violates the definition of source sets by providing an edge (u, v) that goes from $V \setminus S$ to S. Furthermore, (3) and (4) can be proved by induction from (1) and (2), respectively. □

We are going to relate source sets to the strongly connected components of the graph under consideration.

Definition 3.6. Let $G = (V, E)$ be a directed graph. Two nodes $u, v \in V$ are said to be *strongly connected* if there exist in G a directed path from u to v and a directed path from v to u. Strong connectedness defined in this way is an equivalence relation. We refer to its equivalence classes as *strongly connected components* and use $SCC_G(u)$ to refer to the strongly connected component that contains u. Furthermore, we denote the set of all strongly connected components of G by $SCCS_G$, i.e. $SCCS_G = \{SCC_G(u) \mid u \in V\}$.

Note that, by definition, the strongly connected components of a directed graph are disjoint sets. It is not difficult to see by Proposition 3.4 that the source sets cannot split the strongly connected components of a graph.

Proposition 3.5. *Let $G = (V, E)$ be a directed graph, $C \in SCCS_G$ a strongly connected component of G and S a source set of G. If $C \cap S \neq \varnothing$, then $C \subseteq S$.*

Please cite this article in press as: S. Azimi et al., Dependency graphs and mass conservation in reaction systems, Theoret. Comput. Sci. (2015), http://dx.doi.org/10.1016/j.tcs.2015.02.014

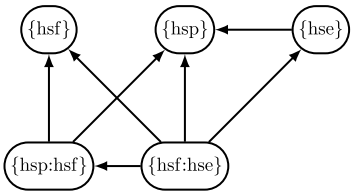


Fig. 3. Condensation of the conservation dependency graph for the HSR model (Example 2.1).

Proof. Choose $u \in C \cap S$. From Proposition 3.4 it follows that all ancestors of u must be in S as well. In particular, this implies that $C \subseteq S$. □

Corollary 3.6. *Any source set of a graph G is a union of (disjoint) strongly connected components of G.*

Proof. Let S be an arbitrary source set of G. For every $u \in S$, we have by Proposition 3.5 that $SCC_G(u) \subseteq S$, so we can write $S = \bigcup_{u \in S} SCC_G(u)$. □

Thus, we have seen that all source sets are unions of strongly connected components. In order to see exactly which of such unions are source sets, we will refer to the condensation of G, i.e. the graph obtained by replacing each strongly connected component of G with a single node.

Definition 3.7. Let $G = (V, E)$ be a directed graph. The *condensation* of G is the directed graph $\widetilde{G} = (\widetilde{V}, \widetilde{E})$ whose nodes are the strongly connected components of G, i.e. $\widetilde{V} = SCCS_G$, and whose edges are defined as follows: $\widetilde{E} = \{(U, W) \in \widetilde{V} \times \widetilde{V} \mid U \neq W \wedge \exists u \in U, \exists w \in W, (u, w) \in E\}$, i.e. there is an edge (U, W) in $\widetilde{G}$ iff there is an edge in G from an element of U to an element of W.

Note that the condensation graph is a directed acyclic graph (DAG). We illustrate the concept by showing, in Fig. 3, the condensation of the conservation dependency graph for our running example. Note that for this particular case the condensation is almost identical to the original graph, only the nodes correspond now to singleton sets and the self-loops are no longer present. This very close similarity is due to the fact that the conservation dependency graph, shown in Fig. 2, does not contain any cycles aside from self-loops.

Proposition 3.7. *Let $G = (V, E)$ be a directed graph and $\widetilde{G} = (\widetilde{V}, \widetilde{E})$ its condensation. A set $S \subseteq V$ is a source set of G iff there exists a set $\widetilde{S} \subseteq \widetilde{V}$ such that $S = \text{cover}(\widetilde{S})$ and $\widetilde{S}$ is a source set of $\widetilde{G}$.*

Proof. We start with the forward implication. We know already from Corollary 3.6 that there exists $\widetilde{S} \subseteq \widetilde{V}$ such that $S = \text{cover}(\widetilde{S})$. Assume that $\widetilde{S}$ is not a source set in $\widetilde{G}$. Then there exist $U \in \widetilde{S}$ and $W \in \widetilde{V} \setminus \widetilde{S}$ such that $(W, U) \in \widetilde{E}$, which means that there exist $u \in U$ and $w \in W$ such that $(w, u) \in E$. But this contradicts the fact that S is a source set, since $u \in S$ and $w \in V \setminus S$. Thus, it must be that $\widetilde{S}$ is a source set of $\widetilde{G}$.

For the reverse implication, consider a source set $\widetilde{S}$ of the condensation graph $\widetilde{G}$ and let $S = \text{cover}(\widetilde{S})$. Assume that S is not a source set of G. Then there exist two nodes $u \in S$ and $w \in V \setminus S$ such that $(w, u) \in E$. Since strongly connected components are either fully contained in a source set or disjoint from it (by Proposition 3.5), it must be that $SCC_G(u) \in \widetilde{S}$ and $SCC_G(w) \in \widetilde{V} \setminus \widetilde{S}$. Furthermore, since $(w, u) \in E$, we have $(SCC_G(w), SCC_G(u)) \in \widetilde{E}$, which contradicts the fact that $\widetilde{S}$ is a source set of $\widetilde{G}$. Thus, it must be that S is a source set of G. □

The practical conclusion we can draw from Proposition 3.7 is that it suffices to have an algorithm for computing the source sets of directed acyclic graphs (DAG's) and use it on the condensation graph.

In what follows, we will use $G{\downarrow_S}$ to denote the restriction of the graph $G = (V, E)$ to a subset of nodes $S \subseteq V$, i.e. $G{\downarrow_S} = (S, E \cap (S \times S))$. We will also use $\text{desc}_G(S)$ to refer to the set containing all the nodes from S and all their descendants. Similarly, we will use $\text{anc}_G(S)$ to refer to the set containing all nodes from S and all their ancestors in G.

Theorem 3.8. *Let $G = (V, E)$ be a directed graph, $T \subseteq V$ an arbitrary set of nodes and $s \in V$ an arbitrary node from G.*

(1) *A source set S includes a set T if and only if $S \setminus \text{anc}_G(T)$ is a source set of $G{\downarrow_{V \setminus \text{anc}_G(T)}}$ and, in addition, S contains all the ancestors of elements from T:*

$$S \in \sigma(G) \wedge T \subseteq S \Leftrightarrow S \setminus \text{anc}_G(T) \in \sigma(G{\downarrow_{V \setminus \text{anc}_G(T)}}) \wedge \text{anc}_G(T) \subseteq S.$$

Please cite this article in press as: S. Azimi et al., Dependency graphs and mass conservation in reaction systems, Theoret. Comput. Sci. (2015), http://dx.doi.org/10.1016/j.tcs.2015.02.014

(2) *A source set S does not intersect a set T if and only if S is a source set in the graph obtained from G by removing all elements of T and their descendants:*

$$S \in \sigma(G) \wedge S \cap T = \varnothing \Leftrightarrow S \in \sigma(G{\downarrow}_{V \setminus \mathrm{desc}_G(T)}).$$

(3) *Given a node s, all source sets of G can be computed recursively by relying on subgraphs of G that do not contain s:*

$$\sigma(G) = \sigma(G{\downarrow}_{V \setminus \mathrm{desc}_G(\{s\})}) \cup \{S \cup \mathrm{anc}_G(\{s\}) \mid S \in \sigma(G{\downarrow}_{V \setminus \mathrm{anc}_G(\{s\})})\}.$$

Proof. (1) We have:

$$\begin{aligned}
& S \in \sigma(G) \wedge T \subseteq S \\
\Leftrightarrow\ & E \cap (V \setminus S) \times S = \varnothing \wedge \mathrm{anc}_G(T) \subseteq S \\
\Leftrightarrow\ & E \cap (V \setminus S) \times (S \setminus \mathrm{anc}_G(T)) = \varnothing \wedge \mathrm{anc}_G(T) \subseteq S \\
\Leftrightarrow\ & E \cap \big((V \setminus \mathrm{anc}_G(T)) \setminus (S \setminus \mathrm{anc}_G(T))\big) \times (S \setminus \mathrm{anc}_G(T)) = \varnothing \wedge \mathrm{anc}_G(T) \subseteq S \\
\Leftrightarrow\ & S \setminus \mathrm{anc}_G(T) \in \sigma(G{\downarrow}_{V \setminus \mathrm{anc}_G(T)}) \wedge \mathrm{anc}_G(T) \subseteq S.
\end{aligned}$$

The first equivalence follows from the definition of conserved sets and Proposition 3.4. The second one follows from the definition of $\mathrm{anc}_G(T)$, as there can be no edges of G going into this set. The third equivalence relies on the relation $A \setminus B = (A \setminus X) \setminus (B \setminus X)$, which holds whenever $X \subseteq B \subseteq A$. Finally, the last step is a direct application of the definition of source sets.

(2) We follow a similar approach and we have:

$$\begin{aligned}
& S \in \sigma(G) \wedge S \cap T = \varnothing \\
\Leftrightarrow\ & E \cap (V \setminus S) \times S = \varnothing \wedge S \cap \mathrm{desc}_G(T) = \varnothing \\
\Leftrightarrow\ & E \cap ((V \setminus \mathrm{desc}_G(T)) \setminus S) \times S = \varnothing \wedge S \cap \mathrm{desc}_G(T) = \varnothing \\
\Leftrightarrow\ & S \in \sigma(G{\downarrow}_{V \setminus \mathrm{desc}_G(T)}).
\end{aligned}$$

Just as before, the first equivalence follows directly from the definition of source sets and from Proposition 3.4. The second equivalence relies on the definition of $\mathrm{desc}_G(T)$, as there can be no edges of G going out of this set. Finally, we use the definition again to get the desired result.

(3) The result follows from (1) and (2) by noting that we can partition the source sets of G into those that contain s and those that do not contain it. We can thus write:

$$\begin{aligned}
& S \in \sigma(G) \wedge s \in S \Leftrightarrow S \setminus \mathrm{anc}_G(\{s\}) \in \sigma(G{\downarrow}_{V \setminus \mathrm{anc}_G(\{s\})}) \wedge \mathrm{anc}_G(\{s\}) \subseteq S, \\
& S \in \sigma(G) \wedge s \notin S \Leftrightarrow S \in \sigma(G{\downarrow}_{V \setminus \mathrm{desc}_G(\{s\})}).
\end{aligned}$$

These statements lead to the desired result. □

We can immediately apply the third claim of Theorem 3.8 to a source node of G (a node with no parents) and write an even simpler decomposition of the source sets of G into two parts.

Corollary 3.9. *Let $G = (V, E)$ be a directed graph and let $s \in V$ be a source node. Then we have:*

$$\sigma(G) = \sigma(G{\downarrow}_{V \setminus \mathrm{desc}_G(\{s\})}) \cup \{S \cup \{s\} \mid S \in \sigma(G{\downarrow}_{V \setminus \{s\}})\}.$$

We can translate the previous formal result into an algorithm for computing the source sets of a directed acyclic graph.

Algorithm 3.1 *(Source sets of a DAG).* Let $G = (V, E)$ be a DAG. If the graph contains no nodes, return the empty set as the only source set. Otherwise choose a source node $s \in V$, compute the source sets of $G{\downarrow}_{V \setminus \{s\}}$ and $G{\downarrow}_{V \setminus \mathrm{desc}_G(\{s\})}$, then aggregate them according to Corollary 3.9 to obtain the source sets of G.

Note that the fact that the graph is acyclic is required for the existence of the source node s.

4. Enumerating the conserved sets of a reaction system

In this section we propose and discuss the advantages of an algorithm that relies on the conservation dependency graph to list all the conserved sets of a given reaction system.

Please cite this article in press as: S. Azimi et al., Dependency graphs and mass conservation in reaction systems, Theoret. Comput. Sci. (2015), http://dx.doi.org/10.1016/j.tcs.2015.02.014

4.1. An algorithm for enumerating all conserved sets

We provide here an algorithm for listing all conserved sets of a reaction system. The actual test for conservation relies on Proposition 3.1, whereas the candidate sets to be tested are computed using Algorithm 3.1, as well as the additional information coming from Proposition 3.2.

Algorithm 4.1 *(Compute all conserved sets).* Let $\mathcal{A} = (S, A)$ be a reaction system.

1. Compute the behavior graph G_b of $\mathcal{A}$.
2. Compute the connected components of G_b and collect the following information:
   (a) compute $P_{out} = \text{cover}(\mathcal{C}_\varnothing)$,
   (b) compute $P_{in} = \{x \in S \mid \text{cover}(C_x) = S\}$.
3. Compute the conservation dependency graph G_{cd} of $\mathcal{A}$.
4. Compute the strongly connected components of G_{cd} and the condensation graph $\widetilde{G}_{cd}$. Collect the following additional information:
   (a) compute the set $\widetilde{P}_{out}$ of nodes from $\widetilde{G}_{cd}$ which contain elements of P_{out}, i.e. $\widetilde{P}_{out} = \{SCC_{G_{cd}}(x) \mid x \in P_{out}\}$,
   (b) compute the set $\widetilde{P}_{in}$ of nodes from $\widetilde{G}_{cd}$ which contain elements of P_{in}, i.e. $\widetilde{P}_{in} = \{SCC_{G_{cd}}(x) \mid x \in P_{in}\}$.
5. Compute the reduced condensation graph $\widetilde{G}'_{cd}$ by removing from $\widetilde{G}_{cd}$ all nodes from $\widetilde{P}_{out}$ with their descendants and all nodes from $\widetilde{P}_{in}$ with their ancestors:
   (a) compute the set $\widetilde{P}'_{out} = \text{desc}_{\widetilde{G}_{cd}}(\widetilde{P}_{out})$,
   (b) compute the set $\widetilde{P}'_{in} = \text{anc}_{\widetilde{G}_{cd}}(\widetilde{P}_{in})$,
   (c) compute the reduced condensation graph $\widetilde{G}'_{cd} = \widetilde{G}_{cd}{\downarrow}_{\widetilde{V}_{cd} \setminus (\widetilde{P}'_{out} \cup \widetilde{P}'_{in})}$.
6. Compute the source sets of $\widetilde{G}'_{cd}$ using Algorithm 3.1.
7. For each source set $\widetilde{T} \in \sigma(\widetilde{G}'_{cd})$, compute the corresponding source set of G_{cd} as $T = \text{cover}(\widetilde{T} \cup \widetilde{P}'_{in})$.
8. For each source set T computed in step 7, test whether T is a conserved set by checking that it is consistent with all connected components of the behavior graph G_b.
9. Add $\varnothing$ to the list of conserved sets, if it was not obtained in the previous step.

Theorem 4.1. *Algorithm 4.1 computes all conserved sets correctly.*

Proof. First, let us see that steps 1–3 compute the data structures required for the rest of the algorithm, i.e. P_{in}, P_{out} and G_{cd}. Furthermore, steps 4–6 translate to computing exactly the source sets of G_{cd} which include P_{in} and are disjoint from P_{out}. Indeed, in order to find all the source sets which are disjoint from P_{out}, we can rely on Proposition 3.7 to conclude that we also need to exclude the full strongly connected components of elements from P_{out}, then based on Theorem 3.8(2) we must also exclude descendant strongly connected components. This translates to steps 4(a) and 5(a) of the algorithm, respectively. A similar justification holds for steps 4(b) and 5(b).

Now note that this algorithm relies on testing for conservation using Proposition 3.1, but only examines a reduced set of candidates by relying on Proposition 3.2 and Proposition 3.3.

The last step is required in order to ensure that the empty set is also included in the list of conserved sets since, whenever $P_{in} \neq \varnothing$, all the candidate sets tested in step 8 are nonempty. □

Remark that the decision problem for conserved sets is coNP-complete [8]. As such, we know already that we cannot test for conservation in polynomial time unless P = NP. On the other hand, we focus here on finding all conserved sets, which means that we can make use of aggregate information from the original reaction system in order to speed up the test for conservation. In particular, once we have the connected components of the behavior graph, we can simply forget about the reactions. Moreover, the analysis of the connected components of the empty set and singleton sets, together with the constraints encoded in the conservation dependency graph, enable us to reduce the actual number of candidates that we need to verify.

4.2. Efficiency of the algorithm

To understand the benefit of the strategy employed in Algorithm 4.1 and also the nature of the reactions systems for which it is efficient, we discuss in this subsection several examples.

We first consider the running example of the previous sections, the HSR model from Example 2.1. Recall that we have already seen the behavior graph in Fig. 1, the conservation dependency graph in Fig. 2, and the condensation of the conservation dependency graph in Fig. 3. These graphs correspond to the execution of steps 1, 3 and 4 of the algorithm, respectively.

The result for step 2 is that $P_{out} = \{\text{hsp}\}$ and $P_{in} = \{\text{hsf:hse}\}$. Thus, when running step 5(a), we remove node {hsp} from the condensation graph (there are no descendants to remove in this case). Similarly, node {hsf:hse} is removed in step 5(b). The resulting graph is presented in Fig. 4.

Please cite this article in press as: S. Azimi et al., Dependency graphs and mass conservation in reaction systems, Theoret. Comput. Sci. (2015), http://dx.doi.org/10.1016/j.tcs.2015.02.014

Fig. 4. Reduced condensation graph for the HSR model (Example 2.1).

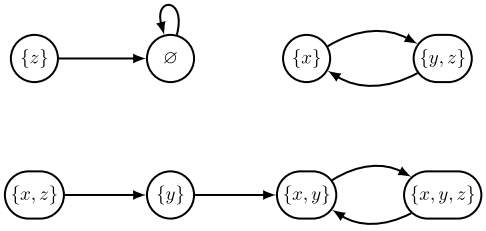


Fig. 5. Behavior graph for the reaction system from Example 4.1.

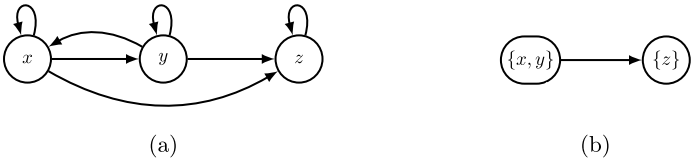


Fig. 6. (a) Conservation dependency graph and (b) its condensation, for the reaction system from Example 4.1.

The reduced condensation graph has six source sets, which are shown below, together with the corresponding source sets of the original conservation dependency graph.

$\widetilde{T}_1 = \varnothing$ $T_1 = \{\text{hsf:hse}\}$
$\widetilde{T}_2 = \{\{\text{hsp:hsf}\}\}$ $T_2 = \{\text{hsf:hse, hsp:hsf}\}$
$\widetilde{T}_3 = \{\{\text{hsp:hsf}\}, \{\text{hsf}\}\}$ $T_3 = \{\text{hsf:hse, hsp:hsf, hsf}\}$
$\widetilde{T}_4 = \{\{\text{hse}\}\}$ $T_4 = \{\text{hsf:hse, hse}\}$
$\widetilde{T}_5 = \{\{\text{hsp:hsf}\}, \{\text{hse}\}\}$ $T_5 = \{\text{hsf:hse, hsp:hsf, hse}\}$
$\widetilde{T}_6 = \{\{\text{hsp:hsf}\}, \{\text{hsf}\}, \{\text{hse}\}\}$ $T_6 = \{\text{hsf:hse, hsp:hsf, hsf, hse}\}$

Of the six source sets of the conservation dependency graph, only T_3, T_4 and T_6 are conserved. Note that T_3 corresponds to all the forms (free and occupied) of gene promoters (hse). Similarly, T_4 corresponds to the conservation of the total amount of heat shock factors (free, bound to the promoter or bound to the heat shock protein). The third conserved set, T_6, is in fact the union of the other two, so it does not bring additional information, since it is in general the case that the union of two conserved sets is also conserved.

Note that, out of the 32 possible sets, the algorithm enabled us to only test 6 candidates for conservation.

Example 4.1. Consider the following reaction system:

$$S = \{x, y, z\};$$

$$A = \{(\{x\}, \{z\}, \{y, z\}), (\{y\}, \varnothing, \{x\}), (\{x, z\}, \varnothing, \{y\}), (\{y\}, \{z\}, \{y\})\}\,.$$

The corresponding behavior graph G_b can be computed by finding $\text{res}_{\mathcal{A}}(W)$ for all $W \subseteq S$. The graph is illustrated in Fig. 5.

The relevant connected components for our algorithm are:

$$\mathcal{C}_x = \{\{x\}, \{y, z\}\};$$

$$\mathcal{C}_y = \{\{y\}, \{x, y\}, \{x, z\}, \{x, y, z\}\};$$

$$\mathcal{C}_z = \{\{z\}, \varnothing\} = \mathcal{C}_\varnothing.$$

The corresponding conservation dependency graph is presented in Fig. 6a.

In step 5 of the algorithm we rely on $P_{out} = \{z\}$ and $P_{in} = \{x, y\}$ (computed in steps $2(a)$ and $2(b)$, respectively) to obtain that all nonempty conserved sets must contain x and y and cannot contain z, which means that $M = \{x, y\}$ is the only nonempty conserved set for this reaction system. In particular, note that the algorithm will work on an empty graph at step 6.

Please cite this article in press as: S. Azimi et al., Dependency graphs and mass conservation in reaction systems, Theoret. Comput. Sci. (2015), http://dx.doi.org/10.1016/j.tcs.2015.02.014

Furthermore, let us see that in this case the conservation dependency graph has only three source sets, namely $\varnothing$, $\{x, y\}$ and S. Thus, even if we are to consider the full condensation of the conservation dependency graph, i.e. skip step 5 of the algorithm, we would still obtain a significant improvement since we need to examine only 3 out of the total number of 8 candidates.

While the previous example is rather simple, it still reveals the main improvements that come from the approach. On the other hand, consider also the case where the behavior and conservation dependency graphs do not provide any useful information, i.e. $P_{out} = \varnothing$, $P_{in} = \varnothing$ and G_{cd} only has self-loops for each node, meaning that all sets are source sets.

Example 4.2. Consider the reaction system $\mathcal{A} = (S, A)$ given by:

$$S = \{x_1, x_2, \ldots, x_n\}$$
$$A = \{(\{x_i\}, \varnothing, \{x_i\}) \mid x_i \in S\}$$

In this case the result function satisfies $\mathrm{res}_{\mathcal{A}}(W) = W$ for all states $W \subseteq S$, i.e. all states are isolated and connected components contain a single state. In particular, $\mathcal{C}_{\varnothing} = \{\varnothing\}$ and $\mathcal{C}_{x_i} = \{\{x_i\}\}$ for all $x_i \in S$.

Thus, for this example we have $P_{out} = P_{in} = \varnothing$ and the conservation dependency graph has only self-loop edges $E_{cd} = \{(x_i, x_i) \mid x_i \in S\}$. This means that every subset of S is a source set, i.e. we need to examine all candidates. But in this case note that in fact all subsets of S are conserved.

Example 4.3. Consider the reaction system $\mathcal{A} = (S, A)$ given by:

$$S = \{x_1, x_2, \ldots, x_n\}$$
$$A = \{(\{x_i\}, S \setminus \{x_i\}, \{x_i\}) \mid x_i \in S\}$$
$$\cup \{(\{x_i, x_j\}, \varnothing, S) \mid x_i, x_j \in S \wedge x_i \neq x_j\}$$

The result function $\mathrm{res}_{\mathcal{A}}$ satisfies:

$$\mathrm{res}_{\mathcal{A}}(W) = \begin{cases} W, & \text{if } |W| \leq 1\,, \\ S, & \text{if } |W| \geq 2\,. \end{cases}$$

Just as in Example 4.2, this reaction system does not give useful information for reducing the number of candidates examined in Algorithm 4.1. Instead, we analyze the behavior graph in relation to Proposition 3.1.

Based on the result function, the behavior graph has $n + 2$ connected components in this case: $\mathcal{C}_{\varnothing} = \{\varnothing\}$, $\mathcal{C}_{x_i} = \{\{x_i\}\}$ for each $x_i \in S$, and one connected component containing all the other states, call it $\mathcal{C}_S$.

We know that any conserved set M must be consistent with all connected components of the behavior graph. For CC's that contain a single set, this holds trivially, so we only need to worry about $\mathcal{C}_S$. The empty set is always conserved, so we focus on $M \neq \varnothing$. Then $M \cap S \neq \varnothing$, so it must be that $M \sqcap \mathcal{C}_S = \mathcal{C}_S$, i.e. M intersects all the elements of $\mathcal{C}_S$.

If more than two elements of S are missing from M, then we can find x_i, x_j such that $M \cap \{x_i, x_j\} = \varnothing$, but $\mathrm{res}_{\mathcal{A}}(\{x_i, x_j\}) = S$, so M is not conserved. If at most one element is missing, on the other hand, M is consistent with $\mathcal{C}_S$ and, thus, it is conserved.

Therefore, for this reaction system, the conserved sets are

$$\mathrm{cons}(\mathcal{A}) = \{\varnothing, S\} \cup \{S \setminus \{x_i\} \mid x_i \in S\},$$

for a total of $n + 2$ sets. However, the number of candidates that we need to examine is 2^n.

Note that in Example 4.2 and Example 4.3 we end up examining all possible states, but there is a fundamental difference between the two. While for the former we actually do need to examine all sets since all are conserved, for the latter the number of conserved sets is $n + 2$ out of the 2^n candidates. On the other hand, remark that the reaction system from Example 4.3 has a number of reactions quadratic in the number of species.

In what follows we aim to characterize the improvement provided by our algorithm over the naive approach. Let us first note that the test for conservation goes through the full behavior graph. Since the number of states of the behavior graph is exponential with respect to the number of species in the reaction system, the running time of our algorithm is not polynomially bounded. Moreover, we have seen in Example 4.2 that it is possible to have an exponential number of conserved sets for a given reaction system. Thus, the number of candidates that are tested for conservation is not polynomially bounded either. Therefore, we are going to assess the "quality" of our algorithm with respect to the number of non-conserved source sets, i.e. the candidate sets which are tested in addition to the conserved sets.

In the general case, one reaction (R_a, I_a, P_a) is enabled for all states W such that $R_a \subseteq W \subseteq S \setminus I_a$. Thus, a reaction that involves only a few species as reactants or inhibitors will have an impact on a significant subset of the edges of the behavior graph. Put differently, the edges of G_b are strongly interrelated and breaking this interdependence, to decouple for example

Please cite this article in press as: S. Azimi et al., Dependency graphs and mass conservation in reaction systems, Theoret. Comput. Sci. (2015), http://dx.doi.org/10.1016/j.tcs.2015.02.014

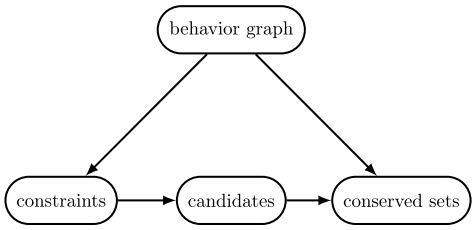


Fig. 7. Schematic flowchart of Algorithm 4.1.

the singleton states from the rest of the graph, requires an increased number of reactions. Thus, we expect that reaction systems for which the number of reactions is linear in the number of species will reveal enough structural information in the conservation dependency graph so that the number of source sets is very close to the number of actual conserved sets. We formulate this idea as a conjecture below.

Conjecture. *Increasing the number of reactions linearly with respect to the number of species, the number of non-conserved source sets increases at most polynomially with respect to the number of species.*

4.3. Flexibility of the algorithm

Testing whether a set is conserved or not is a coNP-complete problem (see, e.g., [8]). Thus, even with our conjectured expectation that for a class of reaction systems we only need to examine a polynomial number of extra candidates, it is still likely that larger reaction systems will render the application of our algorithm infeasible. However, we show in this subsection that Algorithm 4.1 can be customized to take advantage of available resources, as well as additional knowledge about the reaction system or, more generally, about the model that the RS encodes.

First, let us notice that the computations which identify the actual conserved sets happen at the very end of the algorithm, essentially in step 8, relying on the already computed set of candidates and on Proposition 3.1 for the conservation test. For running the test, in addition to the candidates, we also need the behavior graph (more precisely, its connected components).

The computation of the candidates covers steps 4 to 7. This part of the algorithm uses the sets P_{out} and P_{in}, and the conservation dependency graph G_{cd} as inputs. The output consists of the source sets of G_{cd} which include P_{in} and do not intersect P_{out}. Another way to interpret this output is to consider that all three inputs are in fact sets of constraints and the obtained candidates are simply the sets of species that satisfy all the input constraints:

- P_{out} contains species that should not be part of any candidate set;
- P_{in} contains species that should be present in every candidate set;
- G_{cd} encodes constraints of the form "if x is in, then so is y" or, alternatively, "if y is not in, then neither is x".

For example, it only matters that we add to P_{in} species that are included in all conserved sets (or, more precisely, in all conserved sets that we are interested to compute).

Steps 2 and 3 of the algorithm compute the input constraints based on the behavior graph and rely on Proposition 3.2. The flow of the algorithm is schematically summarized in Fig. 7, where each block (also referred to as *module*) is identified by its output and incoming edges stand for inputs.

Note from the diagram that the behavior graph is required not only for computing the constraints to be satisfied by the candidate sets, but also for the conservation test. Thus, if the size of the reaction system is such that storing the behavior graph is impractical, we need to provide alternative implementations for the two modules. The test for conservation can rely on running the reaction system for each state, instead of the connected components of the behavior graph. Furthermore, we can also make use of the polynomial criterion from the next section in order to obtain a negative answer faster for the candidates. Moreover, note that each candidate set is tested separately, so this module is highly parallelizable.

For the constraints part, we can also rely on running the reaction system. Since we are under the assumption that the behavior graph is too large to fit in working memory, it may also be the case that it is not feasible to run the RS from an initial state up to identifying the first repeating state. Nevertheless, even by running the system a limited number of steps from particular initial states, we can partially recover the constraints that were obtained in Proposition 3.2 from the connected components of the behavior graph:

- any element of states that are reachable from $\varnothing$ will be in P_{out};
- any singleton state reachable from S will be included in P_{in};
- elements y of states reachable from singletons $\{x\}$ will contribute edges (x, y) for G_{cd}.

Please cite this article in press as: S. Azimi et al., Dependency graphs and mass conservation in reaction systems, Theoret. Comput. Sci. (2015), http://dx.doi.org/10.1016/j.tcs.2015.02.014

Note that, in addition to the constraints derived from the behavior graph (or by running the reaction system as suggested above), we can include other constraints as well, either by exploiting the particular structure of the reaction system under consideration, or by relying on available knowledge about the model that is described by the RS. For example, if we know a priori that a particular element x should be in all conserved sets (or, alternatively, if we are only interested to find those conserved sets that contain x) we can add this constraint by including x in P_{in}.

As long as all additional constraints that we put in are still provably satisfied by any conserved set of the input reaction system, we are going to obtain the full list of conserved sets as the output of the algorithm. If extra constraints are added which come from the biological model or simply from the need to speed up the algorithm, then we will only obtain a subset of the conserved sets, namely those which satisfy all the constraints.

5. Negative polynomial heuristics for formula correspondence

In this section we give a simple polynomial (in size of the formulae and number of reactions) heuristics which can help decide whether a given set M is not conserved. The provided heuristics will be sufficient, but *not* necessary. We will provide two negative criteria for a more general problem first, and then show how they can be applied to mass conservation.

Note that, since deciding whether a given set of species M is conserved in a reaction system is coNP-complete [8], we could not expect to give such a polynomial criterion which would be both sufficient *and* necessary. This section shows, nevertheless, that analyzing some static properties of the reaction system may help conclude that M is not conserved in polynomial time, without enumerating all subsets of species.

We recall first that a *Boolean formula* φ is said to be over an alphabet S if all its variable names are from S. In the following we assume all Boolean formulae to be given in a disjunctive normal form. A subset $W \subseteq S$ is said to *satisfy* the Boolean formula φ over S if the expression for φ contains a conjunction $x_1 \wedge \ldots \wedge x_n \wedge \bar{y}_1 \wedge \ldots \wedge \bar{y}_m$ such that

(1) $\{x_i \mid 1 \leq i \leq n\} \subseteq W$, and
(2) $\{y_j \mid 1 \leq i \leq m\} \cap W = \varnothing$.

By convention, we write $\varphi(W) = 1$, or simply $\varphi(W)$, if the subset W satisfies φ, and $\varphi(W) = 0$ otherwise. For more details about the relationship between reaction systems and Boolean functions we refer to [2] and [1].

The paper [8] generalizes mass conservation in the form of two *formula correspondence problems*. Given a reaction system $\mathcal{A} = (S, A)$ and two Boolean formulae ϕ and ψ over S, the formula correspondence problems consist in deciding whether the following relations hold for every set $W \subseteq \mathrm{supp}(\mathcal{A})$:

$$\phi(W) \Rightarrow \psi(\mathrm{res}_{\mathcal{A}}(W)),$$
$$\phi(W) \Leftrightarrow \psi(\mathrm{res}_{\mathcal{A}}(W)).$$

It is shown in [8] that deciding either of these questions is coNP-complete.

We can parameterize the formula correspondence problems for a subset T of the background set in the same way as we parameterized mass conservation in Section 2. In such a case, we would define the formulae ϕ and ψ over T, and would require $\phi(W) \Rightarrow \psi(\mathrm{res}_A(W))$ (respectively, $\phi(W) \Leftrightarrow \psi(\mathrm{res}_A(W))$) for all subsets W of T, instead of the support of $\mathcal{A}$. It turns out that, just as with parameterized mass conservation, checking formula correspondence against a subset $T \subseteq S$ can be reduced to testing the same formulae against the projection $\mathcal{A}' = \mathrm{proj}_T(\mathcal{A})$. Indeed, recall that, for any set $W \subseteq T$, we have $\mathrm{res}_{\mathcal{A}'}(W) = \mathrm{res}_{\mathcal{A}}(W) \cap T$. Since ψ is over T as well, the elements from the potentially nonempty $\mathrm{res}_{\mathcal{A}}(W) \setminus T$ will have no influence upon the satisfiability of ψ, i.e. $\psi(\mathrm{res}_{\mathcal{A}}(W)) \Leftrightarrow \psi(\mathrm{res}_{\mathcal{A}}(W) \cap T)$. This implies that formula correspondence in $\mathcal{A}$ with respect to T holds if and only if it holds in $\mathcal{A}'$ with respect to its full background set.

Seeing that conventional formula correspondence can be expressed as parameterized correspondence over $T = \mathrm{supp}(\mathcal{A})$ is a matter of remarking that ϕ and ψ in the conventional formulation can be restricted to the alphabet $\mathrm{supp}(\mathcal{A})$, without losing generality. Indeed, having ϕ include $\bar{x}$, with $x \in S \setminus \mathrm{supp}(\mathcal{A})$, for example, is redundant since we are only checking the correspondence against the subsets of $\mathrm{supp}(\mathcal{A})$ anyway. If, on the other hand, ϕ employs x in its non-negated form, then no subset of $\mathrm{supp}(\mathcal{A})$ will satisfy ϕ. Similar arguments can be given for ψ and the result set $\mathrm{res}_{\mathcal{A}}(W)$.

In view of the fact that any case of parameterized formula correspondence, and, in particular, the conventional correspondence, is reducible to formula correspondence over the full background set, we only focus on the latter problem.

For a given conjunction $\phi_1 = x_1 \wedge \ldots \wedge x_n \wedge \bar{y}_1 \wedge \ldots \wedge \bar{y}_m$, we will use the following shortcut notations:

$$\mathrm{pos}(\phi_1) = \{x_1, \ldots, x_n\},$$
$$\mathrm{neg}(\phi_1) = \{y_1, \ldots, y_n\}.$$

Suppose now that the first formula ϕ over S is given in a disjunctive normal form, $\phi = \bigvee_{i=1}^{n} \phi_i$, and consider a reaction $a = (R_a, I_a, P_a)$ over the same alphabet S. We would like to know the conditions for a to be enabled on at least one subset satisfying ϕ.

Please cite this article in press as: S. Azimi et al., Dependency graphs and mass conservation in reaction systems, Theoret. Comput. Sci. (2015), http://dx.doi.org/10.1016/j.tcs.2015.02.014

Lemma 5.1. *For a reaction system $\mathcal{A} = (A, S)$, a reaction $a = (R_a, I_a, P_a) \in A$, and a Boolean formula $\phi = \bigvee_{i=1}^{n} \phi_i$, both over the same alphabet S, the following conditions are equivalent:*

(1) $\exists W \subseteq S, \phi(W) \wedge \mathrm{en}_a(W)$, *and*
(2) $\exists i \in \{1, \ldots, n\}, R_a \cap \mathrm{neg}(\phi_i) = I_a \cap \mathrm{pos}(\phi_i) = \varnothing$.

Proof. *(1)⇒(2):* Suppose there exists a subset W which both satisfies ϕ and enables a. This means that $R_a \subseteq W$ and $I_a \cap W = \varnothing$, but also that ϕ contains a conjunction ϕ_i such that $\mathrm{pos}(\phi_i) \subseteq W$ and $\mathrm{neg}(\phi_i) \cap W = \varnothing$. Therefore, $I_a \cap \mathrm{pos}(\phi_i) \subseteq I_a \cap W = \varnothing$ and $R_a \cap \mathrm{neg}(\phi_i) \subseteq W \cap \mathrm{neg}(\phi_i) = \varnothing$, which lead to $\Rightarrow I_a \cap \mathrm{pos}(\phi_i) = \varnothing$ and $R_a \cap \mathrm{neg}(\phi_i) = \varnothing$, respectively.

(2)⇒(1): Suppose that ϕ contains a conjunction ϕ_i such that $R_a \cap \mathrm{neg}(\phi_i) = I_a \cap \mathrm{pos}(\phi_i) = \varnothing$ and consider the set $W = R_a \cup \mathrm{pos}(\phi_i)$. Clearly, $\phi_i(W)$ holds, because $\mathrm{pos}(\phi_i) \subseteq W$, and because $R_a \cap \mathrm{neg}(\phi_i) = \mathrm{pos}(\phi_i) \cap \mathrm{neg}(\phi_i) = \varnothing$. On the other hand, we also know that $R_a \subseteq W$ and $I_a \cap R_a = I_a \cap \mathrm{pos}(\phi_i) = \varnothing$, so the reaction a is enabled on $R_a \cup \mathrm{pos}(\phi_i)$. We have therefore successfully constructed a set satisfying the statement (1). □

We will write $\mathrm{en}_a(\phi) = 1$, or just $\mathrm{en}_a(\phi)$, to refer to the fact that a is enabled on a set satisfying ϕ.

The following two observations give negative heuristic criteria for formula correspondence. Both cases are formulated in the setting of a reaction system $\mathcal{A} = (S, A)$ and two Boolean formulae $\varphi = \bigvee_{i=1}^{n} \phi_i$ and $\psi = \bigvee_{j=1}^{m} \psi_j$ over S.

Lemma 5.2. *If $\mathcal{A}$ contains a reaction a such that $\mathrm{en}_a(\phi)$, but $P_a \cap \mathrm{neg}(\psi_j) \neq \varnothing$, for all $1 \leq j \leq m$, then there exists a subset $W \subseteq S$ for which $\phi(W) \not\Rightarrow \psi(\mathrm{res}_{\mathcal{A}}(W))$.*

Proof. Since we know that $\mathrm{en}_a(\phi)$, there exists such a subset $W \subseteq S$ that $\phi(W)$ and $\mathrm{en}_a(W)$. The hypothesis that the product set of a intersects all $\mathrm{neg}(\psi_j)$ means that $\mathrm{res}_{\mathcal{A}}(W)$ intersects all $\mathrm{neg}(\psi_j)$ as well, and therefore $\psi(\mathrm{res}_{\mathcal{A}}(W))$ does not hold. □

Verifying the condition of the previous lemma requires going through both ϕ and ψ for every reaction of $\mathcal{A}$. The time complexity of such a procedure is in $O\big(|\phi_1| \cdot |\psi| \cdot (N_R + N_I + N_P)\big)$, where $|\phi|$ is the number of atomic terms in the disjunctive normal form of ϕ, while N_R, N_I, and N_P are the total sizes of the reactant, inhibitor, and product sets of the reactions in $\mathcal{A}$:

$$N_R = \sum_{a \in A} |R_a|, \;\; N_I = \sum_{a \in A} |I_a|, \;\; N_P = \sum_{a \in A} |P_a|.$$

Lemma 5.3. *Consider the set $B = \{b \mid b \in A, \mathrm{en}_b(\phi)\}$ and take the union of the products of the reactions in this set: $\bar{P} = \bigcup_{b \in B} P_b$. If, for any conjunction ψ_j of Ψ, it is true that $\mathrm{pos}(\psi_j) \not\subseteq P$, then there exists a subset $W \subseteq S$ for which $\phi(W) \not\Rightarrow \psi(\mathrm{res}_{\mathcal{A}}(W))$.*

Proof. Consider a set W such that $\phi(W)$. Then, by definition of the set B, $\mathrm{res}_{\mathcal{A}}(W) \subseteq \bar{P}$. But, since formula ψ contains *no* conjunction ψ_j such that $\mathrm{pos}(\psi_j) \subseteq \bar{P}$, this means that $\mathrm{res}_{\mathcal{A}}(W)$ satisfies no conjunction of ψ and therefore does not satisfy ψ. □

Verifying the condition of this lemma requires going through ϕ for every reaction in the system ($O\big(|\phi| \cdot (N_R + N_I)\big)$ steps), putting together the product sets of certain reactions ($O(N_P)$ steps), and then checking if the non-negated variables of a conjunction of ψ form a subset of this union ($O(|\psi| \cdot N_P)$ steps). The time complexity of such a procedure can therefore be estimated to belong to $O\big(|\phi| \cdot (N_R + N_I) + |\psi| \cdot N_P\big)$.

To formulate a heuristic criterion for mass conservation, we will rewrite this problem in Boolean formulae. The arguments in Section 2 allow us to consider mass conservation over the full background set. For a set $M \subseteq S$, the sets W satisfying the condition $M \cap W \neq \varnothing$ are exactly the sets satisfying the following Boolean formula:

$$\phi = \bigvee_{x \in M} x.$$

The property of M being conserved can then be written as follows (cf. [8]):

$$\forall W \subseteq S, \phi(W) \Leftrightarrow \phi(\mathrm{res}_{\mathcal{A}}(W)).$$

Applying the statement of Lemma 5.2 to this particular instance of the formula correspondence problem is ineffective, because no conjunction in ψ contains negated variables. However, instantiating the statement of Lemma 5.3 (and that of Lemma 5.1) yields the following negative heuristics for mass conservation.

Please cite this article in press as: S. Azimi et al., Dependency graphs and mass conservation in reaction systems, Theoret. Comput. Sci. (2015), http://dx.doi.org/10.1016/j.tcs.2015.02.014

Corollary 5.4. *Consider a reaction system $\mathcal{A} = (S, A)$, a subset of species $M \subseteq S$, and a subset of reactions $B = \{b \mid b = (R_b, I_b, P_b) \in A, M \setminus I_b \neq \varnothing\}$. If it is true that $M \cap \bigcup_{b \in B} P_b = \varnothing$, then M is* not *conserved in $\mathcal{A}$.*

Example 5.1. Consider the reaction system $\mathcal{A} = (S, A)$, with $S = \{x, y\}$ and

$$A = \{(\{x\}, \{y\}, \{y\}), (\{y\}, \{x\}, \{x\})\}.$$

$\mathcal{A}$ conserves no nonempty sets, because $\mathrm{res}_{\mathcal{A}}(S) = \varnothing$. Take $M_1 = \{x\}$, then $B_1 = \{(\{x\}, \{y\}, \{y\})\}$. Since $M_1 \cap \bigcup_{b \in B_1} P_b = \{x\} \cap \{y\} = \varnothing$, the previous corollary allows us to correctly conclude that M_1 is not conserved. Consider, on the other hand, the set $M_2 = \{x, y\}$. Then $B_2 = A$, $M_2 \cap \bigcup_{b \in B_2} P_b = \{x, y\} \cap \{x, y\} = \{x, y\} \neq \varnothing$, and the criterion does not allow us to conclude whether M_2 is conserved or not.

6. Reaction system simulator

Even though it is relatively easy to write out an interactive process of a reaction system given a context sequence, doing this by hand quickly becomes tedious and error-prone. To automate the task, we developed a reaction system simulator, `brsim`. This is a stand-alone software tool which reads the description of a reaction system and a sequence of contexts from a file, runs the system with the supplied contexts, and then outputs the sequence of results. The simulator includes the option of annotating the evolution, in which case, for each evolution step, it will show the previous result, the context added at the current step, as well as the reactions enabled in the current state. Interactively running the reaction system is also supported, in which case the simulator will ask for the new context at each step.

Besides being able to run a reaction system for a given context sequence, the simulator can also show its conservation dependency graph as well as compute and list the conserved sets using an implementation of the algorithm shown in Section 4.

The source code of the simulator is licensed under GPLv3 and is available at [14]. We also provide a web interface to `brsim` at [15].

The input format of the simulator is similar to the notations conventionally used to write reactions. For example, a reaction system containing the reactions $(\{a\}, \{b, x\}, \{a\})$ and $(\{b\}, \{a, x\}, \{b\})$ would be described as follows:

```
a, b x, a
b, a x, b
```

The context sequence $C_0 = \{a, b\}$, $C_1 = \varnothing$, $C_2 = \{a, x\}$ would be represented in the following way:

```
a b
.
a x
```

For further details about using the simulator as a stand-alone application or via its web interface we refer the reader to [14,15].

7. Conclusion

In this paper we focused on the biologically inspired property of mass conservation in reaction systems and unveiled the conservation dependency relation it induces between the species. It turned out that relying on the conservation dependency graph makes it possible to design an algorithm for listing the conserved sets which, in certain cases, performs better than the naive approach. Because conserved sets can well be exponential in number (cf. [8]), we cannot expect to build an algorithm which would always work in subexponential time. Yet, the fact that using the conservation dependency graph allows reducing the number of computational steps in some cases serves as an example of how observing certain *structural* properties of a reaction system can help to answer difficult questions more efficiently.

Several bibliographical references indicate that, in various biochemical networks, the number of reactions is linear in the number of involved species. For example, the metabolic networks of the bacteria *E. coli*, *H. influenzae*, *H. pylori*, and *G. sulfurreducens* employ 660, 296, 291, and 588 genes, and 720, 488, 288, and 523 reactions respectively [16]. It is equally noteworthy that the stoichiometric matrices of biochemical networks are generally sparse, since only a few species usually participate in a reaction [17]. In view of these observations, the conjecture proposed in Section 4 would imply that, for reaction system models of real-life biochemical networks, Algorithm 4.1 will only produce a polynomial number of *non-conserved* candidate sets (in terms of the number of species).

In Section 5 we also provided a sufficient polynomial criterion which can be used to prove that a given set of species is not conserved. The criterion is built around a different series of observations revealing yet other connections between the

Please cite this article in press as: S. Azimi et al., Dependency graphs and mass conservation in reaction systems, Theoret. Comput. Sci. (2015), http://dx.doi.org/10.1016/j.tcs.2015.02.014

inner structure of the reaction system and the sets it conserves. Because deciding the conservation of a set is coNP-complete, we could not hope to have a sufficient *and* necessary criterion which would also be polynomial.

While we do show an important application of the conservation dependency graph to listing the conserved sets of a reaction system, we expect that a number of other properties of this graph remain to be further explored. A promising research direction would be that of establishing in which way the conservation dependency graph is related to other conservation properties, like invariant sets, or the formula correspondence problems (see [8] for the definitions).

Lastly, in Section 6, we presented the simulator `brsim` which automates the process of running a reaction system with a given sequence of contexts, but also supports listing the conserved sets using an implementation of Algorithm 4.1. Since it is possible to both run the simulator as a stand-alone application and work with it via a web interface, we hope that it will be useful to the actively growing community of researchers working in the domain of reaction systems.

Acknowledgements

Sepinoud Azimi, Cristian Gratie and Ion Petre gratefully acknowledge support from Academy of Finland through projects 267915 and 272559. This work was partially done during Sergiu Ivanov's visit at the Computational Biomodeling Laboratory, Turku, Finland in Spring 2014.

References

[1] A. Ehrenfeucht, G. Rozenberg, Reaction systems, Fund. Inform. 75 (1) (2007) 263–280.
[2] R. Brijder, A. Ehrenfeucht, M.G. Main, G. Rozenberg, A tour of reaction systems, Internat. J. Found. Comput. Sci. 22 (7) (2011) 1499–1517.
[3] A. Ehrenfeucht, M. Main, G. Rozenberg, Functions defined by reaction systems, Internat. J. Found. Comput. Sci. 22 (01) (2011) 167–178.
[4] E. Formenti, L. Manzoni, A.E. Porreca, Cycles and global attractors of reaction systems, in: H. Jürgensen, J. Karhumäki, A. Okhotin (Eds.), Descriptional Complexity of Formal Systems, in: Lecture Notes in Computer Science, vol. 8614, Springer, 2014, pp. 114–125.
[5] E. Formenti, L. Manzoni, A.E. Porreca, Fixed points and attractors of reaction systems, in: A. Beckmann, E. Csuhaj-Varjú, K. Meer (Eds.), Language, Life, Limits, 10th Conference on Computability in Europe, CiE 2014, in: Lecture Notes in Computer Science, vol. 8493, Springer, 2014, pp. 194–203.
[6] A. Salomaa, Functions and sequences generated by reaction systems, Theoret. Comput. Sci. 466 (2012) 87–96.
[7] A. Salomaa, Functional constructions between reaction systems and propositional logic, Internat. J. Found. Comput. Sci. 24 (1) (2013) 147–159.
[8] S. Azimi, C. Gratie, S. Ivanov, L. Manzoni, I. Petre, A.E. Porreca, Complexity of model checking for reaction systems, Tech. Rep. 1122, Turku Centre for Computer Science, 2014.
[9] S. Azimi, B. Iancu, I. Petre, Reaction system models for the heat shock response, Fund. Inform. 131 (3) (2014) 299–312.
[10] L. Corolli, C. Maj, F. Marini, D. Besozzi, G. Mauri, An excursion in reaction systems: from computer science to biology, Theoret. Comput. Sci. 454 (2012) 95–108.
[11] A. Meski, W. Penczek, G. Rozenberg, Model checking temporal properties of reaction systems, Tech. Rep. 1028, ICS PAS, Warsaw, 2014.
[12] I. Petre, A. Mizera, C.L. Hyder, A. Meinander, A. Mikhailov, R.I. Morimoto, L. Sistonen, J.E. Eriksson, R.-J. Back, A simple mass-action model for the eukaryotic heat shock response and its mathematical validation, Nat. Comput. 10 (1) (2011) 595–612.
[13] A. Gibbons, Algorithmic Graph Theory, Cambridge University Press, 1985.
[14] The web interface of the reaction system simulator, http://combio.abo.fi/research/reaction-systems/reaction-system-simulator/.
[15] The source code of the reaction system simulator, https://github.com/scolobb/brsim.
[16] B.O. Palsson, Systems Biology, Cambridge University Press, 2006.
[17] L. Edsberg, Introduction to Computation and Modeling for Differential Equations, John Wiley & Sons, 2013.

Turku Centre for Computer Science
TUCS Dissertations

1. **Marjo Lipponen**, On Primitive Solutions of the Post Correspondence Problem
2. **Timo Käkölä**, Dual Information Systems in Hyperknowledge Organizations
3. **Ville Leppänen**, Studies on the Realization of PRAM
4. **Cunsheng Ding**, Cryptographic Counter Generators
5. **Sami Viitanen**, Some New Global Optimization Algorithms
6. **Tapio Salakoski**, Representative Classification of Protein Structures
7. **Thomas Långbacka**, An Interactive Environment Supporting the Development of Formally Correct Programs
8. **Thomas Finne**, A Decision Support System for Improving Information Security
9. **Valeria Mihalache**, Cooperation, Communication, Control. Investigations on Grammar Systems.
10. **Marina Waldén**, Formal Reasoning About Distributed Algorithms
11. **Tero Laihonen**, Estimates on the Covering Radius When the Dual Distance is Known
12. **Lucian Ilie**, Decision Problems on Orders of Words
13. **Jukkapekka Hekanaho**, An Evolutionary Approach to Concept Learning
14. **Jouni Järvinen**, Knowledge Representation and Rough Sets
15. **Tomi Pasanen**, In-Place Algorithms for Sorting Problems
16. **Mika Johnsson**, Operational and Tactical Level Optimization in Printed Circuit Board Assembly
17. **Mats Aspnäs**, Multiprocessor Architecture and Programming: The Hathi-2 System
18. **Anna Mikhajlova**, Ensuring Correctness of Object and Component Systems
19. **Vesa Torvinen**, Construction and Evaluation of the Labour Game Method
20. **Jorma Boberg**, Cluster Analysis. A Mathematical Approach with Applications to Protein Structures
21. **Leonid Mikhajlov**, Software Reuse Mechanisms and Techniques: Safety Versus Flexibility
22. **Timo Kaukoranta**, Iterative and Hierarchical Methods for Codebook Generation in Vector Quantization
23. **Gábor Magyar**, On Solution Approaches for Some Industrially Motivated Combinatorial Optimization Problems
24. **Linas Laibinis**, Mechanised Formal Reasoning About Modular Programs
25. **Shuhua Liu**, Improving Executive Support in Strategic Scanning with Software Agent Systems
26. **Jaakko Järvi**, New Techniques in Generic Programming – C++ is more Intentional than Intended
27. **Jan-Christian Lehtinen**, Reproducing Kernel Splines in the Analysis of Medical Data
28. **Martin Büchi**, Safe Language Mechanisms for Modularization and Concurrency
29. **Elena Troubitsyna**, Stepwise Development of Dependable Systems
30. **Janne Näppi**, Computer-Assisted Diagnosis of Breast Calcifications
31. **Jianming Liang**, Dynamic Chest Images Analysis
32. **Tiberiu Seceleanu**, Systematic Design of Synchronous Digital Circuits
33. **Tero Aittokallio**, Characterization and Modelling of the Cardiorespiratory System in Sleep-Disordered Breathing
34. **Ivan Porres**, Modeling and Analyzing Software Behavior in UML
35. **Mauno Rönkkö**, Stepwise Development of Hybrid Systems
36. **Jouni Smed**, Production Planning in Printed Circuit Board Assembly
37. **Vesa Halava**, The Post Correspondence Problem for Market Morphisms
38. **Ion Petre**, Commutation Problems on Sets of Words and Formal Power Series
39. **Vladimir Kvassov**, Information Technology and the Productivity of Managerial Work
40. **Frank Tétard**, Managers, Fragmentation of Working Time, and Information Systems

41. **Jan Manuch**, Defect Theorems and Infinite Words
42. **Kalle Ranto**, Z_4-Goethals Codes, Decoding and Designs
43. **Arto Lepistö**, On Relations Between Local and Global Periodicity
44. **Mika Hirvensalo**, Studies on Boolean Functions Related to Quantum Computing
45. **Pentti Virtanen**, Measuring and Improving Component-Based Software Development
46. **Adekunle Okunoye**, Knowledge Management and Global Diversity – A Framework to Support Organisations in Developing Countries
47. **Antonina Kloptchenko**, Text Mining Based on the Prototype Matching Method
48. **Juha Kivijärvi**, Optimization Methods for Clustering
49. **Rimvydas Rukšėnas**, Formal Development of Concurrent Components
50. **Dirk Nowotka**, Periodicity and Unbordered Factors of Words
51. **Attila Gyenesei**, Discovering Frequent Fuzzy Patterns in Relations of Quantitative Attributes
52. **Petteri Kaitovaara**, Packaging of IT Services – Conceptual and Empirical Studies
53. **Petri Rosendahl**, Niho Type Cross-Correlation Functions and Related Equations
54. **Péter Majlender**, A Normative Approach to Possibility Theory and Soft Decision Support
55. **Seppo Virtanen**, A Framework for Rapid Design and Evaluation of Protocol Processors
56. **Tomas Eklund**, The Self-Organizing Map in Financial Benchmarking
57. **Mikael Collan**, Giga-Investments: Modelling the Valuation of Very Large Industrial Real Investments
58. **Dag Björklund**, A Kernel Language for Unified Code Synthesis
59. **Shengnan Han**, Understanding User Adoption of Mobile Technology: Focusing on Physicians in Finland
60. **Irina Georgescu**, Rational Choice and Revealed Preference: A Fuzzy Approach
61. **Ping Yan**, Limit Cycles for Generalized Liénard-Type and Lotka-Volterra Systems
62. **Joonas Lehtinen**, Coding of Wavelet-Transformed Images
63. **Tommi Meskanen,** On the NTRU Cryptosystem
64. **Saeed Salehi**, Varieties of Tree Languages
65. **Jukka Arvo**, Efficient Algorithms for Hardware-Accelerated Shadow Computation
66. **Mika Hirvikorpi**, On the Tactical Level Production Planning in Flexible Manufacturing Systems
67. **Adrian Costea**, Computational Intelligence Methods for Quantitative Data Mining
68. **Cristina Seceleanu**, A Methodology for Constructing Correct Reactive Systems
69. **Luigia Petre**, Modeling with Action Systems
70. **Lu Yan**, Systematic Design of Ubiquitous Systems
71. **Mehran Gomari**, On the Generalization Ability of Bayesian Neural Networks
72. **Ville Harkke**, Knowledge Freedom for Medical Professionals – An Evaluation Study of a Mobile Information System for Physicians in Finland
73. **Marius Cosmin Codrea**, Pattern Analysis of Chlorophyll Fluorescence Signals
74. **Aiying Rong**, Cogeneration Planning Under the Deregulated Power Market and Emissions Trading Scheme
75. **Chihab BenMoussa**, Supporting the Sales Force through Mobile Information and Communication Technologies: Focusing on the Pharmaceutical Sales Force
76. **Jussi Salmi**, Improving Data Analysis in Proteomics
77. **Orieta Celiku**, Mechanized Reasoning for Dually-Nondeterministic and Probabilistic Programs
78. **Kaj-Mikael Björk**, Supply Chain Efficiency with Some Forest Industry Improvements
79. **Viorel Preoteasa**, Program Variables – The Core of Mechanical Reasoning about Imperative Programs
80. **Jonne Poikonen**, Absolute Value Extraction and Order Statistic Filtering for a Mixed-Mode Array Image Processor
81. **Luka Milovanov**, Agile Software Development in an Academic Environment
82. **Francisco Augusto Alcaraz Garcia**, Real Options, Default Risk and Soft Applications
83. **Kai K. Kimppa**, Problems with the Justification of Intellectual Property Rights in Relation to Software and Other Digitally Distributable Media
84. **Dragoş Truşcan**, Model Driven Development of Programmable Architectures
85. **Eugen Czeizler**, The Inverse Neighborhood Problem and Applications of Welch Sets in Automata Theory

86. **Sanna Ranto**, Identifying and Locating-Dominating Codes in Binary Hamming Spaces
87. **Tuomas Hakkarainen**, On the Computation of the Class Numbers of Real Abelian Fields
88. **Elena Czeizler**, Intricacies of Word Equations
89. **Marcus Alanen**, A Metamodeling Framework for Software Engineering
90. **Filip Ginter**, Towards Information Extraction in the Biomedical Domain: Methods and Resources
91. **Jarkko Paavola**, Signature Ensembles and Receiver Structures for Oversaturated Synchronous DS-CDMA Systems
92. **Arho Virkki**, The Human Respiratory System: Modelling, Analysis and Control
93. **Olli Luoma**, Efficient Methods for Storing and Querying XML Data with Relational Databases
94. **Dubravka Ilić**, Formal Reasoning about Dependability in Model-Driven Development
95. **Kim Solin**, Abstract Algebra of Program Refinement
96. **Tomi Westerlund**, Time Aware Modelling and Analysis of Systems-on-Chip
97. **Kalle Saari**, On the Frequency and Periodicity of Infinite Words
98. **Tomi Kärki**, Similarity Relations on Words: Relational Codes and Periods
99. **Markus M. Mäkelä**, Essays on Software Product Development: A Strategic Management Viewpoint
100. **Roope Vehkalahti**, Class Field Theoretic Methods in the Design of Lattice Signal Constellations
101. **Anne-Maria Ernvall-Hytönen**, On Short Exponential Sums Involving Fourier Coefficients of Holomorphic Cusp Forms
102. **Chang Li**, Parallelism and Complexity in Gene Assembly
103. **Tapio Pahikkala**, New Kernel Functions and Learning Methods for Text and Data Mining
104. **Denis Shestakov**, Search Interfaces on the Web: Querying and Characterizing
105. **Sampo Pyysalo**, A Dependency Parsing Approach to Biomedical Text Mining
106. **Anna Sell**, Mobile Digital Calendars in Knowledge Work
107. **Dorina Marghescu**, Evaluating Multidimensional Visualization Techniques in Data Mining Tasks
108. **Tero Säntti**, A Co-Processor Approach for Efficient Java Execution in Embedded Systems
109. **Kari Salonen**, Setup Optimization in High-Mix Surface Mount PCB Assembly
110. **Pontus Boström**, Formal Design and Verification of Systems Using Domain-Specific Languages
111. **Camilla J. Hollanti**, Order-Theoretic Mehtods for Space-Time Coding: Symmetric and Asymmetric Designs
112. **Heidi Himmanen**, On Transmission System Design for Wireless Broadcasting
113. **Sébastien Lafond**, Simulation of Embedded Systems for Energy Consumption Estimation
114. **Evgeni Tsivtsivadze**, Learning Preferences with Kernel-Based Methods
115. **Petri Salmela**, On Commutation and Conjugacy of Rational Languages and the Fixed Point Method
116. **Siamak Taati**, Conservation Laws in Cellular Automata
117. **Vladimir Rogojin**, Gene Assembly in Stichotrichous Ciliates: Elementary Operations, Parallelism and Computation
118. **Alexey Dudkov**, Chip and Signature Interleaving in DS CDMA Systems
119. **Janne Savela**, Role of Selected Spectral Attributes in the Perception of Synthetic Vowels
120. **Kristian Nybom**, Low-Density Parity-Check Codes for Wireless Datacast Networks
121. **Johanna Tuominen**, Formal Power Analysis of Systems-on-Chip
122. **Teijo Lehtonen**, On Fault Tolerance Methods for Networks-on-Chip
123. **Eeva Suvitie**, On Inner Products Involving Holomorphic Cusp Forms and Maass Forms
124. **Linda Mannila**, Teaching Mathematics and Programming – New Approaches with Empirical Evaluation
125. **Hanna Suominen**, Machine Learning and Clinical Text: Supporting Health Information Flow
126. **Tuomo Saarni**, Segmental Durations of Speech
127. **Johannes Eriksson**, Tool-Supported Invariant-Based Programming

128. **Tero Jokela**, Design and Analysis of Forward Error Control Coding and Signaling for Guaranteeing QoS in Wireless Broadcast Systems
129. **Ville Lukkarila**, On Undecidable Dynamical Properties of Reversible One-Dimensional Cellular Automata
130. **Qaisar Ahmad Malik**, Combining Model-Based Testing and Stepwise Formal Development
131. **Mikko-Jussi Laakso**, Promoting Programming Learning: Engagement, Automatic Assessment with Immediate Feedback in Visualizations
132. **Riikka Vuokko**, A Practice Perspective on Organizational Implementation of Information Technology
133. **Jeanette Heidenberg**, Towards Increased Productivity and Quality in Software Development Using Agile, Lean and Collaborative Approaches
134. **Yong Liu**, Solving the Puzzle of Mobile Learning Adoption
135. **Stina Ojala**, Towards an Integrative Information Society: Studies on Individuality in Speech and Sign
136. **Matteo Brunelli**, Some Advances in Mathematical Models for Preference Relations
137. **Ville Junnila**, On Identifying and Locating-Dominating Codes
138. **Andrzej Mizera**, Methods for Construction and Analysis of Computational Models in Systems Biology. Applications to the Modelling of the Heat Shock Response and the Self-Assembly of Intermediate Filaments.
139. **Csaba Ráduly-Baka**, Algorithmic Solutions for Combinatorial Problems in Resource Management of Manufacturing Environments
140. **Jari Kyngäs**, Solving Challenging Real-World Scheduling Problems
141. **Arho Suominen**, Notes on Emerging Technologies
142. **József Mezei**, A Quantitative View on Fuzzy Numbers
143. **Marta Olszewska**, On the Impact of Rigorous Approaches on the Quality of Development
144. **Antti Airola**, Kernel-Based Ranking: Methods for Learning and Performace Estimation
145. **Aleksi Saarela**, Word Equations and Related Topics: Independence, Decidability and Characterizations
146. **Lasse Bergroth**, Kahden merkkijonon pisimmän yhteisen alijonon ongelma ja sen ratkaiseminen
147. **Thomas Canhao Xu**, Hardware/Software Co-Design for Multicore Architectures
148. **Tuomas Mäkilä**, Software Development Process Modeling – Developers Perspective to Contemporary Modeling Techniques
149. **Shahrokh Nikou**, Opening the Black-Box of IT Artifacts: Looking into Mobile Service Characteristics and Individual Perception
150. **Alessandro Buoni**, Fraud Detection in the Banking Sector: A Multi-Agent Approach
151. **Mats Neovius**, Trustworthy Context Dependency in Ubiquitous Systems
152. **Fredrik Degerlund**, Scheduling of Guarded Command Based Models
153. **Amir-Mohammad Rahmani-Sane**, Exploration and Design of Power-Efficient Networked Many-Core Systems
154. **Ville Rantala**, On Dynamic Monitoring Methods for Networks-on-Chip
155. **Mikko Pelto**, On Identifying and Locating-Dominating Codes in the Infinite King Grid
156. **Anton Tarasyuk**, Formal Development and Quantitative Verification of Dependable Systems
157. **Muhammad Mohsin Saleemi**, Towards Combining Interactive Mobile TV and Smart Spaces: Architectures, Tools and Application Development
158. **Tommi J. M. Lehtinen**, Numbers and Languages
159. **Peter Sarlin**, Mapping Financial Stability
160. **Alexander Wei Yin**, On Energy Efficient Computing Platforms
161. **Mikołaj Olszewski**, Scaling Up Stepwise Feature Introduction to Construction of Large Software Systems
162. **Maryam Kamali**, Reusable Formal Architectures for Networked Systems
163. **Zhiyuan Yao**, Visual Customer Segmentation and Behavior Analysis – A SOM-Based Approach
164. **Timo Jolivet**, Combinatorics of Pisot Substitutions
165. **Rajeev Kumar Kanth**, Analysis and Life Cycle Assessment of Printed Antennas for Sustainable Wireless Systems
166. **Khalid Latif**, Design Space Exploration for MPSoC Architectures

167. **Bo Yang**, Towards Optimal Application Mapping for Energy-Efficient Many-Core Platforms
168. **Ali Hanzala Khan**, Consistency of UML Based Designs Using Ontology Reasoners
169. **Sonja Leskinen**, m-Equine: IS Support for the Horse Industry
170. **Fareed Ahmed Jokhio**, Video Transcoding in a Distributed Cloud Computing Environment
171. **Moazzam Fareed Niazi**, A Model-Based Development and Verification Framework for Distributed System-on-Chip Architecture
172. **Mari Huova**, Combinatorics on Words: New Aspects on Avoidability, Defect Effect, Equations and Palindromes
173. **Ville Timonen**, Scalable Algorithms for Height Field Illumination
174. **Henri Korvela**, Virtual Communities – A Virtual Treasure Trove for End-User Developers
175. **Kameswar Rao Vaddina**, Thermal-Aware Networked Many-Core Systems
176. **Janne Lahtiranta**, New and Emerging Challenges of the ICT-Mediated Health and Well-Being Services
177. **Irum Rauf**, Design and Validation of Stateful Composite RESTful Web Services
178. **Jari Björne**, Biomedical Event Extraction with Machine Learning
179. **Katri Haverinen**, Natural Language Processing Resources for Finnish: Corpus Development in the General and Clinical Domains
180. **Ville Salo**, Subshifts with Simple Cellular Automata
181. **Johan Ersfolk**, Scheduling Dynamic Dataflow Graphs
182. **Hongyan Liu**, On Advancing Business Intelligence in the Electricity Retail Market
183. **Adnan Ashraf**, Cost-Efficient Virtual Machine Management: Provisioning, Admission Control, and Consolidation
184. **Muhammad Nazrul Islam**, Design and Evaluation of Web Interface Signs to Improve Web Usability: A Semiotic Framework
185. **Johannes Tuikkala**, Algorithmic Techniques in Gene Expression Processing: From Imputation to Visualization
186. **Natalia Díaz Rodríguez**, Semantic and Fuzzy Modelling for Human Behaviour Recognition in Smart Spaces. A Case Study on Ambient Assisted Living
187. **Mikko Pänkäälä**, Potential and Challenges of Analog Reconfigurable Computation in Modern and Future CMOS
188. **Sami Hyrynsalmi**, Letters from the War of Ecosystems – An Analysis of Independent Software Vendors in Mobile Application Marketplaces
189. **Seppo Pulkkinen**, Efficient Optimization Algorithms for Nonlinear Data Analysis
190. **Sami Pyöttiälä**, Optimization and Measuring Techniques for Collect-and-Place Machines in Printed Circuit Board Industry
191. **Syed Mohammad Asad Hassan Jafri**, Virtual Runtime Application Partitions for Resource Management in Massively Parallel Architectures
192. **Toni Ernvall**, On Distributed Storage Codes
193. **Yuliya Prokhorova**, Rigorous Development of Safety-Critical Systems
194. **Olli Lahdenoja**, Local Binary Patterns in Focal-Plane Processing – Analysis and Applications
195. **Annika H. Holmbom**, Visual Analytics for Behavioral and Niche Market Segmentation
196. **Sergey Ostroumov**, Agent-Based Management System for Many-Core Platforms: Rigorous Design and Efficient Implementation
197. **Espen Suenson**, How Computer Programmers Work – Understanding Software Development in Practise
198. **Tuomas Poikela**, Readout Architectures for Hybrid Pixel Detector Readout Chips
199. **Bogdan Iancu**, Quantitative Refinement of Reaction-Based Biomodels
200. **Ilkka Törmä**, Structural and Computational Existence Results for Multidimensional Subshifts
201. **Sebastian Okser**, Scalable Feature Selection Applications for Genome-Wide Association Studies of Complex Diseases
202. **Fredrik Abbors**, Model-Based Testing of Software Systems: Functionality and Performance
203. **Inna Pereverzeva**, Formal Development of Resilient Distributed Systems
204. **Mikhail Barash**, Defining Contexts in Context-Free Grammars
205. **Sepinoud Azimi**, Computational Models for and from Biology: Simple Gene Assembly and Reaction Systems

Turku Centre *for* Computer Science

Joukahaisenkatu 3-5 B, 20520 Turku, Finland | www. tucs.fi

University of Turku
Faculty of Mathematics and Natural Sciences
- Department of Information Technology
- Department of Mathematics and Statistics

Turku School of Economics
- Institute of Information Systems Science

Åbo Akademi University
Faculty of Science and Engineering
- Computer Engineering
- Computer Science

Faculty of Social Sciences, Business and Economics
- Information Systems

ISBN 978-952-12-3289-3
ISSN 1239-1883

Sepinoud Azimi

Computational Models for and from Biology

Sepinoud Azimi

Computational Models for and from Biology

Sepinoud Azimi

Computational Models for and from Biology: Simple Gene Assembly and Reaction Systems