THE UNIVERSITY
of EDINBURGH

Defining Complex Rule-Based Models in Space and over Time

John Roger Wilson-Kanamori

Doctor of Philosophy
Laboratory for Foundations of Computer Science
School of Informatics
University of Edinburgh
2015

Abstract

Computational biology seeks to understand complex spatio-temporal phenomena across multiple levels of structural and functional organisation. However, questions raised in this context are difficult to answer without modelling methodologies that are intuitive and approachable for non-expert users. Stochastic rule-based modelling languages such as Kappa have been the focus of recent attention in developing complex biological models that are nevertheless concise, comprehensible, and easily extensible. We look at further developing Kappa, in terms of how we might define complex models in both the spatial and the temporal axes.

In defining complex models in space, we address the assumption that the reaction mixture of a Kappa model is homogeneous and well-mixed. We propose evolutions of the current iteration of Spatial Kappa to streamline the process of defining spatial structures for different modelling purposes. We also verify the existing implementation against established results in diffusion and narrow escape, thus laying the foundations for querying a wider range of spatial systems with greater confidence in the accuracy of the results.

In defining complex models over time, we draw attention to how non-modelling specialists might define, verify, and analyse rules throughout a rigorous model development process. We propose structured visual methodologies for developing and maintaining knowledge base data structures, incorporating the information needed to construct a Kappa rule-based model. We further extend these methodologies to deal with biological systems defined by the activity of synthetic genetic parts, with the hope of providing tractable operations that allow multiple users to contribute to their development over time according to their area of expertise.

Throughout the thesis we pursue the aim of bridging the divide between information sources such as literature and bioinformatics databases and the abstracting decisions inherent in a model. We consider methodologies for automating the construction of spatial models, providing traceable links from source to model element, and updating a model via an iterative and collaborative development process. By providing frameworks for modellers from multiple domains of expertise to work with the language, we reduce the entry barrier and open the field to further questions and new research.

Lay Summary

When trying to understand what goes on inside a biological cell, we use *models* to explain what we don't know, to form and test hypotheses, to provide a way to predict interactions, and to guide future research. However, we often find it difficult to *define* such models in a way that is easy to understand and to edit, in particular those *complex* models that need to take into account the effect of *space* in the system of interest. For example, if the transfer of information from the outside world occurs in one part of the cell but it has its effect in another, modelling this explicitly adds another layer of complexity to the knowledge that underlies our system.

We also often find it difficult to directly relate what we actually know (in the form of experimental data or biological literature) to what we model. For example, if we derive a certain rate of interaction between two proteins from variables maintained in an online database, it makes sense to track these variables so that we can update our model rate along with any changes to the underlying database. Not doing so makes it difficult to revise and reuse models when our knowledge evolves over *time*, and results in single-use models fitted to a single purpose that are abandoned almost as soon as they are published.

Rule-based modelling, including the Kappa language, is a modelling formalism that takes steps towards solving these problems by intuitively specifying interactions between the biological entities under simulation. We can make use of dedicated modules for defining space such as the Spatial Kappa extension, as well as encode certain assumptions such as equations upon variables into our model such that we may repurpose models to ask newly relevant questions.

In this thesis, we work to make it easier for non-experts in biological modelling to define complex models in space and over time. We compare the current implementation of Spatial Kappa against well-known results in diffusion theory, allowing us to state with confidence that the results it provides are accurate to a certain degree of assumption. We develop more intuitive definitions of spatial structures to suit a wider range of modelling purposes, allowing modellers to choose how best to model their system in space. We introduce a visual methodology for defining, developing, and maintaining model information, including the source data that underlies its assumptions. And we propose an integrated model development environment that allows modellers from multiple domains of expertise to work together with a single knowledge base. Our aim is to open the field of biological modelling to further questions and research on ever-more complex systems of interest.

Throughout the thesis, we illustrate our efforts with biological models developed in collaboration with other researchers. In particular, we not only look at biological systems that exist in nature, but also venture into the exciting field of biological circuits that are developed synthetically and to purpose, tailoring our development environment to both approaches.

Acknowledgements

The author would like to thank Professor Vincent Danos for his support and supervision not only throughout the course of his PhD, but also for nearly two years beforehand as well. He would also like to thank Professor Heinz Koeppl and Assistant Professor Soichiro Hidaka for fruitful internships at ETH Zurich and the National Institute of Informatics in Tokyo respectively.

The author is grateful to Ricardo Honorato-Zimmer for countless discussions throughout the years; to Donal Stewart, John Moore, and Peter Krenn for collaborations on the various models introduced in this paper; to Anatoly Sorokin for his insight and analysis of Spatial Kappa; to Matteo Cavaliere and Goksel Misirli for useful conversations regarding modelling in synthetic biology; to William Waites for collaborative work and support in the Module Integration Simulator.

The author is also grateful to Wolfson Microelectronics and the School of Informatics for the scholarship funding that made possible the journey chronicled in this thesis.

Finally, the author wishes to acknowledge the unfailing support of his family during his university experience, even if it took him clear to the opposite side of the world to accomplish. He also would like to thank all acquaintances in the School of Informatics – in particular his teammates in Informatics FC – for giving him the confidence to step out into life.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*John Roger Wilson-Kanamori*)

To the bright star that has guided my blind meandering for so many years.

Table of Contents

Chapter 1

Introduction

In this chapter we...

- ❖ Introduce the field of computational modelling in systems and synthetic biology.
- ❖ Introduce the Kappa stochastic rule-based modelling language and its spatial extension, Spatial Kappa.
- ❖ Motivate our thesis with the questions we wish to ask regarding the definition of complex models in space and over time.
- ❖ Specify our aims, approaches, and contributions.
- ❖ Outline the chapters to follow.

Models of complex systems educate and explain the previously unfathomed, promote proper scientific process by forming and testing hypotheses, provide a methodology of meta-prediction, and guide future research [30]. They formalise noisy, oft-incomprehensible interactions in concise well-defined hypotheses ("under these assumptions, I would expect this to happen; if I alter my assumptions, that happens instead") by incorporating domain expertise in a rigorous manner, by calibrating to historical data if it is available, and by testing against current data (*in silico*, *in vivo*, and *in vitro*) to the extent that it exists.

Modern biological research provides some of the largest and noisiest datasets that exist in the world of informatics today. A major challenge in modelling complex genetic and biochemical pathways is translating imprecise and often incomplete knowledge into precise mathematical statements suitable for computational analysis. Recent attention [3] has focused on rule-based representations of interactions between agent-based simulations of species, which have a number of advantages compared to traditional methodologies such as systems of differential equations. In this thesis, we ask the question of how we might define such biological rule-based models from the underlying knowledge, including aspects of space, and how we might maintain them over time. We focus on one such language for the development and spec-

ification of complex rule-based models under development in Edinburgh: Kappa [24, 25] and its extension Spatial Kappa [81, 83].

1.1 Background

Traditional biological modelling utilises systems of ordinary differential equations (ODEs) that focus on the rates of change within short time intervals dt. We write a rate equation for each biochemical entity whose concentration changes over time, following kinetic laws that take into account the known properties of the chemicals involved [66]. The advantage of such models is that they can be easily analysed and manipulated via mathematical formalisms, but they also have an inherent disadvantage in that they do not reflect the fact that molecular populations in biological systems are integer variables that evolve probabilistically [38].

Alternatives to such deterministic modelling include stochastic approaches based on the Gillespie method [37]. For systems with large numbers of reacting species, stochastic models tend towards the results obtainable from deterministic models as the noise effects of stochasticity average out. In turn, they tend to be more accurate in reflecting reality when there are small numbers of reactive components and a single event can have a potentially large effect on the progression of the model.

A more recent development in the field of biological modelling has been the development and use of programming languages to generate executable models of biological systems. Models that are not simply solvable allow the flow of control between species and reactions to be identified, requiring that the modeller not only think about time and rates of change but also about causality relations between the events that make up the history of the model. The development of such languages mirrors the rise of modern chemistry, which was also based on the definition of a formal language for expressing the modular architecture of organic molecules and the rules of reaction between their constituent functional groups.

Such languages for computational biology not only include adapted process calculi such as the continuous π-calculus [54] and Bio-PEPA [14, 15], but also rule-based approaches such as the BioNetGen language (BNGL) [5, 31] and Kappa [24, 25].

1.1.1 Rule-Based Modelling in Kappa (Example: Kinase-Phosphatase)

In reaction-based modelling, all species known to participate in the modelled system are explicitly enumerated, along with every reaction operating on these species. This is feasible in models with limited numbers of reactants and reactions, but biological systems tend to consist of many large species with multiple reaction domains. The corresponding explosion in model size makes many biological models computationally intractable to simulate in a reaction-based manner.

Rules differ from reactions in that participating agents need not be fully specified in the precondition (for example, phosphorylation at a particular site may occur independently from whether its neighbouring site is bound or not) which means that a single rule may encompass up to an infinite number of individual reactions. In this way, rule-based approaches alleviate the combinatorial explosion that results from molecular entities existing under multiple different conditions (for example, states of phosphorylation). The combination of different independent rule sets implicitly generates different overall systems, thus allowing modular development of subsystems and their composition into a conjoined whole.

Subtle differences exist between the two main rule-based modelling languages in biology, Kappa and BNGL, reflective of their distinct origins. The former was conceived by computer scientists as a minimalist framework conducive to developing both models and theory, while the latter was developed by computational biologists and is aimed at supporting the specification and execution of biological models [40]. Thus, for example, BNGL allows for identically named sites whereas Kappa insists that sites must be uniquely identifiable; the former better represents proteins that contain multiple copies of the same domain, but the latter circumvents complex treatment of pattern-matching algorithms used to apply rules. Secondly, BNGL uses two operators (+ and .) to combine site graphs whereas Kappa uses one (,) only; this is a result of the fact that the BioNetGen language cares whether the agents in a rule are connected or not, while Kappa's core syntax has no way of expressing this 'relative location' of agents and thus only requires one combinator. One major benefit for modelling in Kappa rather than BNGL is that Kappa tools utilise formal methods, such as causal summaries and reachability analysis, to aid the information discovery in and the debugging of large models.

Other frameworks exist for adopting a rule-based approach to modelling biological systems, for example little b [59], but these are not as well-developed as the aforementioned pair. In the remainder of this thesis, we adopt the Kappa rule-based modelling language as our language of choice. Let us now proceed to introduce the core concepts of Kappa.

An *agent* in Kappa is simply an entity with a name and a number of labelled *sites*. A site may hold internal *state*, typically used to denote a post-translational modification such as the agent's phosphorylation status. State may also be used to denote the agent's location in a model that takes into account different reaction compartments such as the cytosol and nucleus.

Agent interactions are described via *rules*. Rules often correspond to elementary mechanistic interactions such as the binding or unbinding of two agents, modification of the state of a site, or the creation or deletion of an agent. Complex rules can also describe combinations thereof.

As an introductory example, let us consider a basic kinase-phosphatase model. We consider three agents (Figure 1): a `Kinase`, a `Target` with two sites '`x`' and '`y`' that may be independently phosphorylated, and a `Phosphatase`. A phosphorylation event may be described simply

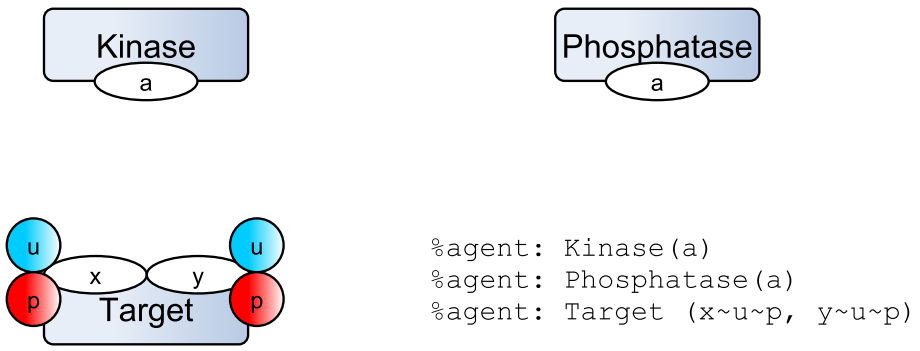


Figure 1: The agents involved in the basic kinase-phosphatase model: a `Kinase` with a single binding site '`a`', a `Phosphatase` also with a single binding site '`a`', and a `Target` with two binding sites '`x`' and '`y`' both of which may also switch state between phosphorylated ('∼`p`') and unphosphorylated ('∼`u`'). We show the textual code corresponding to each agent visualisation at bottom right.

in terms of three elementary actions (binding, modification, and unbinding) and their corresponding rules (Figure 2). In the corresponding textual notation (shown in Figure 1 and Figure 2), internal states are represented as '∼`u`' (unphosphorylated) and '∼`p`' (phosphorylated), and bindings as '`!`' with shared indices across agents indicating the two endpoints of a link. The left hand side of the rule describes the precondition that must be satisfied for the rule to apply. The right hand side describes its effect.

Note that not all sites of an agent need be present in a rule's precondition: the rules shown in Figure 2 never mention the `Target`'s '`y`' site. Likewise, even if a site is mentioned its internal state may be left unspecified: binding a `Kinase` to its `Target` does not take into account the fact that the `Target` may already be phosphorylated at that site. These are both examples of the "don't care, don't write" philosophy, where only the minimal information describing the triggering of a rule need be represented in the left hand side. This is what allows Kappa to alleviate the combinatorial explosion inherent in biological models.

The action of the `Phosphatase`, which works to counteract the `Kinase`, may be described using similar rules. The only difference between the two then lies in the modification rule (Figure 2B), where the state of the `Target` changes from '∼`p`' to '∼`u`' rather than vice versa. Of course, this description is a choice made by the modeller, and it is entirely possible to model the `Phosphatase` (and the `Kinase`) differently but with a similar mechanistic effect. For example, we could design a 'smart' `Phosphatase` that only binds when a `Target` is already phosphorylated. Alternately, we could ensure that it never unbinds without de-phosphorylating by combining the two effects into a single rule.

Every rule is associated with a *rate constant*, which controls the probability of the rule

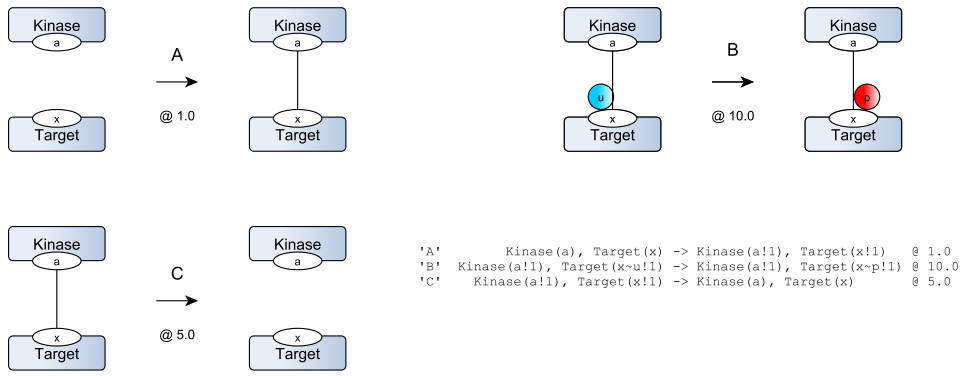


Figure 2: The three rules describing a phosphorylation in the kinase-phosphatase model. A: the `Kinase` binds its `Target` at one of the two sites (in this case 'x'); B: the `Kinase` phosphorylates the site at which it is bound; C: the `Kinase` dissociates (unbinds) from its target. Note the possibility that rule C might fire before rule B, thus not every binding event between a `Kinase` and a `Target` will result in a phosphorylation.

'firing' during the simulation. At any given time in the simulation, any given rule may apply to the *mixture* (the pool of interacting agents, including their binding configuration and internal state) multiple times according to how often the precondition holds. Each possible application has the same rate, hence the number of applications is multiplied by the rate of the rule to determine the rule's propensity (flux) at that point in time. The likelihood that the rule will fire next is proportional to this flux, and assumes that the reaction mixture is homogeneous and well-mixed. Any obtained trajectory, of course, is but one realisation of a stochastic process that may differ when repeated. Manipulating the rate constants will influence these trajectories.

Perturbations form an integrated extension to the base Kappa language allowing the modeller to specify the effect of external influences on the model. For example one may add and delete agents from the mixture, conditioned on simulation time or on other variables pertaining to the state of the system.

Given a set of agents, rules, and perturbations as defined above, as well as appropriate *initial conditions*, we can then track user-defined *observables* in a stochastic *simulation*. The kinase-phosphatase example of a whole Kappa model that may be used for such a simulation is shown in Figure 3. The stochastic trajectories (such as that shown in Figure 4) are obtained by using a rule-based variant of Gillespie's method [37] to simulate a continuous time Markov chain.

As well as this agent-centric view of system dynamics, automated tools can track causality and precedence in a model in a contrasting rule-centric view, for example to answer the question of what succession of events results in a fully phosphorylated `Target`. The idea behind such *stories* is to retain only the events in the causal lineage that contribute a net progression towards

```
### Agents:
%agent: Kinase(a)
%agent: Phosphatase(a)
%agent: Target(x~u~p,y~u~p)


### Rules:

# Kinase action
'Kinase x binding'               Kinase(a), Target(x) -> Kinase(a!1), Target(x!1)    @ 1.0
'Kinase x phosphorylation'  Kinase(a!1), Target(x~u!1) -> Kinase(a!1), Target(x~p!1) @ 10.0
'Kinase x unbinding'          Kinase(a!1), Target(x!1) -> Kinase(a), Target(x)       @ 5.0
'Kinase y binding'               Kinase(a), Target(y) -> Kinase(a!1), Target(y!1)    @ 1.0
'Kinase y phosphorylation'  Kinase(a!1), Target(y~u!1) -> Kinase(a!1), Target(y~p!1) @ 10.0
'Kinase y unbinding'          Kinase(a!1), Target(y!1) -> Kinase(a), Target(y)       @ 5.0

# Phosphatase action
'Phtase x binding'               Phosphatase(a), Target(x) -> Phosphatase(a!1), Target(x!1)    @ 1.0
'Phtase x phosphorylation'  Phosphatase(a!1), Target(x~p!1) -> Phosphatase(a!1), Target(x~u!1) @ 10.0
'Phtase x unbinding'          Phosphatase(a!1), Target(x!1) -> Phosphatase(a), Target(x)       @ 5.0
'Phtase y binding'               Phosphatase(a), Target(y) -> Phosphatase(a!1), Target(y!1)    @ 1.0
'Phtase y phosphorylation'  Phosphatase(a!1), Target(y~p!1) -> Phosphatase(a!1), Target(y~u!1) @ 10.0
'Phtase y unbinding'          Phosphatase(a!1), Target(y!1) -> Phosphatase(a), Target(y)       @ 5.0


### Initial Conditions:
%init: 100 (Target(x~u,y~u))
%init: 10  (Kinase(a))
%init: 10  (Phosphatase(a))
```

Figure 3: The textual representation of the kinase-phosphatase model in Kappa. Agent definitions (Figure 1) are followed by the rules of the model (Figure 2) and the initial population. We do not show here the definition of simulation observables, although we specify them in a similar manner.

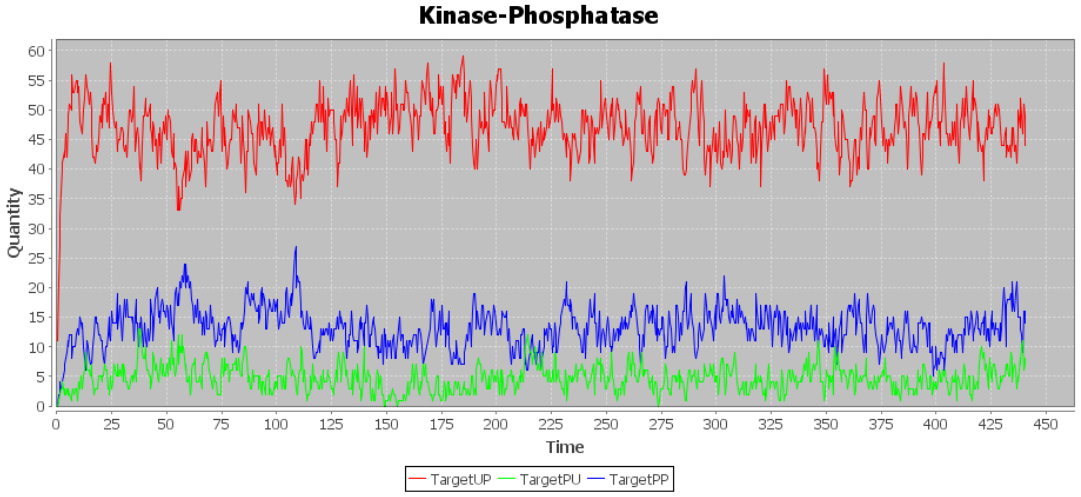


Figure 4: A sample stochastic trajectory for the kinase-phosphatase model in Kappa. The model very swiftly achieves a stochastic equilibrium in which roughly half of the total `Target` population is fully phosphorylated. Simulation conducted in Spatial Kappa tool version 2.1.1.

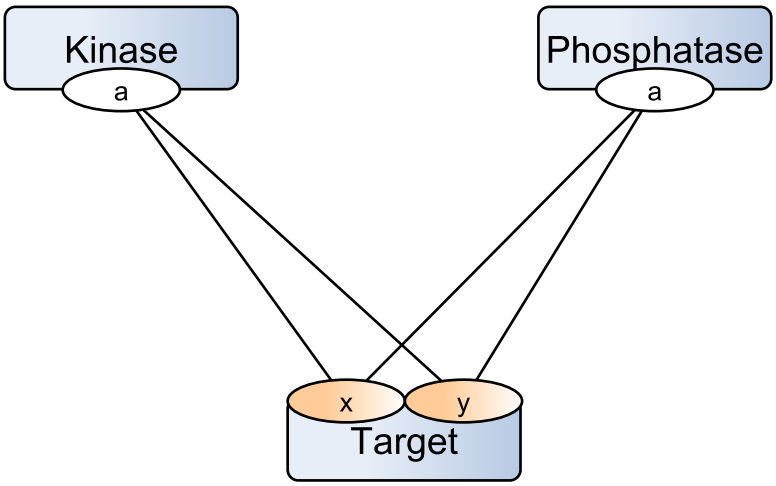


Figure 5: The contact graph for a simple kinase-phosphatase model. The three agents in the model are represented as nodes along with their sites, and each site is connected to the other sites it can bind to. If a site may be modified by this interaction, it is so indicated by a colour code (orange). Although simple enough, the associated rule set (three agents, twelve rules) already generates 38 non-isomorphic complexes (of which 36 contain the `Target` in various stages of binding and phosphorylation).

an event of interest (i.e. by removing causal loops).

A useful overall view of the set of rules in a model is the *contact graph* (Figure 5), akin to a protein-protein interaction map. The contact graph is a graph whose nodes are the agents with their interfacing set of sites, and whose edges represent possible bindings between sites. Possible changes in state are indicated by a colour code upon the site in question.

To summarise: a Kappa model is a collection of agents (sets of sites that may hold state) and their rate-controlled interactions in the form of rules, which may be visualised concisely in the form of a contact graph. Given a set of initial conditions and observables, one may then execute this model to generate a stochastic simulation that tracks agent and mixture evolution over time, modelling external influences via perturbations and observing the causal properties of this evolution via stories.

1.1.2 Extending Rules with Space: Spatial Kappa

Spatial Kappa [81] supports spatial concepts in the context of rule-based modelling, in the form of voxel-composed compartments, channels of diffusion between them, and rules for agents to translocate along these channels. Developed [83, 84] as an extension to Kappa, the aim of Spatial Kappa is to combine the strength of rule-based modelling (alleviating combinatorial complexity between interacting biological entities) with a spatial awareness that the underlying

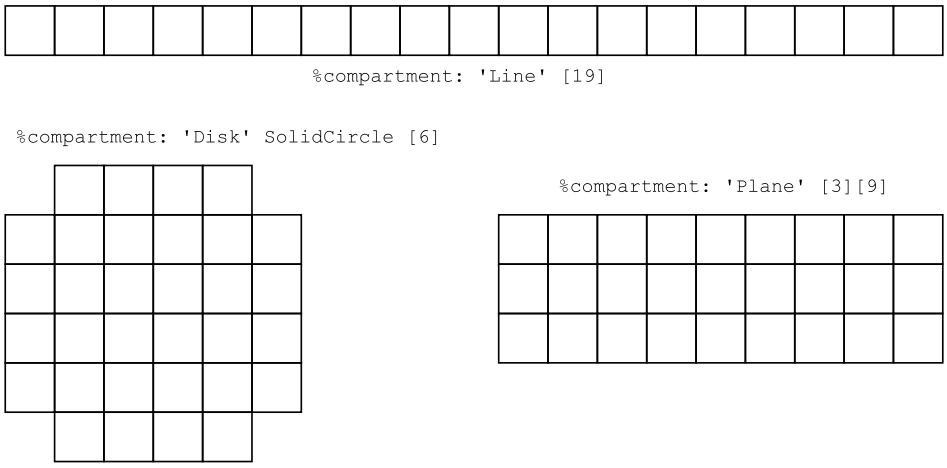


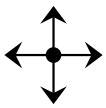
Figure 6: A selection of compartments defined in Spatial Kappa: a one-dimensional `Line` compartment, and two-dimensional `Disk` and `Plane` compartments. Each enclosed space in the grid represents a voxel, and each collection of voxels is a compartment. We provide the corresponding textual definition alongside each graphical representation.

Kappa language had hitherto lacked but was deemed necessary for developing and simulating realistic and complex biological models. The work has since been concretely adopted towards a quantitative model of the post-synaptic proteome [80].

Unlike the Compartmental BioNetGen Language (cBNGL) [41] extension to BNGL, the current implementation of Spatial Kappa does not specify an explicit hierarchical topology for the model. Instead, *compartments* are defined as regular multidimensional arrays of individual *voxels*, examples in one and two dimensions of which are shown in Figure 6. Each voxel may be individually referenced via cell indices according to the dimensions of the compartment. Adopting such a syntax allows us to relax the assumption of spatial homogeneity in Kappa and to model a heterogeneous distribution of agents within the reaction mixture of a compartment.

This spatial structure is further defined by how voxels within the compartment are linked to each other, and by how compartments are linked to neighbouring compartments. We give an example of such definitions in Figure 7. Translocation rules (as differentiated from transformation rules) utilise these *channels* to define the movement of agents and complexes between compartments or voxels at a given stochastic rate. *Channel connectivity* is defined between the left and right hand sides of a compartment reference expression (such as the `:Plane[x][y]` expressions in Figure 7), and may either make use of preset notations such as the 'Neighbour' and 'EdgeNeighbour' connectivities shown or be defined manually by the modeller.

To summarise: Spatial Kappa incorporates into rule-based modelling the specification of voxel-composed compartments and both intra- and inter-compartmental connectivity structures, thus allowing the definition of translocation rules upon explicit spatial constructs. The



A

```
%channel: EdgeNeighbour (:Plane[x][y] -> :Plane[x][y+1]
                       + :Plane[x][y] -> :Plane[x][y-1]
                       + :Plane[x][y] -> :Plane[x+1][y]
                       + :Plane[x][y] -> :Plane[x-1][y])
```

B

```
%channel: Neighbour (:Plane[x][y] -> :Plane[x][y+1]
                   + :Plane[x][y] -> :Plane[x][y-1]
                   + :Plane[x][y] -> :Plane[x+1][y]
                   + :Plane[x][y] -> :Plane[x-1][y]
                   + :Plane[x][y] -> :Plane[x+1][y+1]
                   + :Plane[x][y] -> :Plane[x+1][y-1]
                   + :Plane[x][y] -> :Plane[x-1][y+1]
                   + :Plane[x][y] -> :Plane[x-1][y-1])
```

Figure 7: Examples of two-dimensional connectivity on a regular grid in Spatial Kappa: 4-way EdgeNeighbour (A) and 8-way Neighbour (B). We provide the corresponding textual definition alongside each graphical representation.

current implementation of Spatial Kappa (version 2.1.1) as yet is unable to handle potential features such as compartment volumes and their effect on activity rate, hierarchical topologies and the corresponding implicit compartment links, or dynamic compartments.

1.2 Motivation

Modern understanding of computational biology, as described by Olaf Wolkenhauer [97], is as "an approach to understanding complex, i.e., non-linear spatio-temporal phenomena across multiple levels of structural and functional organisation". However, many current modelling approaches, and the questions they are equipped to ask, are unable to tackle these questions in a succinct and widely-accessible manner. We motivate this thesis by identifying the need to focus further on complex interconnected and evolving 'webs' of biological information in addition to more traditional 'pathways', to better understand how biological interactions and associated experimental data may be reflected *in silico*.

In many cases simple abstract models may be enough to study emergent system properties, up to and including informal diagrams simply identifying the constituent components. For a detailed understanding of the system we often wish to create more complex, more detailed models. Traditional modelling methodologies often do not scale well to this complexity – for example, the Michaelis-Menten assumption often used in systems of ODEs fails to hold in a crowded molecular environment in which agents and their interactions are constrained in space. Methodologies that do begin to scale, such as Kappa, require an infusion of further expertise and effort to utilise, which can often be hard to come by. Furthermore we often fail to properly track the domain knowledge and the abstractions encoded in the model, making it difficult to revise and reuse models as the underlying biological knowledge evolves, and resulting in a glut of single-use, fit-to-purpose models that are abandoned almost as immediately as they are published. The gap between bioinformatics and modelling remains as divisive as ever.

To address the twin problems of defining complex models and making them more accessible and mutable over time, we consider two particular aspects of the current modelling approach in Kappa and Spatial Kappa. Refining and verifying existing modelling formalisms by adopting tenets from related disciplines including biophysics, software engineering, and data visualisation, we pave the way to asking questions that previously could not be asked regarding complex rule-based models in space and over time.

1.2.1 Complex Models in Space

Spatial Kappa addresses the oft-faulty assumption that the reaction mixture of a Kappa model is homogeneous and well-mixed. It does so, however, in a rigid and inflexible manner through the use of voxels engrained in a predefined lattice-like structure. Although we do not wish to adopt

the fully abstract definition of compartments as used in cBNGL, we do recognise its advantages in swiftly and intuitively defining compartments without the necessity of explicitly describing voxel connectivity. We also appreciate that it allows for automatically deriving translocation rates based on biophysical and geometric attributes.

We wish to evolve the current iteration of Spatial Kappa to reflect the needs of the community in terms of future sustainability and computational efficiency. In particular, we wish to verify the operational semantics of the language against established results, and to streamline the process of defining spatial structures by proposing alternatives that are better suited to different modelling purposes. By doing so we hope to lay the foundations for querying a wider range of spatially-oriented systems, with greater confidence in the accuracy of the results. We also wish to propose tentative methodologies for the automated derivation of diffusion rates and of the spatial structures themselves.

1.2.2 Complex Models over Time

Drawing attention to the textual Kappa model displayed in Figure 3, we see how non-modelling specialists might find it difficult to define rules, much less to iteratively verify and analyse them in a rigorous model development process. Current Kappa models are defined declaratively but not as a set of mutable data structures, making them prone to unintentional error and hidden assumptions.

We wish to introduce structured methodologies for developing and maintaining knowledge base data structures built around rule-based models. Said knowledge bases would eventually incorporate not only all of the information needed to construct a Kappa rule-based model, but also annotations in the form of traceable links to bioinformatics databases and other sources of underlying knowledge. We look to provide a graphical alternative to textual definition of the model, via an unambiguous and concise visualisation of agents and rules amenable to editing. We also look to develop further modelling methodologies for use with synthetic genetic parts. By doing so, we wish to provide a framework for new modellers to work with the language, reducing the entry barrier and opening the field to further questions and new research.

1.3 Aims, Approaches, Contributions

In addressing the aforementioned two major problems in this thesis, we raise the following questions.

How can we intuitively define a spatial projection (compartments and channels) upon a set of rules, agents, initial conditions, and perturbations? Furthermore, how can we verify that simulations on this spatial projection are accurate in relation to established diffusivity theory? In the first part of this thesis, we utilise well-understood theory such as the Stokes-Einstein

relation and the narrow escape problem to automatically derive translocation rates based on biophysical and geometric attributes, and then to verify the accuracy of the stochastic approximations made. We then propose redefinitions of how to define space in Spatial Kappa, whether by extending the notion of Kappa-like agents and complexes or by incorporating Cayley graphs for the automated generation of complex spatial structures.

What are the essential ingredients in translating a rule-based model in either Kappa or Spatial Kappa into a mutable data structure amenable to development over time? How might we provide an alternative to the textual representation of a Kappa or Spatial Kappa model based on this theoretical data structure, so as to assist in its definition and visualisation? In the second part of this thesis we propose a graphical representation corresponding to the underlying textual model, and illustrate it with examples introduced throughout the thesis.

How might we then adapt this work to a concrete example in synthetic biology? We use this example to evaluate the methodologies, comparing them to the traditional approach espousing none of the concepts introduced in the thesis. In doing so, we hope to contribute significantly to presenting Kappa as an accessible modelling language with a structured development paradigm and verifiable spatial extensions.

1.4 Thesis Outline

We begin by introducing rule-based modelling in Kappa via a series of sample models. We then tackle the two questions in turn: the evolution of rule-diffusion from reaction-diffusion, and a proposal for model engineering based on the tenets of software engineering. Concretely, the subsequent chapters of this thesis are organised as follows.

Chapter 2: We introduce six Kappa models developed or refined further during the course of this thesis. By doing so we clarify typical problems faced by current models written in Kappa in light of the motivations raised above, focusing on existing issues in visualising and working with such models. We show how even seemingly simple protein-protein interaction models may have complex dynamics and results, and how comprehensive pathway models built over multiple compartments are difficult to visualise, analyse, and maintain. We also venture into the world of modelling in synthetic biology, explaining the difficulties involved in introducing synthetic genetic constructs to protein-protein models.

Chapter 3: We take up the task of verifying the soundness of Spatial Kappa in relation to basic diffusivity theory, including the Stokes-Einstein relation and the narrow escape problem, upon the current implementation of space as a Cartesian grid. In particular, we investigate the mean first passage times of travelling agents in one, two, and three dimensional structures, for different pre-defined Spatial Kappa compartments and with different connectivities, and on boundaries that are both fully absorbing and reflective with a small absorbing aperture.

In doing so, we perform a brief comparison of diffusion models between Spatial Kappa and normal Kappa, and encounter limitations in the current implementation of Spatial Kappa and the issues that arise from them.

Chapter 4: Building on shortcomings identified in Chapters 2 and 3, we introduce refinements to how Spatial Kappa defines and generates space in a rule-based model. We look at an example extension to the existing Spatial Kappa syntax (space as a Cartesian grid) that limits the occupancy of a voxel. Then we propose alternative spatial definitions, including Kappa-like complexes and Cayley graphs, and investigate the advantages of each via hypothetical use cases. For example, assuming a well-defined spatial structure and connectivity, alongside geometric and physical attributes for the compartments and agents involved, we investigate a methodology for automatically deriving rate constants for a model utilising well-defined diffusivity theory. We also consider how we might automate the derivation and definition of spatial structures in respect to these three methodologies. Finally, we also propose a preliminary operational semantics for the alternative spatial definitions.

Chapter 5: Moving on from the implementation of Spatial Kappa, we propose a means for graphically representing a Kappa or Spatial Kappa model. The primary aim of such visualisations is to reduce the difficulty threshold for non-experts in computational modelling in creating and presenting rule-based models. By developing a visualisation with a one-to-one correspondence with agents, rules, initial conditions, and perturbations, we provide an alternative to their textual definition as well as a simplified representation for displaying rules and their effects. We define the components of such a visualisation as they correspond to the underlying model data structure, and illustrate it with an in-depth investigation of the most complex model introduced in Chapter 2, highlighting advantages and disadvantages of the chosen approach.

Chapter 6: Our goal in this chapter is to address the specific field of modelling in synthetic biology. We introduce the current iteration of the Kappa BioBrick Framework (KBBF), which provides a structured framework for modelling BioBrick parts in Kappa, and illustrate it via an example introduced in Chapter 2. Then we adapt the visualisations defined in Chapter 5 to the KBBF, using them to provide the necessary definitions for the framework. The KBBF may thus form a possible structured methodology for overcoming the shortfalls of the visualisations in defining and displaying complex genetic interactions. We take a look at iterative development within this context, and briefly comment on how we might automate the compilation of models created via the KBBF.

Chapter 7: We finish off by addressing shortcomings and future directions identified in Chapters 5 and 6 with regard to working with rule-based models over time. Firstly, we introduce to biological modelling the concept of bidirectional model transformations, allowing users to maintain consistency between sources of model information over multiple user views in the IDE. This gives us a means for synchronising and maintaining the consistency of the data

structure as it evolves through the process of creating and refining a bio-model knowledge base, using graph transformation techniques to perform queries and complex partitions. We illustrate the approach via a small toy example, and present a series of use cases involving the use of bidirectional transformations on a model knowledge base. Secondly, we study improvements to the visualisations introduced in Chapter 5, in particular automating the placement and layout of visualisations, hierarchically linking contact graphs with a chapter overview, and displaying dynamic stories on the static visualisations. We demonstrate these improvements on a range of use cases from Chapter 2. Finally, we take a quick look at future endeavours that may affect the Kappa BioBrick Framework and its visualisations, in particular how we might specify such a model from underlying data: the definition of a Virtual Parts Repository, and the development of synthetic whole cell models. We concentrate especially on work done in the Module Integration Simulator, with one eye on how this may ease the work of future modellers in creating and testing large, complex models over time.

Chapter 8: This chapter reviews the contributions made by, and the deficiencies of, the work within this thesis, and considers future research to address these issues. The latter includes three main foci: implementing and verifying the remainder of our research proposals, clarifying the workflow from bioinformatics to modelling by which we mean them to be used, and actually developing models via the methodologies enshrined in our proposed integrated model development environment. Eventually, we aim to make accessible to a wide range of users a workflow that allows users to specify and modify a rule-based biological model embedded in a well-understood notion of space.

Parts of this thesis have previously been published elsewhere. Much of the underlying material on bidirectional transformations on the Kappa model data structure, as discussed in Chapter 7, was initially published by John Wilson-Kanamori with Soichiro Hidaka in [95]. Of the models developed in Chapter 2, Vincent Danos and Heinz Koeppl have examined the cooperative assembly model in far greater detail in [23]; John Wilson-Kanamori developed an earlier iteration of the mammalian circadian clock in [94], as well as the synthetic switch for John Moore's PhD thesis in [65]; Vincent Danos originally published the model of the MAPK cascade in [21]. Finally, an exploration of an earlier version of the Kappa BioBrick Framework (Chapter 6), via the light-mediated repressilator model presented in Chapter 2, was published in [84] by John Wilson-Kanamori with Donal Stewart. An updated version of this paper is pending publishing as a chapter in the forthcoming book "Computational Methods in Synthetic Biology".

Chapter 2

From Reactions to Rules: Modelling in Kappa

In this chapter we...

- ❖ Introduce six Kappa models developed or refined further during the course of this thesis.
- ❖ Clarify typical problems faced by current models written in Kappa, in light of visualising such models and working with them over time.
- ❖ Show how even seemingly simple protein-protein interaction models may have complex dynamics and results.
- ❖ Show how comprehensive pathway models built over multiple compartments are difficult to visualise, analyse, and maintain.
- ❖ Venture into the world of modelling in synthetic biology, explaining the difficulties involved in introducing synthetic genetic constructs to protein-protein models.

In this chapter we expand on the motivations provided in Chapter 1, highlighting the issues we raised therein by exploring actual models. We work our way through increasingly complex biological systems, focusing on protein-protein interactions before introducing genetic modelling in the context of synthetic biology.

Many of the models in this chapter have been introduced in previous publications. The cooperative assembly model was originally defined and analysed by Vincent Danos and Heinz Koeppl, and modified and verified in simulation by John Wilson-Kanamori [23]. The simple MAPK cascade was introduced in [21] by Vincent Danos et al.; we describe it here in the context of a tutorial for Kappa written and presented over multiple occasions by John Wilson-Kanamori. The mammalian circadian clock model was developed by John Wilson-Kanamori

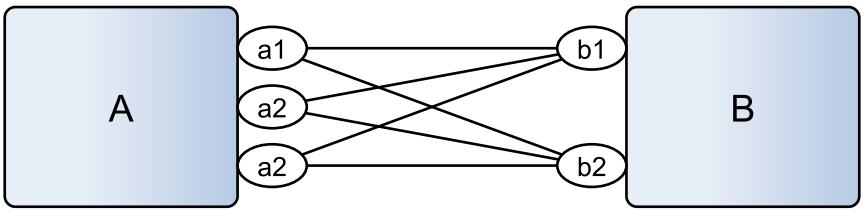


Figure 8: The contact graph of the bipartite cooperative assembly model where $v_A = 3$ and $v_B = 2$.

under the guidance of Vincent Danos [94], based on previous models by Daniel Forger and Charles Peskin [32, 33]. The model of the high osmolarity glycerol web was curated and assembled by Peter Krenn, then translated and extensively modified and analysed by John Wilson-Kanamori. The model of the synthetic switch was developed by John Wilson-Kanamori based on work by John Moore [65]. Finally, the light-mediated repressilator model was originally developed for iGEM 2010 by John Wilson-Kanamori, with the assistance of Donal Stewart and the rest of the University of Edinburgh 2010 iGEM team [84]; the model of the underlying synthetic repressilator was originally crafted by Ty Thomson [88].

The Kappa code for each of the models is provided in Appendix 1.

2.1 Cooperative Assembly

We begin with a simple model of the cooperative assembly of the chemotaxis sensor in *Escherichia coli*. Numerous intracellular molecular systems, including post-synaptic densities and bacterial chemotaxis sensors, form extensive structures intermediate between traditional signalling complexes and large-scale objects from statistical physics. However, the specific regulation controlling their assembly, and how this regulation relates to their information processing capabilities, remains relatively unknown.

We consider a set of assembly models with n_A agents of type `A` and n_B agents of type `B`; each agent is equipped with a set of functionally identical and indistinguishable sites (`A` has $v_A > 0$ sites of type `a`, whilst `B` has $v_B > 0$ sites of type `b`). It is supposed that `a` and `b` can bind and unbind, under the assumption that the binding dynamics depend only on the current occupancy state of the agents that the sites in question belong to. Since Kappa cannot handle indistinguishable sites as described above, we instead model distinguishable sites with identical behaviour, which increases the complexity of the model if not that of the underlying system. In particular, we look at the case where $v_A = 3$ and $v_B = 2$, for which we show the contact graph in Figure 8.

Investigating the phase transition structure of the modelled class of bipartite cooperative as-

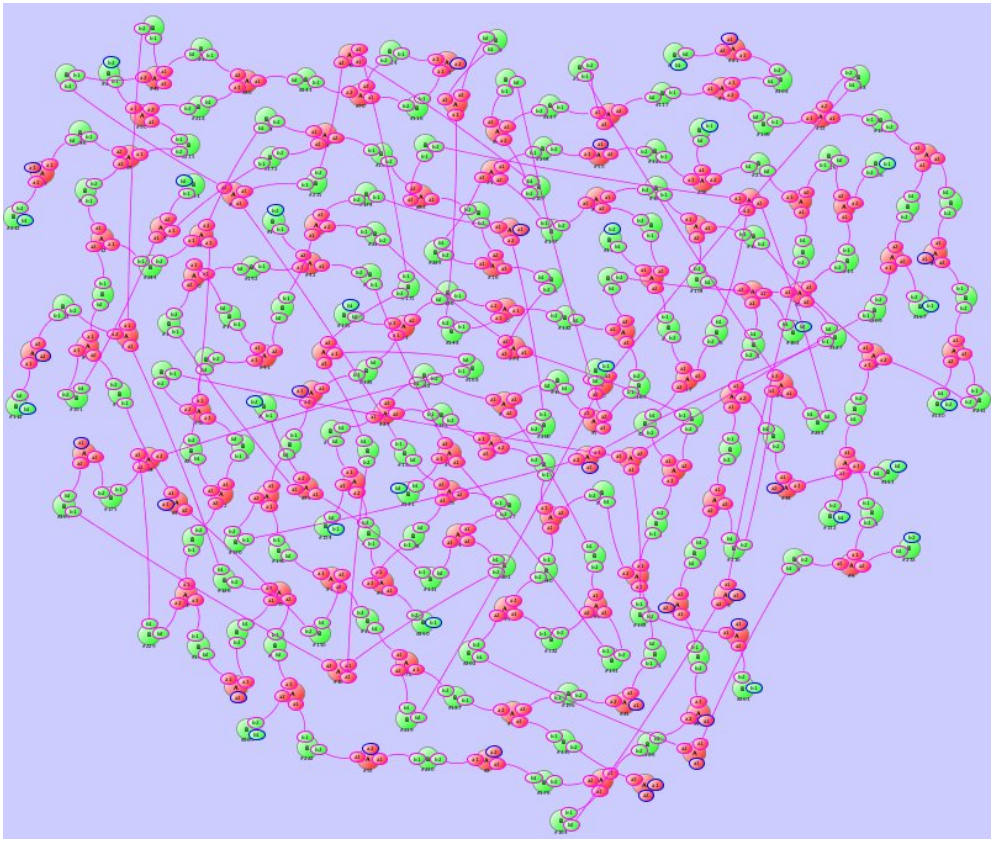


Figure 9: A long-term snapshot from a cooperative assembly system ($v_A = 3$, $v_B = 2$), showing the creation of a single connected site graph with alternating A and B agents; with the parameters chosen, thus, criticality is achieved in the form of a large complex. Snapshot extracted from Kappa Factory.

sembly systems, Danos and Koeppl obtained the conditions for phase transition to a large cluster [23]. Characterising the probabilistic equilibrium for the system, they identified an explicit form for its associated energy functional. Then, drawing on the principles of fragmentation, they extracted a differential system describing the average dynamics of the motifs appearing in the energy functional. In particular, this gave the average steady state concentration as a function of the cooperativity parameters in the system, reducing the phase transition condition to a branching process parameterised by these mean values. The estimates achieved were consistent with analysis of simulations run on the model run by Wilson-Kanamori, a snapshot of which is shown in Figure 9. The work built on previous efforts [22, 26] in attempting to understand related problems.

Ultimately, the model supported the determination of the average steady state concentrations of the partial observables of the system as implicit functions of the cooperativity parameters. Using this methodology, Danos and Koeppl derived a concrete criticality condition for the appearance of a large cluster.

From a modelling perspective, this model shows how simple binding and unbinding between two agents can rapidly spiral into a model of infinite size: the set of species that can be produced by the cooperative assembly model is theoretically unbounded, and practically limited only by the initial populations n_A and n_B. It serves not only as justification for rule-based modelling, since a system with infinite species is impossible to model via classical reaction-based methods such as systems of ODEs, but also raises the question of how one could begin to visualise such a system in a succinctly concise manner (for now we answer this via the contact graph of Figure 8).

2.2 A Simple MAPK Cascade

In Chapter 1, we introduce a simple mechanistic motif wherein a target protein is bound and modified by a kinase and a phosphatase. We now extend and re-arrange this simple motif into a model of the canonical Huang-Ferrell MAPK cascade [47].

The mitogen-activated protein kinase (MAPK) cascade is a signalling cascade known to operate in various forms in many biological pathways. An input signal (ligand) triggers the activation of a MAP kinase kinase kinase (MAPKKK). This activated MAPKKK causes double phosphorylation of its substrate MAP kinase kinase (MAPKK), which then in turn causes double phosphorylation of its substrate MAP kinase (MAPK). The cascade allows a small level of input signal to quickly trigger a large response, the activated MAPK. In one example of such a cascade [21], the GTPase `RAS` activates the MAPKKK `RAF`, which in turn activates the MAPKK `MEK`, finally resulting in activation of the MAPK `ERK`; each intermediate also has a dedicated phosphatase (in order of appearance `PP2A1`, `PP2A2`, and `MKP3`). Because of the ubiquity of such MAPK models within protein biology, and their interconnected binding and modification rules, we often use them in introductory tutorials for modelling in Kappa.

The hierarchical nature of the cascade is reflected in its contact graph (Figure 10). Its rules decompose naturally into five groups, one for each modifiable site. Each of these groups is similar to the rule set for the Goldbeter-Koshland loop considered in Chapter 1, containing six rules (two bindings, two modifications, two unbindings) with three each related to phosphorylation and de-phosphorylation respectively. The first of these groups deals with the activation of `RAF` by `RAS` and the inactivation of `RAF` by `PP2A1`; the second and third with the subsequent activation of `MEK` by activated `RAF` and its inactivation by `PP2A2`; and the fourth and fifth with the final activation of `ERK` by activated `MEK` and its inactivation by `MKP3`. The rules combine to propagate the initial `RAS` signal throughout the system. A sample dynamic trajectory of this process is shown in Figure 11.

Although the contact graph in Figure 10 succinctly displays the agents involved in the model, it fails to convey any rule information beyond which agents (sites) interact with another

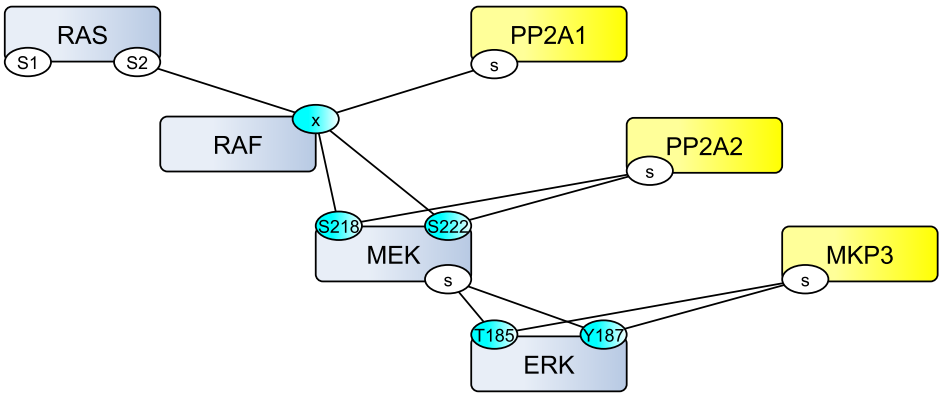


Figure 10: The contact graph of the canonical `RAS` MAPK cascade, modified to show the kinase-target cascade in blue and the phosphatases in yellow. Note the five states amenable to phosphorylation in teal.

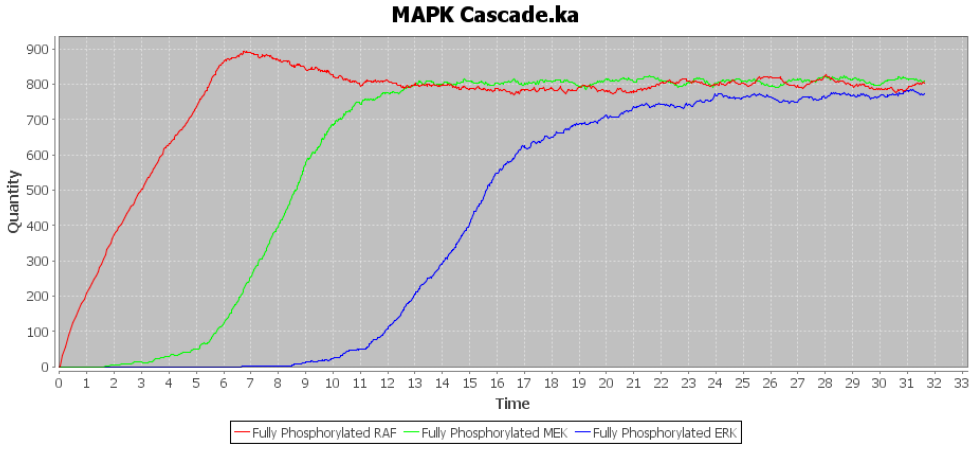


Figure 11: Tracking the number of fully phosphorylated `RAF`, `MEK`, and `ERK` in the model provides a visual representation of the protein activation cascade propagating through the system. Simulation run for 1000 simulation events, with 1000 each of the kinases `RAF`, `MEK`, and `ERK`, 300 active `RAS`, and 200 each of the phosphates `PP2A1`, `PP2A2`, and `MKP3`. Simulation conducted in Spatial Kappa tool version 2.1.1.

and which sites are modified by the effects of such rules. In particular, at first glance it is impossible to trace the causal flow which is the main desired observable. We thus ask the question: how may we modify the base contact graph into something that provides a better description of the dynamics of the model?

Furthermore, in teaching this simple tutorial model, we encounter time and again the difficulty experienced by novice biological modellers in defining the textual code for the model. In our experience, this is not necessarily due to difficulty in grasping the core concepts of rule-based modelling and the Kappa language, but rather due to the fundamental impenetrability of expressing the agents and rules themselves in a textual format (although this in itself is less obfuscating than the equivalent mathematical definition of a system of ODEs). A second question we thus ask is whether we can use the enhanced contact graphs described above as a basis for defining, rather than simply visualising, a model.

2.3 The Mammalian Circadian Clock

We next examine a model originally developed for John Wilson-Kanamori's Honours dissertation at the University of Edinburgh in 2010 [94]. The mammalian circadian clock has been the subject of much study over the years, but only recently have advances in science provided a detailed picture of what actually occurs in this complex system. The dissertation focused on constructing a stochastic rule-based model of the mammalian circadian clock in Kappa, transcribed from interactions encoded in existing deterministic and stochastic models (systems of ODEs) of the circadian clock [32, 33]. We benchmarked the model against its existing counterparts to determine the accuracy of the transcribed results, and investigated how mutant versions of the clock affected the robustness of the circuit in comparison to established results throughout related literature.

The model (Figure 12) is based on an oscillating negative feedback loop, in which the PER1 and PER2 PERIOD proteins transport the CRY1 and CRY2 CRYPTOCHROME proteins into the nucleus of the simulated cell, where the transcription factor complex inhibits the production of its constituent proteins. A second negative feedback loop, centred on the REV-ERBα protein, reinforces this oscillation. Finally, light perturbations induce the transcription of PER1 and PER2 independently of the transcription factors upon the genes, reinforcing the 'reset' of the clock. In whole, the wildtype model contains 46 rules and two sets of perturbations.

In simulation we verify oscillations at roughly 24-hour frequencies (Figure 13). However, the peaks of the oscillations vary from the targets specified in the initial conditions (which in turn are derived from an 'average' value of the deterministic model); for example, the maximum values of the CRY proteins fail to achieve their initial peaks throughout the simulation. This highlights some of the difficulties involved in direct translation from one modelling sub-

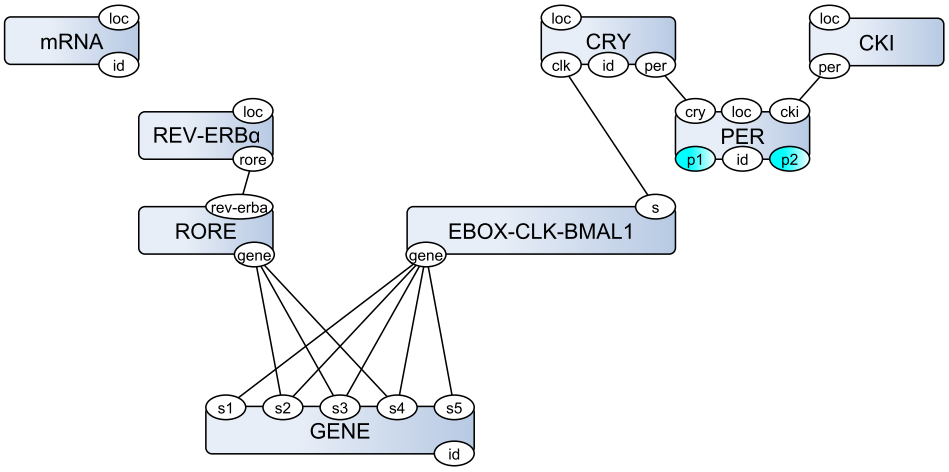


Figure 12: The contact graph of the mammalian circadian clock. Note in particular the '`loc`' site that we use to model the location of the agent within the simulator in non-Spatial Kappa syntax. In the case of this model, the '`loc`' site can take either '`cytosol`' or '`nucleus`' state.

discipline to another. We touch upon this issue once again in Chapter 7.

We establish via evaluation of the model that its accuracy (in statistical terms) and predictivity (in terms of mutant modelling) is at least comparable to the differential equations underlying it. On the other hand, we also identify limitations in the Kappa language throughout the modelling process: difficulties in expressing the spatial location of the entities involved, a lack of expressiveness limiting the conciseness of the rules, and a deficiency in terms of the tools available to prospective modellers. For example, the modelling process would have been greatly eased by access to a curated repository of kinetic rates, or the ability to link directly to bioinformatics protein databases.

In particular, this model was one of the first complex systems written in Kappa to highlight the necessity of spatial structures in biological modelling, given that the model dynamics relied heavily on translocation of the `PER` and `CRY` proteins to the nucleus. Practices of the time (modelling space as a state on a non-functional site as commented in Figure 12) were incapable of succinctly defining such space, requiring multiple similar translocation rules that bloated the model without possibility of simplification. Due to these characteristics, the model was later also used to test the development of the Spatial Kappa extension. However, revisiting the model in this manner raised further issues regarding the compatibility of the syntax between different versions (nearly the entire model had to be rewritten despite the core interactions remaining constant) as well as traceability of the encapsulated data (the modeller had to manually re-verify the kinetic rates and design choices made based on the references provided). In this way, the model sowed the seeds of a Kappa model representing not only an implicit definition for simulation, but also as an explicit (update-friendly) repository of biological data.

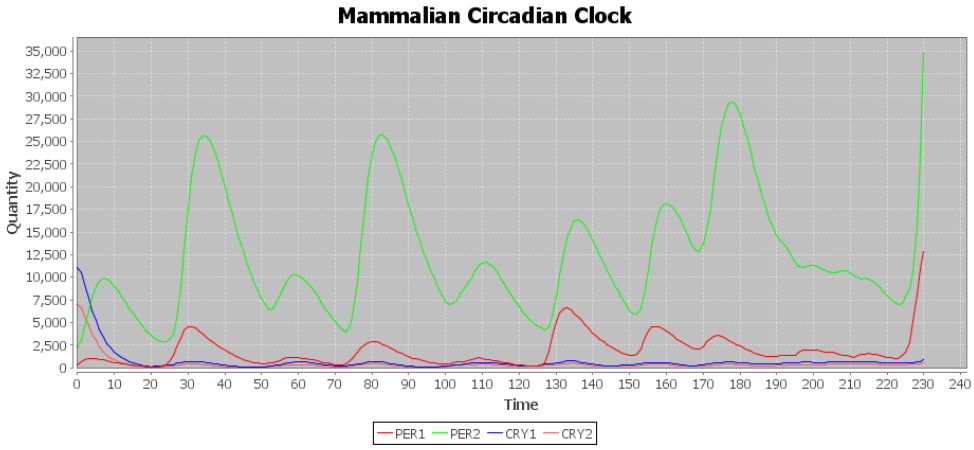


Figure 13: A sample simulation trajectory of the mammalian circadian clock. We observe oscillations in the primary circadian proteins at roughly 24-hour intervals. Simulation conducted in Spatial Kappa tool version 2.1.1.

2.4 The High Osmolarity Glycerol Web in Yeast

We next discuss a complex, highly detailed model of the adaption of yeast (*Saccharomyces cerevisiae*) to external osmotic change. Osmotic shock initiates an increase in external osmotic pressure and a decrease in internal turgor pressure of a yeast cell, compelling it to re-establish balance via the high osmolarity glycerol (HOG) pathway. More specifically, well studied mitogen-activated protein cascades (similar to the Huang-Ferrell example discussed above) serve to activate the `HOG1` protein, which then translocates to the nucleus and initiates a genetic response. The gene expression is reported to vary depending on the intensity of osmotic stress, producing a bimodal expression behaviour within a cell population [68]. A detailed Kappa model of the HOG pathway in yeast, based on both pre-existing models [51, 53] and on literature evidence curated by Peter Krenn, aimed to establish a platform for *in silico* experiments to analyse this bimodal response.

The model (a modified contact graph of which is shown in Figure 14) comprises 41 agents and 540 rules detailing the `SLN1` and `SHO1` branches of `HOG1` activation as well as the downstream proteomic and genetic responses including a fluorescent reporter. Each of the two osmosensors at the cell membrane activate a MAPKKK (`STE11` and `SSK2/22` respectively), which bind to a common MAPKK `PBS2`. `PBS2` then doubly phosphorylates the MAPK `HOG1`, initiating a protein response and also translocating to the nucleus and acting as a transcription factor. In the meantime other proteins work to de-phosphorylate active `HOG1`, switching off the osmotic response in the absence of further shock.

Each of the two pathways show a characteristic response to osmotic stress. The `SLN1` pathway is a two-component phosphorelay that dominates the kinetic response with swift induction but also fast-acting negative feedback, whereas the `SHO1` pathway is a more complicated system

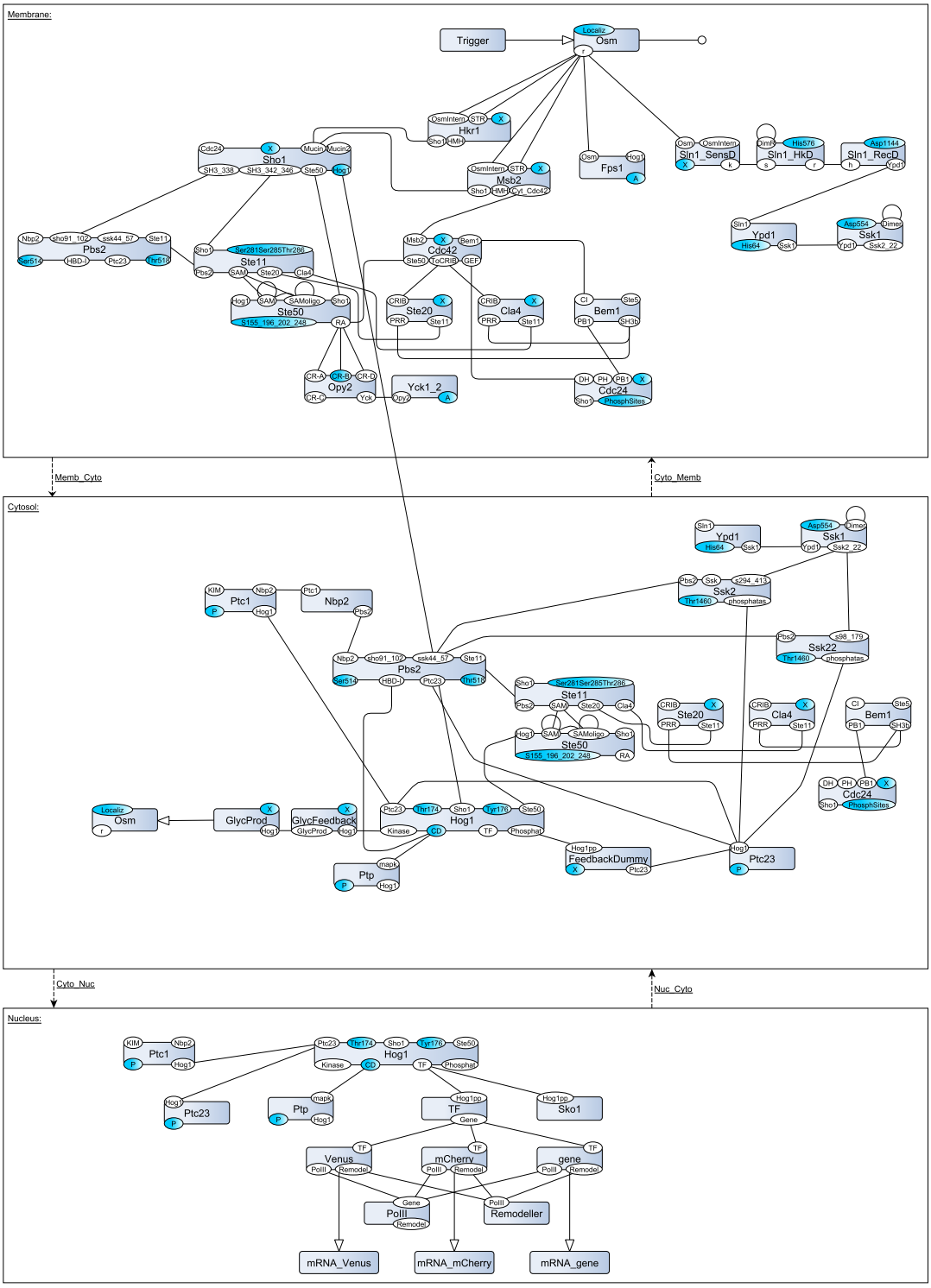


Figure 14: The contact graph of the high osmolarity glycerol web in yeast, adjusted for clarity to explicitly represent the compartments present in the model. Note that not all agents in the model represent proteins that actually exist, nor do all sites represent actual domains. For example, `FeedbackDummy` and `GlycFeedback` are incorporated as placeholders for osmotic feedback mechanisms that are as yet not fully studied. Note also the presence of a inter-compartmental link indicating the possible binding of cytosolic `Hog1` with membrane `Sho1`.

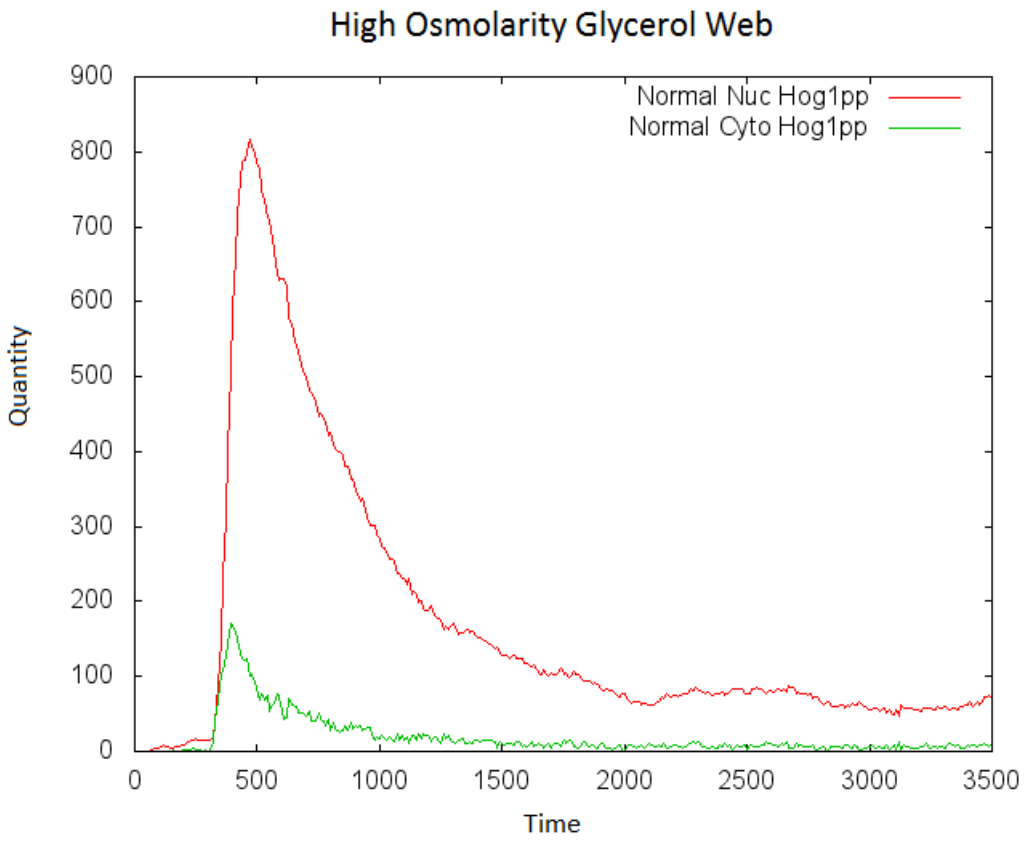


Figure 15: A sample simulation run of the high osmolarity glycerol web in yeast, stimulating the system with osmotic shock at arbitrary time $T = 300$. We observe a swift initial response to the stimulus, followed by a long tail. Simulation conducted in KaSim 2.0.

where adaptor proteins control signal flow into a slower-acting pathway with long-lasting effect. We observe these characteristics in simulation time series (Figure 15), where the `HOG1` output has a high initial response but also a long trailing tail. Such a response curve also matches the existing ODE simulations of the system, as well as experimental verification of the actual biology [51].

Immediately we see that this is another example where local interactions in global space play an important role in the dynamics of a model, thus further advancing the argument fora dedicated spatial extension to Kappa. Furthermore we note the sheer size and combinatorial complexity of the model. According to fragmentation analysis performed by Tatjana Petrov, the phosphorylation and binding states of the agents participating in the cascade create over 10^9 individual species. Simply defining a model on this scale is a daunting task, involving many hours of literature curation (undertaken in this case by Krenn). Furthermore, much of this work is lost in the actual model definition proper, as we currently lack any method (bar code comments, which Krenn declined to use as 'clunky') to annotate the model with the information sources. Simulating it is another difficulty, with even the simplest simulations running for days

and even weeks. The need for iterative development and modification, as well as dynamics and perturbation analysis, adds a further layer of complexity to working with such models due to the sheer number of parameter and observation variables involved and the hidden complexity caused by secondary and tertiary interactions.

Similar to the mammalian circadian clock above, we again encountered time-consuming difficulties when translating the model between different tools and versions (i.e. Kappa to Spatial Kappa) of the language. In particular, whilst Krenn developed the model in an obsolete iteration of the tool and language for the graphical support thus provided, we found it necessary to upgrade the model syntax to the latest version to perform scalable perturbation analysis. Both translation and verification for this purpose needed to be performed manually, which substantially reduced modeller productivity.

Most importantly, however, we see in this model the beginning of a paradigm shift away from linear pathways to complex, interconnected, evolving webs of protein and genetic interactions. In fact, we could take this to its logical conclusion by merging with this model a Kappa model of the yeast pheromone pathway [85, 89], creating one large complex model incorporating two of the major MAPK cascades in yeast. This would allow us to ask new questions such as the effect of combined perturbations (osmostress followed by pheromone alpha factor), although we would also have to deal with related issues of model compatibility during amalgamation (non-unique agent and site names, the difference in underlying model timescales and thus rate parameters, etc.).

We currently lack a unified development framework in which to enact such a bold move. Correctly developed, such a framework would allow us to track the domain knowledge within the model over time, to switch between different model representations with ease, and to iteratively maintain a model as the underlying knowledge base evolves. Although we are still in the process of defining its necessary components (which we explore in greater detail in Chapter 7), in Chapter 6 we especially make use of alternate model representations to visualise and better understand the model.

2.5 A Synthetic Switch

To introduce Kappa modelling to synthetic biology, we begin by applying it to a real-life example of a synthetic switch. Concretely, we use inducible promoters to control the ultrasensitive toggle-like properties of the Goldbeter-Koshland loop, represented previously in Chapter 1 by generic `Kinase` and `Phosphatase` agents. In the model to follow these are `TRXh5` and `GSNO` respectively.

The constitutively expressed `NPR1` transcription cofactor is an integral coordinator [92] of the multi-layered cellular defence system in Arabidopsis. In resting cells, it is subject to

S-nitrosylation at a known amino acid (Cys156) by the GSNO protein acting to promote the assembly of the NPR1 oligomer. Modification to the intracellular redox environment by invasive pathogens, and the subsequent action of the enzymes TRXh5 and NTRA, releases the NPR1 monomer instead. Monomeric NPR1 translocates to the nucleus and interacts with the TGA subclass of transcription factors essential for activating immune defence genes, thus stimulating the cell's genetic response to pathogen invasion.

Investigating this plant immune cascade allows us to identify system function influencing the outcome of defence signalling. In particular, much of the processes surrounding NPR1 function are not fully understood. Work by John Moore [65] approaches this problem by creating a synthetic protein circuit based on a theoretical interpretation of plant immunity, using a *Saccharomyces cerevisiae* yeast chassis engineered to be amenable to redox manipulation. By designing a synthetic circuit (Figure 16) with selectively inducible inputs in the form of TRXh5 and the GSNO reductase GSNOR, Moore specifically promotes the reduction of NPR1 to its monomeric form. This creates a unidirectional 'on' switch with the standard luciferase reporter gene as the circuit output. It is not a true toggle switch as described by Gardner et al. [36], since there is no inducible mechanism for directly turning the circuit 'off' again, but rather a gradual return to the pre-induction state as the inducible inputs filter out of the system.

In vivo studies of the synthetic circuit have determined that it takes two hours for GSNOR/TRXh5 to appear once induced, four to five hours for the threshold level of de-nitrosylated NPR1 to be reached, 90 minutes for the cytosolic oligomer to convert to a nucleic monomer, and a further two to three hours for LUC protein creation. Unfortunately these abstract observations are insufficient in themselves to fully specify the kinetic rates that determine model dynamics. As a first step we simply guess at these rate constants in the actual model, although simultaneously we maintain the initial protein populations faithfully to observed biology. As a result, simulations of the model (Figure 17) display highly qualitative dynamics that are useful for tracking causality but not for reproducing the *in vivo* temporal behaviours described above.

We thus highlight here the growing need for structured repositories for working in synthetic biology, where we can begin to determine such parameters for modelling usage. The ability to switch from Kappa to other methodologies more amenable to parameter search algorithms (for example, genetic algorithms on systems of ODEs) would be useful too. We also point out again the added complexity in working with designed genetic components in a system, where the delay involved in translation and transcription is not simply an observable factor but often a crucial stage in the feedback process. Hence we must pay careful attention to another set of parameters in the model other than protein interaction and modification.

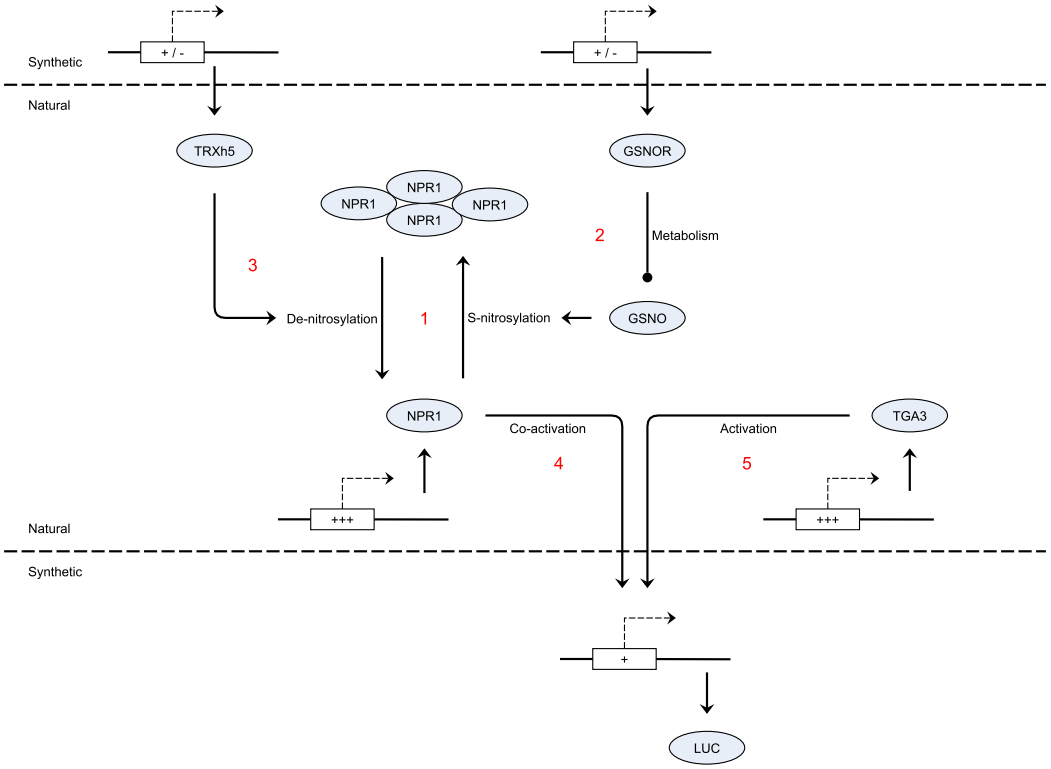


Figure 16: A schematic of the unidirectional synthetic protein switch. The labels on the promoters indicate inducible by an outside chemical compound (+/-), constitutively expressed (+++), and activated as pathway output (+) respectively. Although the system itself is natural (albeit transcribed from Arabidopsis to yeast), the inducible promoters and the reporter are synthetic. Note the structural similarities to the basic kinase-phosphatase model – a target (NPR1) that switching between conformations (oligomeric to monomeric) under the influence of TRXh5 (forwards) and GSNO (backwards). 1: Under resting conditions, GSNO promotes oligomer formation of constitutively expressed NPR1. 2: Selectively activating GSNOR increases the rate at which GSNO is metabolised, thereby reducing S-nitrosylation of NPR1 and indirectly limiting oligomer formation. 3: Activation of TRXh5 (together with NTRA) de-nitrosylates the NPR1 oligomer and promotes monomer release. 4: Monomeric NPR1 interacts with constitutively expressed TGA3, readying it for activity as a transcription factor. 5: This then activates the expression of a LUC reporter gene.

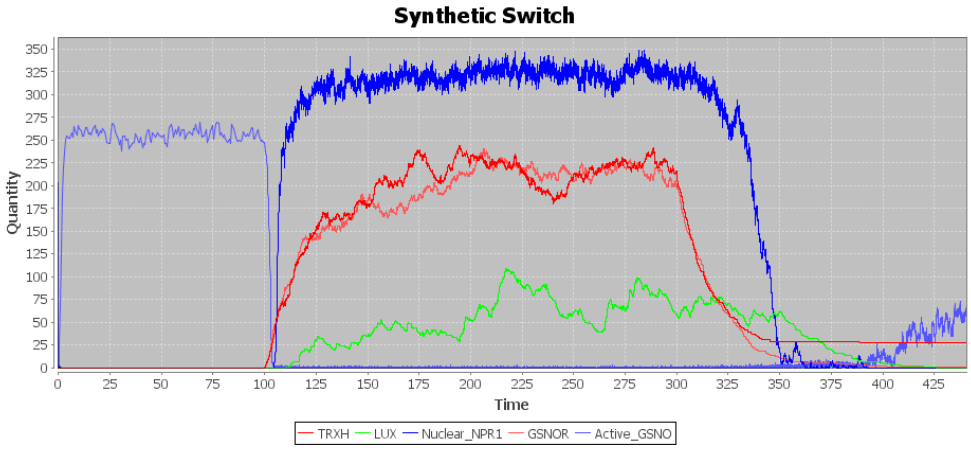


Figure 17: Stochastic simulation of the `NPR1` oligomer-to-monomer switch modelled in Kappa. Plot demonstrating that nuclear-localised monomeric `NPR1` and subsequent reporter gene expression is dependent on induction of `GSNOR` and `TRXh5`. `GSNOR` and `TRXh5` are switched on at simulation time $T = 100$ and off at $T = 300$. Units (time and quantity) are arbitrary. Simulation conducted in Spatial Kappa tool version 2.1.1.

2.6 Light-Based Communication in *E.coli*

We address some of the problems highlighted in the model of the synthetic switch with a Kappa model of the hallowed Elowitz repressilator [29] created from BioBrick parts [74]. BioBrick parts are a concept in synthetic biology allowing the construction of modular genetic systems designed for an artificial purpose. Based on BioBrick parts, we make full use of a modelling methodology we call the Kappa BioBrick Framework to structure the genetic components of the system and to simplify the definition of their stochastic rates of interaction and modification. We explain this framework in greater detail in Chapter 6; for now, we simply concentrate on an abstract view of the model and its results.

The repressilator combines three simple synthetic genes connected in a loop (Figure 18), such that the product of each gene represses the next gene in the loop and is in turn repressed by the product of the previous gene. The output is oscillatory as the repression of one protein by the second allows the third to build up to repress the second, which in turn allows the first to accumulate to repress the third, and so forth.

Our model consists of seven types of agents (`DNA`, `RNA`, `Ribosome`, `RNAP`, and the three repressor proteins involved) and 61 rules. The agents and rules are composed as described in the previous paragraph for eight BioBrick parts: three protein coding sequences (`LacI`, `TetR`, and `λ-CI`), their corresponding promoters, and one each of a terminator and a ribosome binding site (RBS) Each promoter is designed to be cooperatively bound by up to two of its associated transcription factors (in this case a repressor), and thus is modelled as a concatenation of four DNA agents (two dedicated to repressor binding, one for RNAP binding, and a linker or spacer

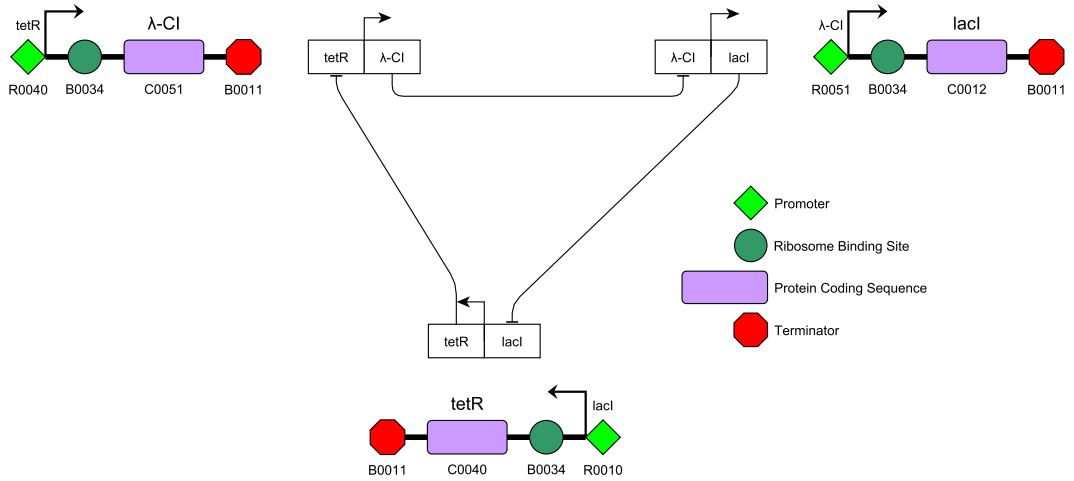


Figure 18: The Elowitz repressilator constructed from BioBrick parts, with each repressor inhibiting production of the next repressor in the loop through the action of the appropriate promoter. Each part may be represented as a modular component (a set of agents and associated rules) in the Kappa BioBrick Framework, and the composition of a number of these parts creates a full circuit as shown. The central representation hides the ribosome binding site and terminator components of the BioBrick device, since we reuse the same parts throughout the model.

separating the promoter from any preceding part). Every other BioBrick part is modelled as a single `DNA` agent. The terminator and RBS are reused across all three devices. We obtain the characteristic oscillation of the repressilator as shown in Figure 19.

One of the weaknesses of the Elowitz repressilator is that without external regulation the system oscillations are extremely imprecise and do not last over time, so that the oscillations of multiple cells each running their own version of the repressilator rapidly de-synchronise or die off. To address this problem, the 2010 Edinburgh iGEM team designed a light-mediated communication system based on the Elowitz repressilator (Figure 20), involving the establishment of three independent channels of communication in different spectral wavelengths [84]. The goal of the project was to develop a multi-cellular system capable of self-reinforcing collective synchronisation, with the eventual aim of enabling bacterial populations to interact with each other as well as with purely electronic systems via light.

Each gene product in the repressilator loop is used to repress the production of light of a particular wavelength. In other words, each wavelength is associated with the lack of an associated repressor (red light with lack of `LacI`, blue light with lack of `TetR`, and green light with lack of λ`-CI`). In the meantime, the light sensing pathways provide input to the core repressilator to reinforce the oscillatory response. The activated green and blue sensors explicitly inhibit `LacI` and λ`-CI` respectively, while the activated red sensor ceases promotion of `TetR`, hence inhibiting it implicitly. If the effect of the light emitting pathways is to ‘broadcast’ the current

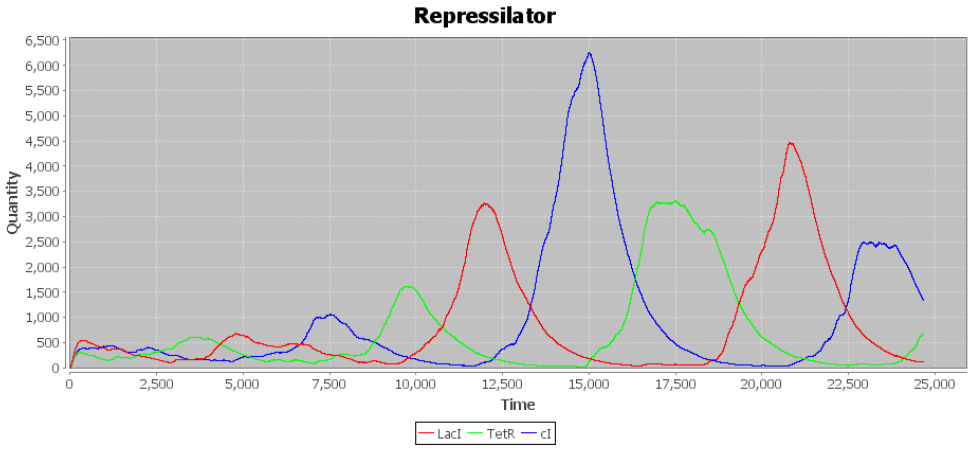


Figure 19: Simulation results for the Elowitz repressilator modelled using the Kappa BioBrick Framework. The system stutters to begin with due to the initial conditions of the model favouring none of the three repressors, but as soon as one begins to assert dominance the system falls into its familiar oscillating pattern. Simulation conducted in Spatial Kappa tool version 2.1.1.

state of the host's repressilator to its neighbours, the proposed effect of the light sensing pathways is to 'adjust' its state to match those of its neighbours instead. When activated, ideally the sensors either reinforce the current state of the repressilator in the host (because it is already synchronised with its neighbours) or modify it to bring it closer to the desired behaviour. In this way, we hope to synchronise cells running individual instances of the repressilator such that they express the same output wavelength. This communication occurs at a much faster rate than the transcriptional interactions of the core repressilator, but it is unclear at this stage (when approaching this system guided only by intuition) whether or not this synchronisation will actually occur if implemented *in vivo*. Modelling the light sensors and emitters in conjunction with the core repressilator, and analysis of the proposed system as a whole, is thus crucial to understanding its dynamics.

This model of light communication is thus composed from seven components (the core repressilator, along with emitters and sensors for the three wavelengths), each of which we may consider a functional model in its own right. We developed the model iteratively, validating individually before combining modules into a single host, in a manner which we suggest serves as a basis for merging, maintaining, and updating a biological knowledge base data structure. We explain this process further in Chapter 6, and look at methodologies to support it in Chapter 7.

We extend this intracellular model by simulating the behaviour of an idealised virtual colony of bacteria communicating with each other using the light produced within each cell. This is achieved via the simplifying assumption that the bacteria are non-motile and closely packed in a two-dimensional hexagonal biofilm, and we leverage a custom implementation of

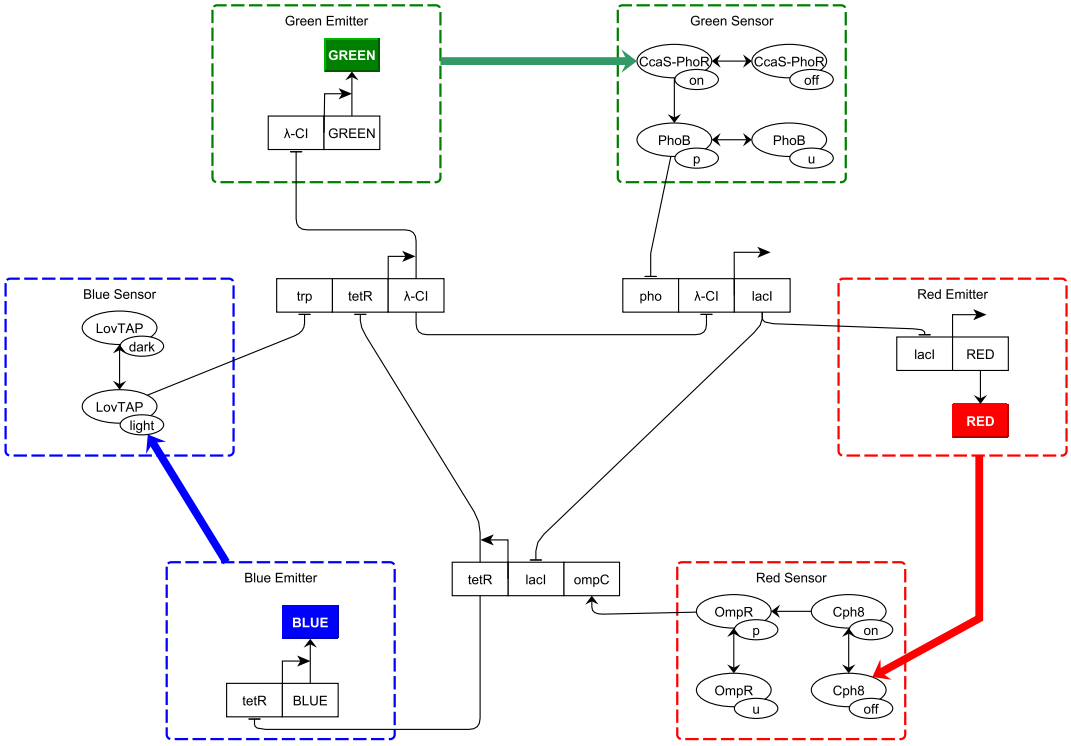


Figure 20: Modelling light emitting and light sensing pathways coupled to an Elowitz repressilator. At the centre the oscillating repressilator regulates the emission of light in the system. The light sensing pathways then provide input to this central regulator via a second promoter coupled to the same coding sequence, reinforcing or adjusting responses to synchronise the system. The central repressilator and the light emitting pathways are genetic components, built solely around the Kappa BioBrick Framework, but the light sensing pathways are protein interaction components that must need be manually specified by the modeller.

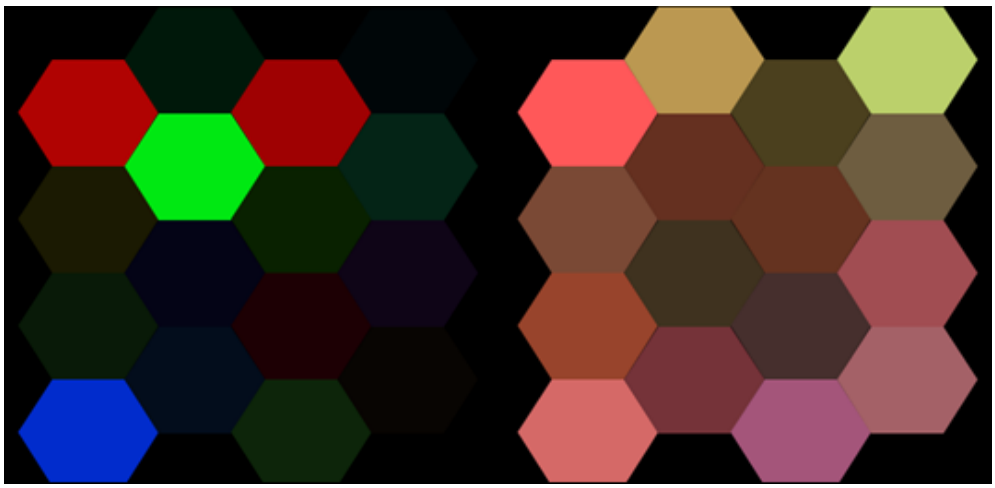


Figure 21: Modelling light emitting and light sensing pathways coupled to an Elowitz repressilator. At the centre the oscillating repressilator regulates the emission of light in the system. The light sensing pathways then provide input to this central regulator via a second promoter coupled to the same coding sequence, reinforcing or adjusting responses to synchronise the system. The central repressilator and the light emitting pathways are genetic components, built solely around the Kappa BioBrick Framework, but the light sensing pathways are protein interaction components that must need be manually specified by the modeller.

Spatial Kappa to provide us with this spatial definition and visualisations of its results. Additional rules to the model represent the communication of light between neighbouring cells, each responsible for maintaining its own repressilator. A snapshot of a sample simulation is shown in Figure 21, with isolated non-communicating cells shown on the left and communicating cells on the right.

The light levels in each cell are recorded at each sample point during the simulation, along with the colony mean light levels and the standard deviation of the individual cell light levels from these colony means. Accurate average behaviour is recorded by measuring results (Figure 22) over sufficiently long simulations. These results show that a communicating colony has less time-averaged deviation in light levels between cells, and therefore increased synchronisation.

Again, we revisit this model and the underlying Kappa BioBrick Framework in greater detail in Chapter 6 of this thesis. We also revisit the iterative development of the model in light of further advances we make in the concept of model engineering (as related to software engineering) in Chapter 7. For now, we return to models in space, and further explore the Spatial Kappa extension for the Kappa stochastic rule-based modelling language.

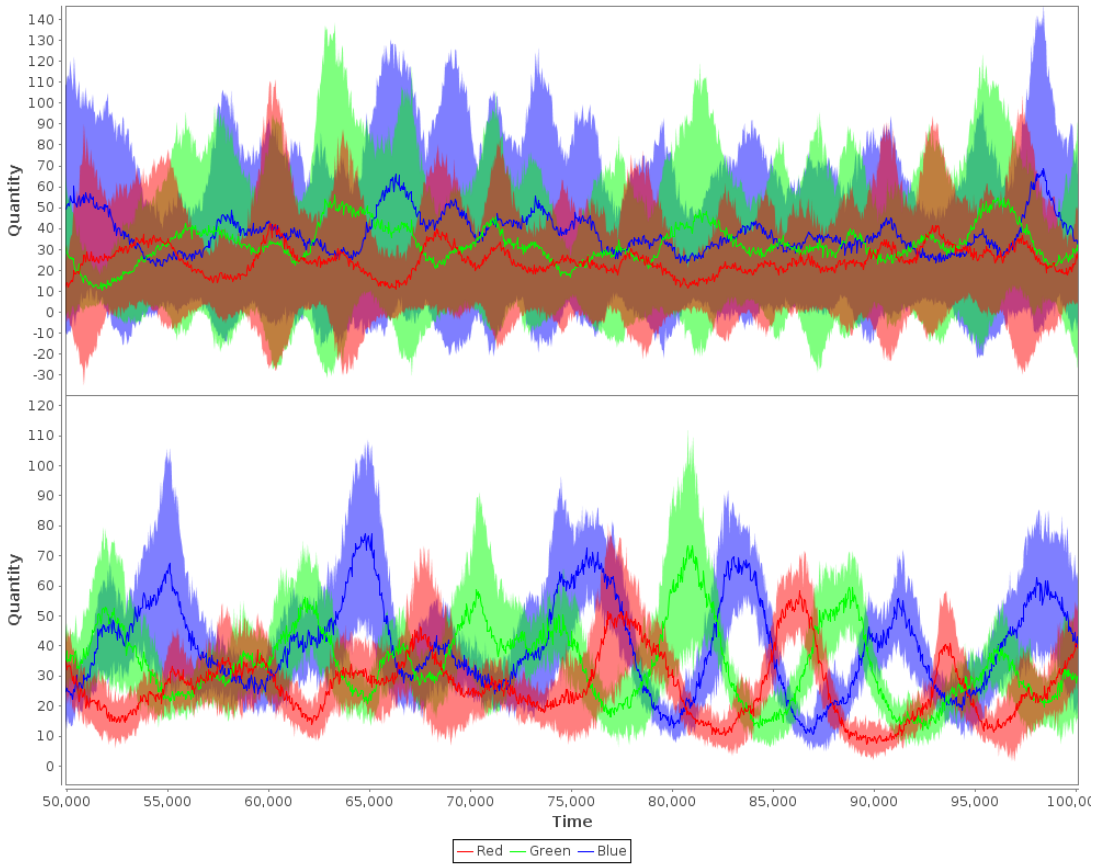


Figure 22: Comparison of mean cell light levels in 4x4 colonies of cells, both isolated (top) and communicating between cells in the colony (bottom). Shaded areas on the graphs show standard deviation of cell light levels from colony mean. Communicating colony shows lower standard deviations, and hence increased coherence in cell activity levels. Units (time and concentration) are arbitrary. Simulations conducted in Spatial Kappa tool version 1.0 (output modified manually to display mean and standard deviation).

Chapter 3

Verifying Space as a Cartesian Grid

In this chapter we...

- ❖ Test the limitations of the current implementation of Spatial Kappa and its representation of space as a Cartesian grid of voxels.
- ❖ Review the Einstein-Stokes relation and the narrow escape problem for the mean first passage time of diffusing particles.
- ❖ Perform a brief comparison of simple diffusion models across Spatial Kappa, normal Kappa, and normal Kappa using tokens.
- ❖ Investigate mean first passage times for travelling agents exploring various spatial structures in one, two, and three dimensions until they arrive at a fully absorbing boundary.
- ❖ Investigate the narrow escape problem for travelling agents exploring various spatial structures in two and three dimensions until they arrive at an absorbing aperture set in a reflecting boundary.
- ❖ Alter the origin of a travelling agent on a two-dimensional disk and explore how this affects the dynamics of fully absorbing diffusion and the narrow escape problem.
- ❖ Investigate how we might scale a diffusion problem across different voxel size resolutions, and the cost of doing so.
- ❖ Encounter and explore the 'jumping' problem in which the travelling agent arrives at the absorbing boundary in pulses conditioned to simulation time $t = \frac{1}{4}, \frac{1}{2}, 1, 2, 4, 8...$, and posit that this is a bug that needs to be further investigated.

Developed by Donal Stewart and Vincent Danos as an extension to the core Kappa language in summer 2010 and 2011 [83, 84], Spatial Kappa aims to combine the strength of rule-based

modelling (alleviating combinatorial complexity between interacting biological entities) with a spatial awareness that the underlying Kappa language has hitherto lacked but is useful for developing and simulating complex, realistic biological models. The work has since been adopted by Oksana Sorokina, Anatoly Sorokin, and J Douglas Armstrong [81] towards a quantitative model of the post-synaptic proteome [80]. We introduced the basic tenets of Spatial Kappa in Chapter 1.

Despite this development process, no effort has yet been made to formally verify and validate the soundness and correctness of the Spatial Kappa simulator and its underlying algorithms. We seek to address this outstanding question within this chapter. In particular, we wish to calibrate the spatial aspects of the language by reproducing standard theoretic results in diffusion.

We begin with the seminal results embodied by Fick's Laws. In particular we work with Fick's Second Law, which predicts how diffusion causes the concentration of a particle to change with time. Fick's Second Law states that:

$$\frac{\partial \phi}{\partial t} = D\frac{\partial^2 \phi}{\partial x^2} \tag{3.1}$$

where ϕ is the concentration, t is time, D is the diffusion coefficient, and x is the position. The accumulation or depletion of concentration within the volume is proportional to the local curvature of the concentration gradient.

An intuitive derivation follows: let us assume a one-dimensional realm of diffusion. Given position x, we might expect the following changes (normalised for D, where $f(x)$ implies the flow into position x at time t):

$$\begin{aligned} \tfrac{1}{D}\tfrac{\partial \phi}{\partial t} &= -2f(x) + f(x-dx) + f(x+dx) \\ &= (f(x+dx) - f(x)) - (f(x) - f(x-dx)) \\ &= \tfrac{\partial^2 \phi}{\partial x^2} \end{aligned} \tag{3.2}$$

Based on Fick's Second Law, the Stokes-Einstein relation gives us a formula for how long a diffusing particle takes to cover a specified distance x (the mean squared displacement of said particle). For d dimensions, this is:

$$t = \frac{x^2}{2dD} \tag{3.3}$$

where x is the distance to be diffused, D is the diffusion coefficient, and t is the time required. Alternately we may rewrite this as:

$$x = \sqrt{2dDt} \tag{3.4}$$

which represents the typical distance covered by a particle in d-dimensional continuous space in time t. This corresponds to the standard deviation of a Gaussian proportional to a solution to the Fokker-Planck equation for Brownian motion.

To use these equations as a benchmark for the performance of Spatial Kappa, we must first relate the diffusion coefficient D to its stochastic translocation rate γ for a given model. D has units of $\frac{length^2}{time}$ and is proportional to the squared velocity of the diffusing particles (agents), which in turn depends on temperature, viscosity, and the size of the particle. For biological macromolecules the diffusion coefficients normally range from 10^{11} to 10^{10} $\frac{m^2}{s}$. For ease of calculation, we adopt the convention that diffusion may be approximated by the rules of Brownian motion.

Throughout this section, we run simulations in Spatial Kappa 2.0.9. In reference to the next chapter, our models are defined using the currently implemented representation of space as a lattice-like Cartesian grid.

3.1 In One Dimension

Let us derive the stochastic translocation rate γ from the conventional diffusion coefficient D in the simple one-dimensional case of a linear array of identical voxels.

We begin by applying the central limit theorem to a random walk X_i (considered as a sequence of independent and identically distributed random variables that determine the next step) upon a theoretically unbounded diffusive realm. The central limit theorem allows us to state that after large n steps, the random walk converges:

$$\sum_{1}^{n} X_i \sim n \cdot E(X) + \sqrt{n} \cdot N(0, V(X)) + o(\sqrt{n}) \tag{3.5}$$

Taking $X(n)$ as the sum of n independent identically distributed copies of the random walk X_i, where $N(0,V(X))$ is a random variable distributed with the following properties:

- $E(X(n)) = 0$, since the expectation for an unbiased random walk is that the particle will return to the origin.
- $V(X(n)) = E(X(n^2)) - E^2(X(n)) = (2R_{vox})^2$, where R_{vox} represents the radius of a voxel.

Hence, based on Equation 3.5, we may approximate $X(n)$ as:

$$X(n) \sim \sqrt{n} \cdot N(0, (2R_{vox})^2) = N(0, n(2R_{vox})^2) \tag{3.6}$$

We now wish to equate Equation 3.6 to the expected results of a Continuous-Time Markov Chain. The voxel exit rate γ^* of this CTMC is the sum of all rates of translocation exiting a given voxel. For the regular grid-like structures as considered in this chapter, $\gamma^* = \gamma \cdot c$ where c is the number of neighbours to the voxel in question. In the infinite limit, n may be approximated in a stochastic process by the voxel exit rate multiplied by the simulation time ($n \sim \gamma^* t$). Hence:

$$X(t) \sim \sqrt{\gamma^* t} \cdot N(0, (2R_{vox})^2) = N(0, \gamma^* t (2R_{vox})^2) \tag{3.7}$$

In this CTMC, we expect a typical trajectory to attain a value equivalent to the standard deviation of the Gaussian distribution of $X(t)$ in Equation 3.7. Equating this with the typical travelling distance provided in Equation 3.4 for a particle in one-dimensional space gives us the following:

$$\begin{aligned} \gamma^* t \cdot (2R_{vox})^2 &= 2Dt \\ \gamma^* &= \tfrac{2D}{(2R_{vox})^2} \\ \gamma = \tfrac{\gamma^*}{c} &= \tfrac{2D}{c(2R_{vox})^2} \end{aligned} \tag{3.8}$$

In this way, given a known diffusion coefficient D, a constant voxel radius R_{vox} for a grid-like spatial structure, and a defined connectivity with a regular degree c, we can deduce the appropriate stochastic rate constant γ.

3.1.1 Linear Compartments vs. the Stokes-Einstein Relation

For ease of simulation, let us assume a travelling agent `T` with diffusion coefficient $D = 0.084\frac{\mu m^2}{s}$, travelling across $n = 10$ voxels with total length $x = 0.1\mu m$ such that $R_{vox} = 0.005\mu m$.

$$\begin{aligned} \gamma^*_T &= \tfrac{2 \cdot 0.084}{(2 \cdot 0.005)^2} = 1680.0 \\ \gamma_T &= \tfrac{\gamma^*_T}{2} = \tfrac{1680}{2} = 840.0 \\ t &= \tfrac{(0.1)^2}{2(1)(0.084)} = 0.0595 \end{aligned} \tag{3.9}$$

To reiterate, t is the typical mean passage time expected from Fick's Second Law and the Stokes-Einstein relation. Our claim is that this should equate to the mean passage time obtained from actual simulation.

To test this claim via simulation, we create a linear compartment of 21 voxels, with absorbing boundaries a placed a full 10 voxels distant on either side of the origin voxel `o` such that any travelling agent `T` must travel $0.1\mu m$ to arrive at a boundary. We seed 10000 `T` at `o` and track their progress as they diffuse randomly through the compartment, only halting once they arrive at an absorbing boundary. This equates to 10000 independent identically distributed runs of `T`.

We compare this compartment structure in simulation against alternatives that approximate this linear compartment. These have a reduced size of only 11 voxels, with the voxel on one end of the compartment an absorbing boundary and `o` at the other end (the origin). The approximation lies in the fact that the origin voxel implicitly forms a reflecting boundary: whereas in the baseline compartment there are two directions for the diffusing agent `T` to travel from the origin, this is condensed to one in the approximation with a corresponding loss of potential diffusion. Hence this reflecting boundary can have either an unmodified translocation rate, or

Structure	Mean	Median	Variance
a10o10a	0.0594	0.0448	0.0023
ro10a	0.0650	0.0487	0.0028
rro10a	0.0592	0.0451	0.0023

Table 1: Mean and median first passage times, along with variance, of 10000 T with $\gamma = 840.0$, $R_{vox} = 0.005\mu m$, and compartment structure as shown. *rro10a* is a much better approximation of *a10o10a* than *ro10a*, and the mean matches the expected time given in Equation 3.9.

one that compensates for the difference in structure between the full linear compartment and its approximation (by doubling the rate of reflection at the reflecting boundary).

We introduce notation to describe this one-dimensional compartment structure: *r* indicates a reflecting boundary, *rr* a double reflecting boundary with compensated diffusion rate, *o* the origin voxel, and *a* an absorbing boundary from which the travelling agent can no longer escape. We also display in the notation the number of voxels that T must travel from the origin to the boundary. The compartments described above may thus be represented, respectively, as *a10o10a*, *ro10a*, and *rro10a*.

Table 1 shows that doubling the rate of reflection at the reflecting boundary compensates more efficiently for the difference in compartment structure, as we might expect. The modified compartment *rro10a* approximates *a10o10a* extremely well, and both are accurate to within a minimal margin of error compared to the theoretical typical time derived in Equation 3.9.

We visualise the distribution of arrival times in Figure 23. The long tail of the distribution, caused by stragglers meandering through the compartment and absorbed late at the boundary, accounts for why the mean arrival time is substantially greater than the median.

Further experiments (Table 2) extend this analysis for both larger symmetrical and non-symmetrical structures. In all cases, compartments are constructed with uniform voxels each with a radius of $R_{vox} = 0.005\mu m$.

- For a20o20a the expected time is: $t = \frac{(0.2)^2}{2(1)(0.084)} = 0.2381$
- For a40o40a the expected time is: $t = \frac{(0.4)^2}{2(1)(0.084)} = 0.9524$
- For a31o10a the expected time is: $t = \frac{(0.31)(0.1)}{2(1)(0.084)} = 0.1845$
- For a41o20a the expected time is: $t = \frac{(0.41)(0.2)}{2(1)(0.084)} = 0.4881$
- For a61o10a the expected time is: $t = \frac{(0.61)(0.2)}{2(1)(0.084)} = 0.7262$

We see that observations from simulation match the expected times provided by theoretical estimation as calculated from the Stokes-Einstein relation, and that the mean is a good approximation of the system for both symmetrical and non-symmetrical structures.

Structure	Mean	Median	Variance
a20o20a	0.2379	0.1776	0.0379
ro20a	0.2507	0.1891	0.0426
rro20a	0.2381	0.1805	0.0373
a40o40a	0.9566	0.7739	0.5989
ro40a	0.9683	0.7410	0.6288
rro40a	0.9634	0.7267	0.6183
a31o10a	0.1872	0.1188	0.0396
r10o10a	0.1791	0.1120	0.0368
rr10o10a	0.1793	0.1119	0.0366
a41o20a	0.4883	0.3501	0.1977
r10o20a	0.4832	0.3481	0.1933
rr10o20a	0.4810	0.3430	0.1950
a61o20a	0.7180	0.4497	0.5905
r20o20a	0.7258	0.4619	0.5904
rr20o20a	0.7075	0.4455	0.5798

Table 2: Mean and median first passage times, along with variance, of 10000 $\mathtt{T}$ with $\gamma = 840.0$, $R_{vox} = 0.005\mu m$, and compartment structure as shown. In all cases, the doubly reflecting boundary is a better approximation than the singly reflecting boundary, and the mean time matches the expected.

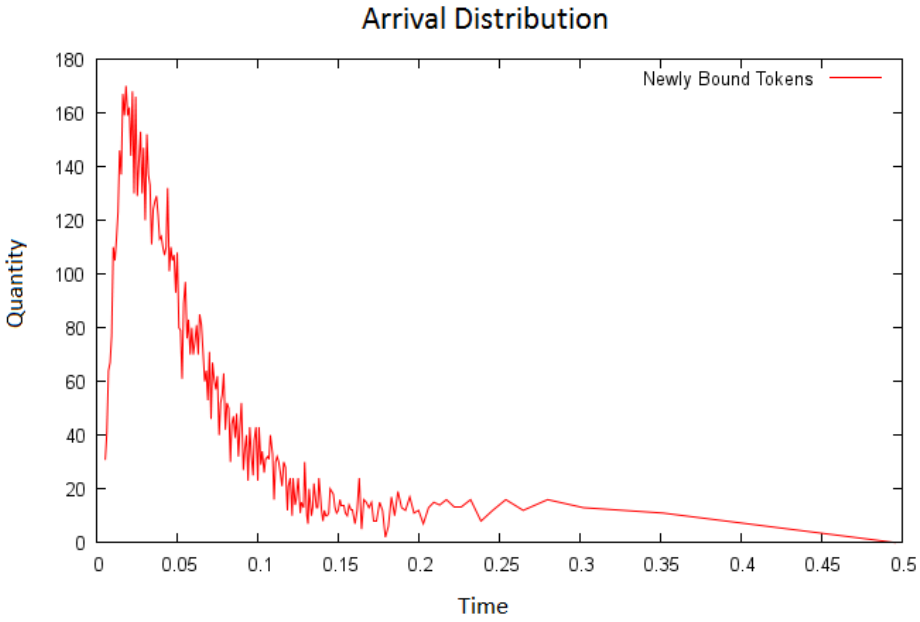


Figure 23: A plot of the arrival distribution of travelling agents `T` at the absorbing boundaries of an *a10o10a* compartment with translocation rate $\gamma = 840.0$. We observe a high-peaking distribution with a long trailing tail.

3.1.2 Spatial Kappa, Kappa, and Kappa Tokens

Our next test of Spatial Kappa compares it against standard Kappa simulations for simple one-dimensional diffusion. We verify the consistency of Spatial Kappa with regards to standard Kappa as a baseline from which to apply some of the results derived above.

We revisit the linear compartment model of n voxels inhabited by a single class of travelling agent `T`. We seed 10000 `T` at the origin, and record the arrival times of `T` at the absorbing boundaries. In Spatial Kappa syntax, the compartment and voxel channels can each be defined in a single line of code regardless of the size of n. Translocation rules are a single line of code based on these channels, and the simulator then implicitly takes care of the locations of the individual `T`.

In standard Kappa, however, the location voxels must be represented as states of an arbitrary '`location`' site maintained by the travelling agent `T`. Furthermore, a total of $2(n-1)-2$ rules must be manually written to describe the translocation (diffusion) of `T`: $2(n-1)$ rules for two-way diffusion between a string of n voxels, minus a further 2 since the end voxels of the linear compartment are absorbing and thus cannot be escaped from.

Alternatively, we may keep track of `T` in Kappa by counting tokens per voxel rather than explicit agents with state. This does not cut back on the number of rules that needs to be written in comparison with standard Kappa. On the other hand it should improve significantly on

Compartment Structure	Pure Kappa	Token Kappa	Spatial Kappa
a10o10a	0.0598	0.0593	0.0594
a20o20a	0.2395	0.2420	0.2379
a40o40a	0.9526	0.9501	0.9566
a31o10a	0.1839	0.1860	0.1872

Table 3: Mean first passage times of 10000 T with $\gamma = 840.0$, $R_{vox} = 0.005\mu m$, and compartment structure and simulation environment as shown. The three versions of the Kappa model differ only within reasonably expected parameters.

γ	Pure Kappa	Token Kappa	Spatial Kappa
210.0	0.2391	0.2360	0.2376
420.0	0.1186	0.1194	0.1195
840.0	0.0598	0.0593	0.0594
1680.0	0.0297	0.0300	0.0297
3360.0	0.0148	0.0149	0.0150

Table 4: Mean first passage times of 10000 T with compartment structure *a10o10a*, $R_{vox} = 0.005\mu m$, and γ and simulation environment as shown. Again, the three versions of the Kappa model differ only within reasonably expected parameters.

simulation time without sacrificing expressiveness, since in the model as described we simply adjust token counts in a linear fashion, analogous to a system of ordinary differential equations.

We begin with $\gamma = 840.0$ as a base figure for the stochastic translocation rate across all three types of Kappa and different compartments. We then proceed to compare how the simulations behave over varying rates.

The results (Table 3 and Table 4) obtained show that there is very little difference in the results observed between Spatial Kappa and both types of standard Kappa in modelling one-dimensional diffusion. Furthermore in terms of computation speed we observe similar performance between standard Kappa and Spatial Kappa when run in the current version of the Spatial Kappa tool. Standard Kappa simulations run much faster in the latest KaSim tool compared to Spatial Kappa simulations in the Spatial Kappa tool, however, and token-based Kappa simulations of the above compartment run even faster still. The primary reason for the disparity between the KaSim and Spatial Kappa tools is the lack of optimisation for the Spatial Kappa simulation algorithm and its Java-based implementation with dynamic tracking of observable trajectories. This disadvantage for Spatial Kappa is offset by the ease with which we can define both spatial structures and rules upon said spatial structures.

Compartment Structure	γ	Mean
a10o10a	52.5	0.9447
a20o20a	210.0	0.9459
a40o40a	840.0	0.9566
a31o10a	210.0	0.7304
a61o20a	840.0	0.7180

Table 5: Mean first passage times of 10000 T with compartment structure (total length = $0.4\mu m$) and γ as shown. Scaling γ across the compartment structure results in a standardised mean arrival time.

3.1.3 Scaling

If we assume that the compartment length x is equivalent across all simulations regardless of the number of voxels n, by adapting the equation for deriving the stochastic translocation rate γ we obtain the following equations.

$$\begin{aligned} \gamma_{10} &= \frac{2D}{(2R_{vox10})^2} \\ \gamma_{20} &= \frac{2D}{(\frac{2R_{vox10}}{2})^2} \\ \gamma_{40} &= \frac{2D}{(\frac{2R_{vox10}}{4})^2} \end{aligned} \tag{3.10}$$

These resolve to:

$$\begin{aligned} \gamma_{20} &= 4\gamma_{10} \\ \gamma_{40} &= 16\gamma_{10} \end{aligned} \tag{3.11}$$

The above indicates that should we wish to achieve the same mean first passage time across travel length $n = 10, 20, 40$, we should use stochastic rates $\gamma_{40} = 4\gamma_{20} = 16\gamma_{10}$ respectively. This is borne out by actual simulations as shown in Table 5. What is not shown in the results is the additional computational cost incurred in fine-graining the voxel resolution; simulations of a larger number of voxels incur an exponential penalty to how long they take to complete, due to the increase in individual voxels whose population the simulation must track.

What happens if, instead of observing mean first passage time (i.e. $k_{absorption} = \infty$), we instead set the boundary to only absorb travelling agents at a set rate (i.e. $k_{absorption} = 0.1$) and observe how the absorption profile changes as the compartment scales? We might expect the mean absorption time to increase with the translocation rate (and thus the size of the compartment), since by increasing the translocation rate the travelling agent T would diffuse away from the boundary before it can be absorbed.

We do not observe in Table 6 the effect we might have expected: the mean absorption time

Compartment Structure	γ	Mean
a10o10a	52.5	0.9808
a20o20a	210.0	0.9608
a40o40a	840.0	0.9953
a80o80a	3360.0	0.9632
a500o500a	131250.0	1.5329

Table 6: Mean first passage times of 10000 T with compartment structure (total length = $0.4\mu m$) and γ as shown, at absorption rate $k_{absorption} = 0.1$. Again, we see that scaling γ across the compartment structure results in a standardised mean arrival time.

to increase with the translocation rate. This is because, in the simulation time that $k_{absorption}$ might occur ($\tau = k^{-1}_{absorption} = 10$), the Stokes-Einstein relation tells us that average displacement over that time would be $x = \sqrt{2D\tau} = 1.3\mu m = 3240$ voxels in this structure. Given that this allows T plenty of opportunity for another attempt at absorption, in effect the overall absorption probability (and thus the mean absorption time) balances out.

A side effect of this analysis is the unfortunate realisation that it requires the equivalent of many weeks of computational time to run simulations with a thousand voxels, even of simple reaction-diffusion models. Compounding this problem is the fact that current implementations of Spatial Kappa are not implemented with parallelisation or other methodologies for improving efficiency.

3.1.4 The 'Jumping' Effect

In addition to the results in Table 6, we manually observe an additional effect: at $t = \frac{1}{4}, \frac{1}{2}, 1, 2, 4, 8, ...$, the simulation observable appears to 'jump' (Figure 24) in time. Concretely, we observe a large increase in the population of T bound to the absorbing boundary immediately following the specified timepoints. These jumps are not observed upon a fully absorbing boundary where $k_{absorption}$ is infinite.

We verify that this problem occurs only in Spatial Kappa by comparing it against equivalent simulations in Kappa (with the travelling agents represented as both agents in the rule-based modelling sense as well as tokens), and also against an implementation of the Gillespie algorithm (customised to this particular problem) written in Scala aided by the Module Integration Simulator (*mois*) modelling framework. By 'equivalent', we mean that we employ the same compartment structure, agent numbers, and diffusion and binding rates across all four models. We observe the 'jumping' effect in Spatial Kappa simulations only, in such a way as to inflate the cumulative arrival count of the agents compared to the three baseline simulations. We can see this effect from a different angle when we graph the arrival distribution of travelling agents

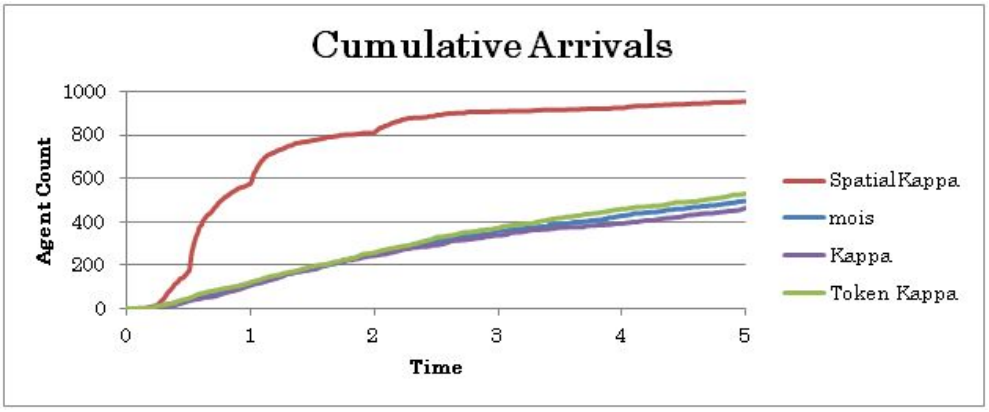


Figure 24: A plot of bound T on a linear *a500o500a* compartment with translocation rate $\gamma = 131250.0$ and absorption rate $k_{absorption} = 0.1$, modelled in Spatial Kappa, standard Kappa, standard Kappa with tokens, and an implementation of the Gillespie algorithm in Scala (via the Module Integration Simulator *mois* [91]). We observe noticeably aberrant 'jumps' in the time series for the Spatial Kappa simulation, which we originally possibly related to the 'ripple effect' of T being rejected from one partially-absorbing boundary and diffusing towards the other.

in Figure 25: whereas in the baseline simulation these are more or less constant over time, in Spatial Kappa they are notably concentrated around the 0.5 and 1.0 time points.

We initially analyse this phenomenon with the assistance of Anatoly Sorokin, current developer of the Spatial Kappa tool. Let us consider the situation on the terminal voxel from the point of view of the Gillespie algorithm: once an agent arrives, there are two possible reactions, binding and translocation. The probability of this binding is defined by the rate constants, such that $p_{bind} = \frac{k_{absorption}}{\gamma + k_{absorption}}$. Thus with $k_{absorption} = 0.1$, the model boundary is effectively reflecting with $\gamma = 131250.0$.

Why then do we observe such pronounced jumps with such regularity? Initially we observe that the average time required for T to travel across the entire compartment is $\tau_{full} = \frac{0.4^2}{2D} = 0.95$. Our original hypothesis was that the 'jumps' result from the 'ripple effect' of Ts reflecting off the boundaries, arriving at the single point of absorption in large waves that become visible in the time series. However, we do not believe this explains why the intervals for the 'jumps' double rather than maintain a consistent frequency, nor why they also occur at $t = \frac{1}{4}$ and $t = \frac{1}{2}$. Further models where the rates are kept equivalent but the compartment is narrowed to a 300 voxel radius display the same 'jumping' behaviour at $t = \frac{1}{4}, \frac{1}{2}, 1, 2, 4, 8, ...$, despite $\tau_{full} = \frac{0.24^2}{2D} = 0.34$. We also observe the jumping effect in a supposedly equivalent structure (*ro500a*) that we might expect to have a lesser 'ripple' effect (Figure 26). Hence we believe this hypothesis to be mistaken, but we have yet to develop one to take its place.

Varying the absorption rate has little effect (Figure 27 top); similarly, upgrading to 10000

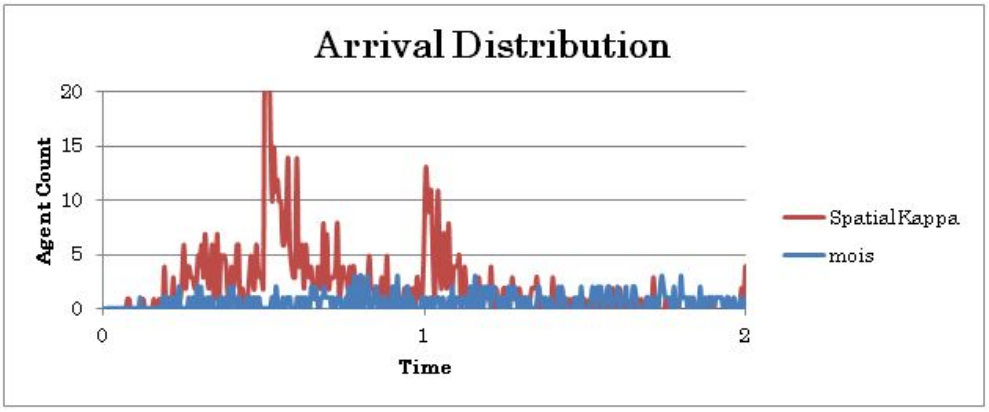


Figure 25: A closer look at the arrival distribution of travelling agents `T` at the boundaries of an *a500o500a* linear compartment structure in Spatial Kappa and *mois* respectively. Whereas in *mois* the arrivals are more or less constant over time, this is not the case for the 'jumping' effect in Spatial Kappa.

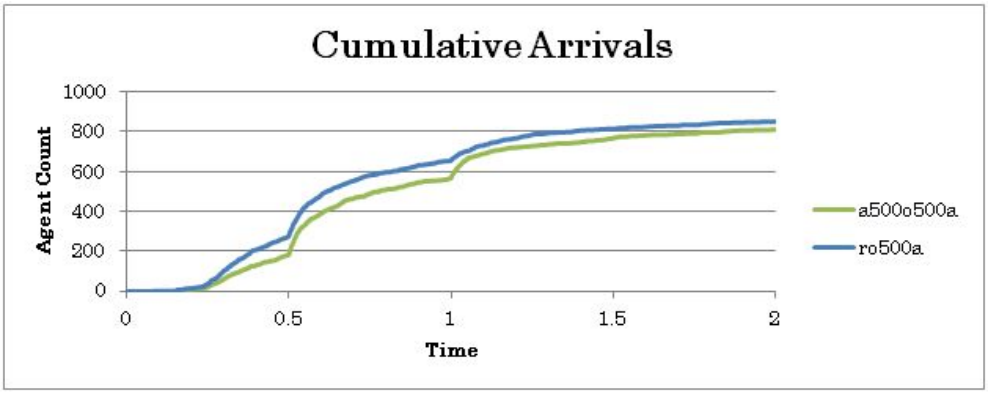


Figure 26: Cumulative arrivals (bound travelling agent `T`) on linear *a500o500a* and *ro500a* compartments, displaying a similar 'jumping' effect on both.

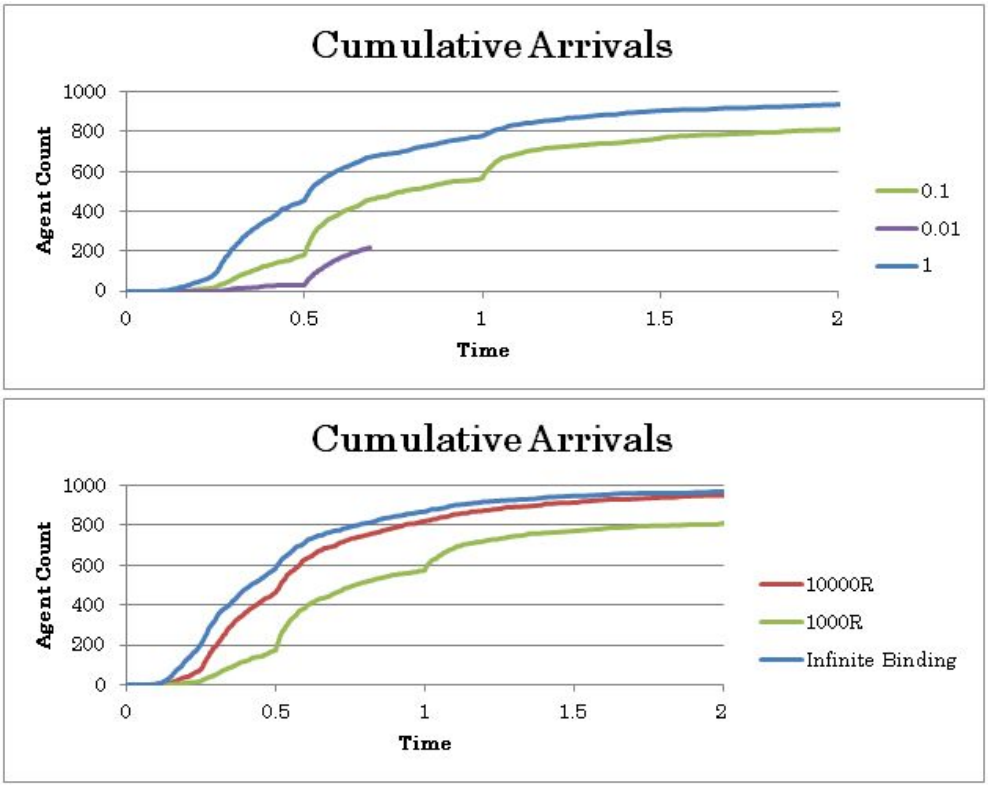


Figure 27: Cumulative arrivals (bound travelling agent `T`) on a linear *a500o500a* compartment with translocation rate $\gamma = 131250.0$ modelled in Spatial Kappa, varying the binding rate (1000 receptors with absorption rate as specified, top) or number of receptors (1000 and 10000 with absorption rate $k_{absorption} = 0.1$) and infinite binding (bottom). The greater the binding rate the less prominent the 'jumping' effect, but it never disappears.

such travelling agents does not remove the 'jumping' effect, merely makes the simulation take a week to run the first 0.5 time units (not shown). Upgrading to 10000 receptors (keeping the number of travelling agents the same) has the effect of increasing the effective binding rate at the absorbing boundary, but also does not remove the 'jumping' effect. Upgrading to an infinite binding rate has a similar result (Figure 27 bottom).

Further models where the rates are kept equivalent but the compartment is narrowed to a 300 voxel radius or lengthened to a 1000 voxel radius display the same 'jumping' behaviour at the same intervals $t = \frac{1}{4}, \frac{1}{2}, 1, 2, 4, 8, \ldots$ despite the corresponding difference in expected mean arrival time.

We have thus yet to develop a hypothesis to account for the 'jumping' effect, except to posit that this is not a natural phenomenon of linear compartment structure and diffusion rates, but rather an issue with the internal logic of Spatial Kappa and its lattice-based view of space that must be addressed in later releases.

3.2 In Two Dimensions

In d dimensions the central limit theorem generalises as follows:

$$\sum_{1}^{n} X_i \sim n \cdot E(X) + \sqrt{n} \cdot N(0, \Sigma(X)) + o(\sqrt{n}) \tag{3.12}$$

where $E(X)$ is the mean vector and $\Sigma(X)$ the $d \times d$ covariance matrix of X (X^i being the ith component of X): $\Sigma_{ij} := E(X^i X^j) - E(X^i)E(X^j)$

Assuming steps in all dimensions have zero mean, and adopting a similar approach to that described for the one-dimensional case in Equation 3.7 based on Equation 3.12:

$$\begin{aligned} X(t) &= \sum \varepsilon_i \, 0 \cdot E(\varepsilon) + \sqrt{\gamma^* t} \cdot N(0, \Sigma(\varepsilon)) \\ &= N(0, \gamma^* t \cdot \Sigma(\varepsilon)) \end{aligned} \tag{3.13}$$

Again, equating the variation for the distribution of $X(t)$ with the variation associated to the multidimensional distribution for Brownian motion gives:

$$\gamma^* \Sigma(\varepsilon) = 2DI \tag{3.14}$$

These equations allow us to once more determine the stochastic translocation rate from the diffusion coefficient based on the connectivity of the spatial structure. We show the calculations for this procedure in the next section.

For compartments in two and higher dimensions with restricted absorbing boundaries, the problem of calculating the mean first passage time reduces to a well-known variation of a standard narrow escape problem, in which a Brownian particle confined to a bounded domain by a reflecting boundary seeks to make its escape through a small absorbing window. We turn

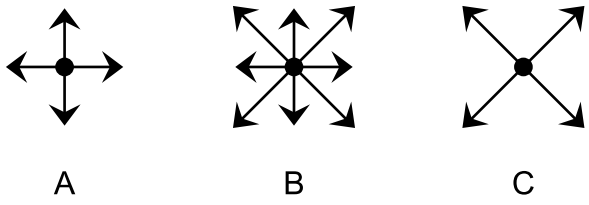


Figure 28: A variety of two-dimensional connectivity schemes. (A) 4-way square grid (predefined in Spatial Kappa as EdgeNeighbour channels). (B) 8-way compass grid (predefined in Spatial Kappa as Neighbour channels). (C) 4-way diagonal grid.

to published equations for calculating the mean escape time, noting that they have as-yet unresolved error terms, and thus should be regarded as ballpark lower bounds rather than exact targets.

3.2.1 Connectivity in Two Dimensions

Deriving the stochastic translocation rate of a model depends heavily on the *connectivity* displayed within the compartmental structure. Figure 28 shows the examples of connectivity we explore further for two-dimensional diffusion in Spatial Kappa. Using Equation 3.13 and Equation 3.14, we calculate the covariance matrices and hence the stochastic translocation rates of (1) 4-way square lattice (Figure 28A, Equation 3.15), (2) 8-way compass lattice (Figure 28B, Equation 3.16), and (3) 4-way diagonal lattice (Figure 28C, Equation 3.17) connectivity schemes.

$$\begin{gathered}(X^1,X^2) = \tfrac{1}{4}((-h,0)+(h,0)+(0,-h)+(0,h)) \\ \Sigma(\varepsilon) = \begin{pmatrix} (\frac{2h^2}{4}) - (0)(0) & 0-(0)(0) \\ 0-(0)(0) & (\frac{2h^2}{4}) - (0)(0) \end{pmatrix} = \begin{pmatrix} \frac{h^2}{2} & 0 \\ 0 & \frac{h^2}{2} \end{pmatrix} \\ \gamma^* \cdot \Sigma(\varepsilon) = 4\gamma \cdot \Sigma(\varepsilon) = 2h^2\gamma I = 2DI \\ \gamma = \tfrac{D}{h^2}\end{gathered} \tag{3.15}$$

$$\begin{gathered}(X^1,X^2) = \tfrac{1}{8}((-h,-h)+(-h,0)+(-h,h)+(0,-h)+(0,0)+(0,h)+(h,-h)+(h,0)+(h,h)) \\ \Sigma(\varepsilon) = \begin{pmatrix} (\frac{6h^2}{8}) - (0)(0) & 0-(0)(0) \\ 0-(0)(0) & (\frac{6h^2}{8}) - (0)(0) \end{pmatrix} = \begin{pmatrix} \frac{3h^2}{4} & 0 \\ 0 & \frac{3h^2}{4} \end{pmatrix} \\ \gamma^* \cdot \Sigma(\varepsilon) = 8\gamma \cdot \Sigma(\varepsilon) = 6h^2\gamma I = 2DI \\ \gamma = \tfrac{D}{3h^2}\end{gathered} \tag{3.16}$$

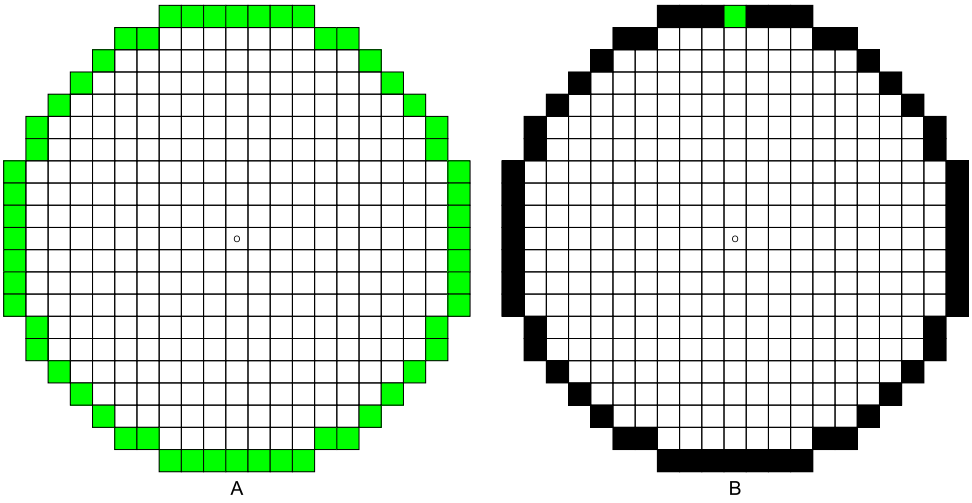


Figure 29: Defining two-dimensional disks in Spatial Kappa: a disk with radius 10 voxels. The centre of the disk, marked ○, is the origin voxel for the travelling agent T. (A) A fully absorbing boundary. (B) A reflective boundary with a single absorbing voxel directly 'north' of the origin.

$$
\begin{aligned}
(X^1, X^2) &= \tfrac{1}{4}((-h,-h)+(-h,h)+(h,-h)+(h,h)) \\
\Sigma(\varepsilon) &= \begin{pmatrix} (\frac{4h^2}{4}) - (0)(0) & 0-(0)(0) \\ 0-(0)(0) & (\frac{4h^2}{4}) - (0)(0) \end{pmatrix} = \begin{pmatrix} h^2 & 0 \\ 0 & h^2 \end{pmatrix} \\
\gamma^* \cdot \Sigma(\varepsilon) &= 4\gamma \cdot \Sigma(\varepsilon) = 4h^2\gamma I = 2DI \\
\gamma &= \tfrac{D}{2h^2}
\end{aligned}
\tag{3.17}
$$

3.2.2 Two Dimensions: Disks

For our first simulation in two dimensions, let us assume a regular disk of radius $n = 10$ voxels and $0.1\mu m$, such that $R_{vox} = 0.005\mu m$. We observe the translocation of a travelling agent T with diffusion coefficient $D = 0.084\frac{\mu m^2}{s}$.

As shown in Figure 29, the travelling agents are seeded at the exact centre of the disk. We consider two types of boundaries. The first is a fully absorbing boundary, where every voxel on the outer edge of the disk absorbs any travelling agents that enter. The second is a boundary with a single absorbing voxel located directly 'north' of the origin.

The expected time for an agent T to travel in two dimensions from its origin to the absorbing border (hereon noted as its full border absorption time (t_{FBA})), according to the Stokes-Einstein relation in Equation 3.3, is given in Equation 3.18.

$$
\begin{aligned}
t_{FBA} &= \tfrac{R^2}{2dD} \\
&= \tfrac{(0.1)^2}{2(2)(0.084)} \\
&= 0.0298
\end{aligned}
\tag{3.18}
$$

Connectivity	γ	t_{FBA}	t_{SPA}
4-way Square (1)	840.0	0.0302	0.6931
8-way Compass (2)	280.0	0.0317	0.6418
4-way Diagonal (3)	420.0	0.0321	0.4591

Table 7: Mean first passage times of 10000 T on a disk with radius $n = 10$ voxels and $R_{vox} = 0.005\mu m$; connectivity, absorbing borders, and γ as shown. We see that the t_{FBA} match the expected result from Equation 3.18, but that the t_{SPA} largely deviate.

To calculate the expected time for an agent T to travel in two dimensions from its origin to a single absorbing voxel (hereon noted as its single point absorption time (t_{SPA})), we equate the problem to that of a narrow escape. If the pi-adjusted ratio between the absorbing boundary and the circumference ($\varepsilon = \frac{\pi a}{2\pi R}$) is low, we can use the diameter of a single voxel ($a = 2R_{vox} = 0.01\mu m$) to calculate the mean first passage time (MFPT) of a narrow escape problem. In this case, the target t_{SPA} is equivalent to the MFPT, which according to equations presented by Singer, Schuss, and Holcman [75, 76, 78] and verified by Kolokolnikov et al. [69] may be approximated by Equation 3.19.

$$\begin{aligned} t_{SPA} &= \frac{R^2}{D}[ln\frac{R}{a} + 2ln2 + \frac{1}{4} + O(\varepsilon)] \\ &= \frac{(0.1)^2}{(0.084)}[ln\frac{(0.1)}{(0.01)} + 2ln2 + \frac{1}{4} + O(\varepsilon)] \\ &\sim 0.4689 \end{aligned} \tag{3.19}$$

In calculating the t_{SPA} we encounter the impact of the Cartesian approximation of circular and spherical structures in Spatial Kappa. In particular, the Spatial Kappa approximation of the circumference of the disk, as visualised in Figure 29, leaves us prone to error in comparison to MFPT equations that assume a pi-derived relation between the length of the aperture on the circumference on the disk (a) and the disk's radius (R).

We also introduce a further measure of the usefulness of the mean as an approximation of the time it takes for a typical travelling agent to arrive at the absorbing boundary: the relative error ($\frac{standarddeviation}{mean}$). A relative error greater than 1 may be thought to indicate unreliability of the mean, whereas a relative error substantially less than 1 shows that the mean is a good approximation of the median. In all tables to follow, we colour the t_{FBA} and t_{SPA} according to the relative error. Blue indicates relative errors of less than 0.95, red indicates relative errors of greater than 1.05, and orange everything in between.

The results for the t_{FBA} in Table 7 are a good match to the target times obtained from Equation 3.18. The slight increase in mean t_{FBA} for 8-way Compass and 4-way Diagonal connectivity may be attributed to the greater step distance involved in travelling diagonally upon a lattice grid, which subtly increases the margin of error in calculating the MFPT.

Our results for the t_{SPA}, however, are far less reliable. The t_{SPA} of 4-way Square and 8-way Compass connectivities is considerably greater than the t_{SPA} of the 4-way Diagonal connectivity; we posit that this is due to the fact that the latter effectively halves the diffusive area of the compartment (much like a bishop on a chessboard as opposed to a rook (Square connectivity) or a queen (Compass connectivity)). Overall, Equation 3.19 considerably underestimates the observed MFPT to a narrow aperture – perhaps the big $O(\varepsilon)$ component of the equation bears considerable weight that is not reflected in our estimations of the result. Alternatively the Spatial Kappa approximation of disks (a grid-like layout of voxels with channels and channel connectivities between them) is not analytically conducive to narrow escape equations including Equation 3.19.

Building on these results and in an attempt to verify them further, we compare full border absorption and single point absorption over different disk diameters.

For disks with 20 voxel radius:

$$\begin{aligned} t_{FBA} &= \frac{(0.2)^2}{2(2)(0.084)} \\ &= 0.1190 \end{aligned} \tag{3.20}$$

$$\begin{aligned} t_{SPA} &= \frac{(0.2)^2}{(0.084)}[ln\frac{(0.2)}{(0.01)} + 2ln2 + \frac{1}{4} + O(\varepsilon)] \\ &\sim 2.2057 \end{aligned} \tag{3.21}$$

For disks with 40 voxel radius:

$$\begin{aligned} t_{FBA} &= \frac{(0.4)^2}{2(2)(0.084)} \\ &= 0.4762 \end{aligned} \tag{3.22}$$

$$\begin{aligned} t_{SPA} &= \frac{(0.4)^2}{(0.084)}[ln\frac{(0.4)}{(0.01)} + 2ln2 + \frac{1}{4} + O(\varepsilon)] \\ &\sim 10.1432 \end{aligned} \tag{3.23}$$

In all cases for radii 10/20/40 voxels, Table 8 shows that the t_{FBA} match the Stokes-Einstein relation well. On the other hand, the results for t_{SPA} again deviate considerably from the target narrow escape result. Given that the MFPT formulae are dependent on area but 4-way Diagonal connectivity utilises only half of a disk's surface, we might expect some deviation in the t_{SPA} for said connectivity in comparison to its counterparts (4-way Neighbour and 8-way Compass). However, this is not enough to explain the discrepancy to the formulae. The larger the structure (and the smaller the corresponding aperture in comparison to the perimeter and surface area of the disk), the greater the deviation we observe.

Furthermore, we once again glimpse the 'jumping' problem exhibited in one-dimensional scaling with probabilistic absorption – this is particularly noticeable in the two t_{SPA} with high relative error in Table 8 (Figure 30). This phenomenon occurs because a single point absorbing boundary shares the property with our previous 'jumping' example that even if `T` reaches the

Radius	Connectivity (γ)	t_{FBA}	t_{SPA}
20 voxels (0.2 μm)	4-way Square ($\gamma = 840.0$)	0.1187	2.7311
20 voxels (0.2 μm)	8-way Compass ($\gamma = 280.0$)	0.1198	2.7902
20 voxels (0.2 μm)	4-way Diagonal ($\gamma = 420.0$)	0.1204	2.1715
40 voxels (0.4 μm)	4-way Square ($\gamma = 840.0$)	0.4712	6.5025
40 voxels (0.4 μm)	8-way Compass ($\gamma = 280.0$)	0.4773	7.7221
40 voxels (0.4 μm)	4-way Diagonal ($\gamma = 420.0$)	0.4876	7.5081

Table 8: Mean first passage times of 10000 T on a disk with $R_{vox} = 0.005\mu m$; connectivity, absorbing borders, disk radius, and γ as shown. We see that the t_{FBA} match the expected result from Equation 3.20 and Equation 3.22, but that the t_{SPA} largely deviate.

boundary it is only absorbed with a certain probability. In this two-dimensional single point absorption case, the probability in question is the probability that the boundary voxel it has arrived at is the absorbing one. The effect is less pronounced in two dimensions, possibly due to the greater field of diffusion, but occurs clearly when the probability of arriving at the absorbing voxel is sufficiently low in comparison to the possibility of reflecting from the boundary (as in the single point absorption on a disk with radius 40 voxels and connectivity that utilises the entire surface area of the disk).

The next question we wish to ask concerns scaling, as investigated for the one-dimensional case. Does varying the number of voxels and translocation rate for a disk with a given radius produce similar results, as we have successfully shown previously for linear compartments? Here, we must not forget also to scale the size of the aperture: a single voxel absorption point in a disk of radius 10 voxels is equivalent to two voxels in a disk of radius 20 voxels and four in a disk of radius 40 voxels, assuming that we scale accurately. We may also check here if this negates the 'jumping' problem experienced above.

Table 9 indicates that the t_{FBA} results scale correctly. However, we once again face the caveat that the t_{SPA} results are not comparable to the expected target. The 40 voxel radius 4-way Square and 8-way Compass t_{SPA} results in particular are anomalous in comparison to their 10 and 20 voxel radius equivalents. In particular, we encounter a noticeable 'jumping' phenomenon during these simulations which we do not encounter when simulating t_{SPA} on disks with radius 10 and 20 voxels.

In Figure 31 we plot the 'jumping' effect of radius 40 voxel Neighbour connectivity against its scaled counterparts with radius 10 and 20 voxels respectively (data from Table 9). We see that the jumping effect causes the simulation to overshoot the 'correct' time series, thus decreasing the observed MFPT. This leads us to believe that the 'jumping' effect is responsible for skewing the results, but we are as yet unsure as to *why* it occurs.

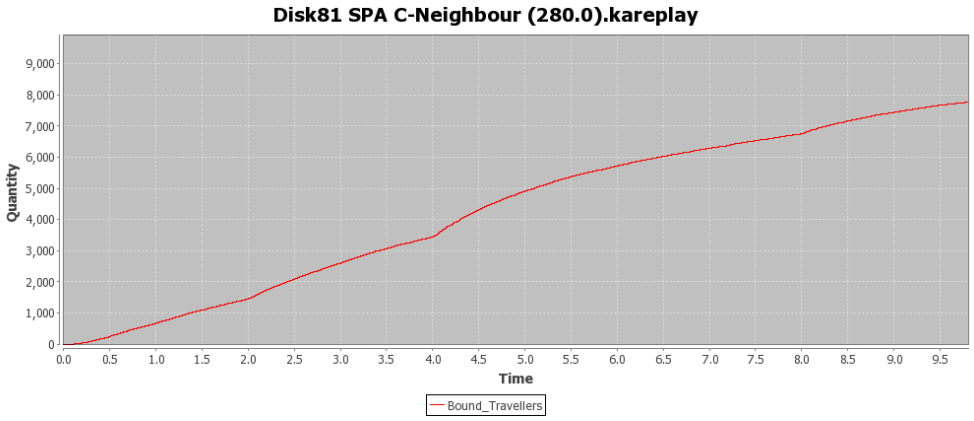


Figure 30: A plot of bound $\mathtt{T}$ on a disk with radius 40 voxels, $R_{vox} = 0.005\mu m$, 8-way Compass (Neighbour) connectivity, single point absorption, and $\gamma = 280.0$. Once again we observe the 'jumping' effect, albeit not as pronounced in the partially absorbing linear compartment of Figure 24.

Radius	Connectivity (γ)	t_{FBA}	t_{SPA}
10 voxels (0.4 μm)	4-way Square ($\gamma = 52.5$)	0.4842	11.1781
10 voxels (0.4 μm)	8-way Compass ($\gamma = 17.5$)	0.5030	10.3398
10 voxels (0.4 μm)	4-way Diagonal ($\gamma = 26.25$)	0.5089	7.2055
20 voxels (0.4 μm)	4-way Square ($\gamma = 210.0$)	0.4719	10.8502
20 voxels (0.4 μm)	8-way Compass ($\gamma = 70.0$)	0.4820	11.1830
20 voxels (0.4 μm)	4-way Diagonal ($\gamma = 105.0$)	0.4767	8.6033
40 voxels (0.4 μm)	4-way Square ($\gamma = 840.0$)	0.4712	4.4283
40 voxels (0.4 μm)	8-way Compass ($\gamma = 280.0$)	0.4773	4.2292
40 voxels (0.4 μm)	4-way Diagonal ($\gamma = 420.0$)	0.4876	7.4635

Table 9: Mean first passage times of 10000 $\mathtt{T}$ on a disk with $R_{vox} = 0.005\mu m$; connectivity, absorbing borders, disk radius, and γ as shown. We see that the t_{FBA} scale well, and that by varying the aperture size so do the t_{SPA}, bar the continued anomalous results for 40 voxel radius 4-way Square and 8-way Compass connectivity.

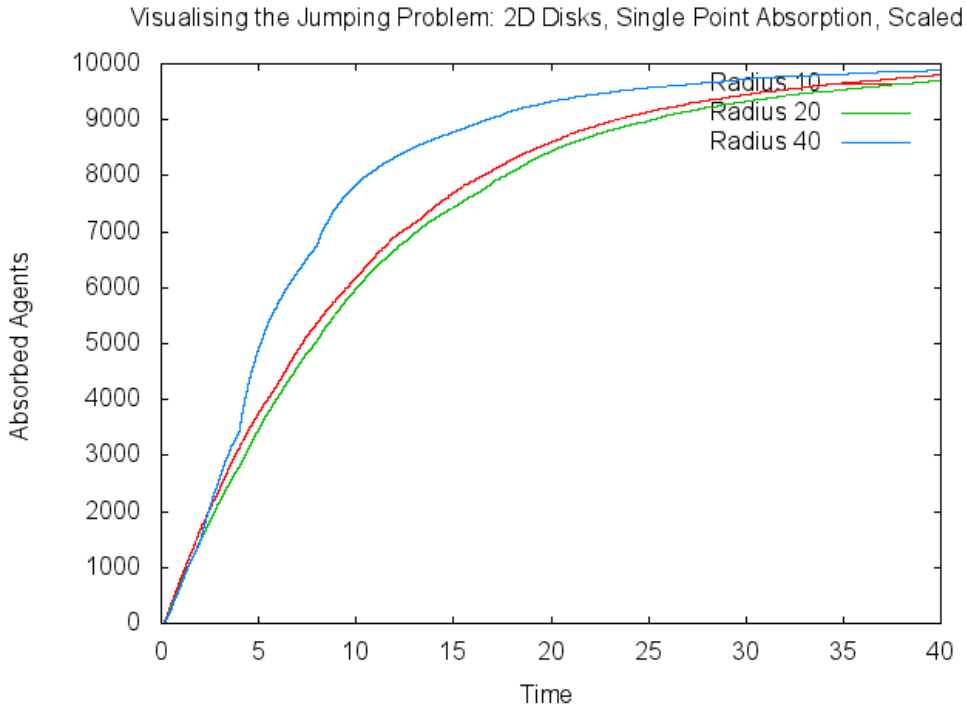


Figure 31: Plotting the 'jumping' effect of Figure 30 against its scaled counterparts with radius 10 and 20 voxels respectively (as noted in Table 9). We see that the jumping effect causes the simulation to overshoot the 'correct' time series, thus lessening the MFPT time. Again, the graph shown is for 8-way Compass connectivity; we observe a similar effect for 4-way Neighbour connectivity, but not for 4-way Diagonal connectivity.

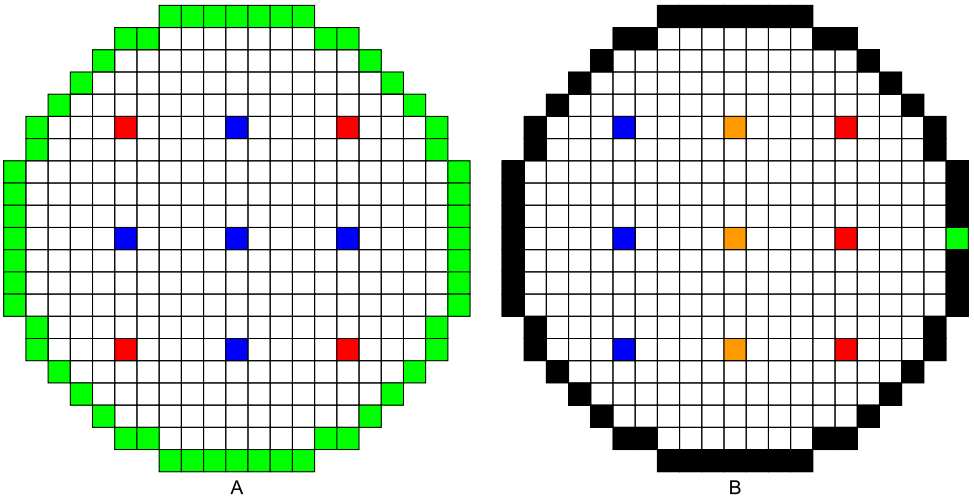


Figure 32: Varying the origin voxel for the travelling agent T upon two-dimensional disks in Spatial Kappa: a disk with radius 10 voxels. (A) A fully absorbing boundary. (B) A reflective boundary with a single absorbing voxel directly 'east' of the origin. We colour the origins with respect to their relative error: blue indicates relative errors of less than 0.95, red indicates relative errors of greater than 1.05, and orange everything in between.

Additionally, we see once more that the computational cost incurred in fine-graining the voxel resolution means that simulations of larger numbers of voxels take far longer to complete, meaning that fine-grained analysis can only be obtained at substantial cost.

Finally, let us investigate what happens if we vary the origin of the travelling agent T. Evidence provided by Mattos et al. [62] suggests that the MFPT is not a good approximation of the behaviour of T if it originates in certain sub-regions for a given absorbing boundary(for example, if the origin is too close to the absorbing boundary). Let us assume once more our standard disk, of radius $n = 10$ voxels with total length $0.1\mu m$ such that $R_{vox} = 0.005\mu m$, T with diffusion coefficient $D = 0.084\frac{\mu m^2}{s}$, and either a fully absorbing boundary or one with a single absorbing voxel located directly 'north' on the circumference.

Varying the origin of T within the disk gives the results shown in Table 10 and Table 11 and Figure 32 for t_{FBA} and t_{SPA} respectively. The results are not as comprehensive as the study in [62], which would necessitate running the above for every voxel in the disk. However, the relative errors match the intuition provided in the paper, that the closer to the absorbing boundary the higher the variance of a single traveller T with respect to the overall mean, and thus the less representative of a typical run the MFPT becomes.

Origin	t_{FBA}	Relative Error
Top Left [4][4]	0.01636	1.1572
Top Centre [4][9]	0.02372	0.9119
Top Right [4][14]	0.01658	1.2104
Centre Left [9][4]	0.02385	0.8923
Centre [9][9]	0.03173	0.7036
Centre Right [9][14]	0.02415	0.8990
Bottom Left [14][4]	0.01692	1.1967
Bottom Centre [14][9]	0.02431	0.8981
Bottom Right [14][14]	0.01671	1.1399

Table 10: Mean first passage times (full border absorption) and relative errors of 10000 T on a disk with radius $n = 10$ voxels, $R_{vox} = 0.005\mu m$, Neighbour connectivity ($\gamma = 840.0$), and origin as shown. We see that the closer to the boundary, the larger the relative error.

Origin	t_{SPA}	Relative Error
Top Left [4][4]	0.6843	0.9406
Top Centre [4][9]	0.6472	0.9783
Top Right [4][14]	0.5872	1.0848
Centre Left [9][4]	0.6968	0.9406
Centre [9][9]	0.6418	0.9916
Centre Right [9][14]	0.5546	1.1357
Bottom Left [14][4]	0.6862	0.9364
Bottom Centre [14][9]	0.6562	0.9706
Bottom Right [14][14]	0.5830	1.0525

Table 11: Mean first passage times (single point absorption) and relative errors of 10000 T on a disk with radius $n = 10$ voxels, $R_{vox} = 0.005\mu m$, Neighbour connectivity ($\gamma = 840.0$), and origin as shown. Again, the closer to the absorbing boundary, the larger the relative error.

3.2.3 Two Dimensions: Planes

Formulae for MFPT on planes with dimension $a \times b$ are not as clear as those for disks; for example, the standard Stokes-Einstein relation doesn't apply for full border absorption. We begin by deriving rough approximations of the t_{FBA} for planes.

It is obvious that the standard Stokes-Einstein relation does provide a lower bound on the MFPT for $x = min(a,b)$ (simply imagine a sphere embedded within the plane). Hence we arrive at Equation 3.24.

$$t_{FBAmin} = \frac{(\frac{min(a,b)}{2})^2}{4D} \tag{3.24}$$

A naive approach to approximating an upper bound would be to apply the same approach to either $x = max(a,b)$, which of course makes no sense when $a = b$ and the sphere fits inside the square anyways. Alternatively, we might choose the distance from the origin to the corner of the plane: $x = \sqrt{(\frac{a}{2})^2 + (\frac{b}{2})^2}$. However, we may do better than this for our discrete voxel formulation of this problem. Let us take N_{vp} to indicate the number of voxels on the perimeter of the plane and $h = 2R_{vox}$ the diameter of a voxel (or the length of a 'hop' along the perimeter). We obtain Equation 3.25 by calculating the time to arrive at any particular boundary voxel according to the Stokes-Einstein relation.

$$t_{FBAmax} = \frac{(6((\frac{a}{2})^2+(\frac{b}{2})^2)+4\sum_{i=1}^{\frac{b}{2h}-1}((\frac{a}{2})^2+(ih)^2)+4\sum_{i=1}^{\frac{a}{2h}-1}((\frac{b}{2})^2+(ih)^2))}{N_{vp} \cdot 4D} \tag{3.25}$$

When $a = b$ and the plane is actually a square, Equation 3.25 condenses to Equation 3.26.

$$t_{FBAmax} = \frac{12((\frac{a}{2})^2)+8\sum_{i=1}^{\frac{a}{2h}-1}((\frac{a}{2})^2+(ih)^2)}{N_{vp} \cdot 4D} \tag{3.26}$$

Strictly speaking, the equations above provide an overestimation rather than an upper bound, relying on the assumption of equi-probable arrival at all points on the boundary. In reality, travelling agents tend to arrive at voxels closer to the origin. In particular, the formula still overestimates the arrival times for uneven planes (Table 12), where it places far too high a bias on voxels a long distance from the origin. It remains a good approximation for when a is roughly equal to b.

Is there a way of integrating the probability of arrival at a voxel into the equation, as we might assume the MFPT formulae for narrow escape problems do? Assuming that $a < b$ we can estimate the MFPT as Equation 3.27. In this equation we multiply the expected time of arrival at each voxel (as calculated by the Stokes-Einstein relation) by the weight of the voxel (minimum voxel distance to origin divided by the actual voxel distance to origin, thus penalising any voxel that is further away from the origin than the minimum). Because of the need to define an origin voxel equidistant from opposing borders, the Spatial Kappa structure is actually defined as $(a+1) \times (b+1)$, with distance measured by the number of 'hops' (translocations)

the travelling agent T is forced to make before encountering a boundary. This assumption is reflected inherently in Equation 3.27, meaning that the equation does not generalise to other planar structures.

$$\begin{aligned} t_{FBA} = \quad & \tfrac{1}{N_{vp}\cdot 4D}(2(\tfrac{a}{2})^2 + 2(\tfrac{(\frac{a}{2})}{(\frac{b}{2})})(\tfrac{b}{2})^2 + 4(\tfrac{(\frac{a}{2})}{(\sqrt{(\frac{a}{2})^2+(\frac{b}{2})^2})})((\tfrac{a}{2})^2+(\tfrac{b}{2})^2) \\ & +4\textstyle\sum_{i=1}^{\frac{b}{2h}-1}(\tfrac{(\frac{a}{2})}{(\sqrt{(\frac{a}{2})^2+(ih)^2})})((\tfrac{a}{2})^2+(ih)^2) + 4\textstyle\sum_{i=1}^{\frac{a}{2h}-1}(\tfrac{(\frac{a}{2})}{(\sqrt{(\frac{b}{2})^2+(ih)^2})})((\tfrac{a}{2})^2+(ih)^2)) \end{aligned} \tag{3.27}$$

Our results (Table 12) approximate this equation well, albeit diverging again for uneven a and b. Interestingly, if we square the weights then we derive t_{FBAmin} as proposed above.

To find the mean first passage times of single point absorption (t_{SPA}), adaptations of Equation 3.28 seem to be most relevant, where $|\Omega|$ is the area of the plane and ε the ratio between the absorbing boundary and the plane perimeter.

$$t_{SPA} = \quad \tfrac{|\Omega|}{\pi D}[ln\tfrac{1}{\varepsilon} + O(1)] \tag{3.28}$$

Equation 3.28 serves as an acceptable lower bound for our purpose. However, since it does not take into account the geometry of the plane (for example, 41x41 vs. 21x81 vs. 11x161 planes), we find it difficult to accept it at face value. We might wish to compare against formulae that assume a disk of the appropriate size centred on the origin and touching the appropriate edge (Equation 3.29). A more precise formula for absorption when the single point is found at the corner of the plane, given by Singer et al. [76], is Equation 3.30 where $\beta = e^{\frac{-\pi b}{a}}$.

$$t_{SPAedge} = \quad \tfrac{R^2}{D}[ln\tfrac{R}{a} + 2ln2 + \tfrac{1}{4} + O(\varepsilon)] \tag{3.29}$$

$$t_{SPAcorner} = \quad \tfrac{2|\Omega|}{\pi D}[ln\tfrac{a}{\varepsilon} + ln\tfrac{2}{\pi} + \tfrac{\pi}{6}\tfrac{b}{a} + 2\beta^2 + O(\tfrac{\varepsilon}{a}, \beta^4)] \tag{3.30}$$

We present the results of investigating t_{FBA} and t_{SPA} on planar structures in Table 12, Table 13, and Table 14. Again, the dimensions of the planes are displayed and defined in model as $(a+1)\times(b+1)$, since we need to include a separate origin voxel equidistant from both sets of edges.

Our approximations for t_{FBA} successfully straddle the simulation-observed result for all planar structures, and our primary formula (Equation 3.27) is especially accurate for regular planes. The low relative error observed shows that the MFPT is a good measure for the typical travel time. On the other hand we once again encounter difficulties in approximating t_{SPA}; in no case are we able to estimate a t_{SPA} with confidence that it will match the observed result. Furthermore, since we find that the larger and less balanced the structure the worse the MFPT as a measure of the typical narrow escape time, it becomes difficult to justify the exploration of narrow escape problems on these Spatial Kappa structures.

Plane Size	Connectivity (γ)	Est. t_{FBAmin}	Est. t_{FBA}	Est. t_{FBAmax}	Obs. t_{FBA}
21 x 21	4-way Square (γ = 840.0)	0.0298	0.0360	0.0397	0.0356
21 x 21	8-way Compass (γ = 280.0)	0.0298	0.0360	0.0397	0.0350
21 x 21	4-way Diagonal (γ = 420.0)	0.0298	0.0360	0.0397	0.0355
41 x 41	4-way Square (γ = 840.0)	0.1190	0.1402	0.1588	0.1404
41 x 41	8-way Compass (γ = 280.0)	0.1190	0.1402	0.1588	0.1413
41 x 41	4-way Diagonal (γ = 420.0)	0.1190	0.1402	0.1588	0.1416
21 x 81	4-way Square (γ = 840.0)	0.0298	0.0810	0.2481	0.0597
21 x 81	8-way Compass (γ = 280.0)	0.0298	0.0810	0.2481	0.0596
21 x 81	4-way Diagonal (γ = 420.0)	0.0298	0.0810	0.2481	0.0594
11 x 161	4-way Square (γ = 840.0)	0.0074	0.0647	0.7084	0.0150
11 x 161	8-way Compass (γ = 280.0)	0.0074	0.0647	0.7084	0.0150
11 x 161	4-way Diagonal (γ = 420.0)	0.0074	0.0647	0.7084	0.0152
81 x 81	4-way Square (γ = 840.0)	0.4762	0.5535	0.6350	0.5597
81 x 81	8-way Compass (γ = 280.0)	0.4762	0.5535	0.6350	0.5548
81 x 81	4-way Diagonal (γ = 420.0)	0.4762	0.5535	0.6350	0.5631

Table 12: Estimated and observed mean first passage times (full border absorption) of 10000 T on a plane with $R_{vox} = 0.005\mu m$, and plane size, connectivity, and γ as shown. In all cases the observed values fall within the expected ranges; the more regular the structure, the better we can approximate it via Equation 3.27.

Plane Size	Connectivity (γ)	Est. t_{SPA}	Est. $t_{SPAedge}$	Obs. $t_{SPAedge}$
21 x 21	4-way Square ($\gamma = 840.0$)	0.6564	0.4392	0.8343
21 x 21	8-way Compass ($\gamma = 280.0$)	0.6564	0.4392	0.7769
21 x 21	4-way Diagonal ($\gamma = 420.0$)	0.6564	0.4392	0.5526
41 x 41	4-way Square ($\gamma = 840.0$)	3.0618	2.0867	3.1671
41 x 41	8-way Compass ($\gamma = 280.0$)	3.0618	2.0867	3.2636
41 x 41	4-way Diagonal ($\gamma = 420.0$)	3.0618	2.0867	2.6674
21 x 81	4-way Square ($\gamma = 840.0$)	3.2001	0.4392 / 9.6670	2.8574 / 4.0088
21 x 81	8-way Compass ($\gamma = 280.0$)	3.2001	0.4392 / 9.6670	2.9284 / 4.4741
21 x 81	4-way Diagonal ($\gamma = 420.0$)	3.2001	0.4392 / 9.6670	2.1433 / 4.1788
11 x 161	4-way Square ($\gamma = 840.0$)	3.5269	0.0892 / 43.9491	2.5565 / 6.5241
11 x 161	8-way Compass ($\gamma = 280.0$)	3.5269	0.0892 / 43.9491	2.4915 / 8.2661
11 x 161	4-way Diagonal ($\gamma = 420.0$)	3.5269	0.0892 / 43.9491	1.6980 / 9.3439
81 x 81	4-way Square ($\gamma = 840.0$)	13.95894	9.6670	7.1098
81 x 81	8-way Compass ($\gamma = 280.0$)	13.9589	9.6670	8.6345
81 x 81	4-way Diagonal ($\gamma = 420.0$)	13.9589	9.6670	8.5271

Table 13: Estimated and observed mean first passage times (single point edge absorption) of 10000 T on a plane with $R_{vox} = 0.005\mu m$, and plane size, connectivity, and γ as shown. Although our ballpark figures are correct, we are unable to reproduce the expected t_{SPA} with any degree of accuracy.

Plane Size	Connectivity (γ)	Est. t_{SPA}	Est. $t_{SPAcorner}$	Obs. $t_{SPAcorner}$
21 x 21	4-way Square (γ = 840.0)	0.6564	0.8479	1.1213
21 x 21	8-way Compass (γ = 280.0)	0.6564	0.8479	2.1764
21 x 21	4-way Diagonal (γ = 420.0)	0.6564	0.8635	1.3998
41 x 41	4-way Square (γ = 840.0)	3.0618	5.1043	4.0001
41 x 41	8-way Compass (γ = 280.0)	3.0618	5.1043	6.8274
41 x 41	4-way Diagonal (γ = 420.0)	3.0618	5.1043	5.6972
21 x 81	4-way Square (γ = 840.0)	3.2001	6.4408	4.5123
21 x 81	8-way Compass (γ = 280.0)	3.2001	6.4408	7.2223
21 x 81	4-way Diagonal (γ = 420.0)	3.2001	6.4408	6.4182
11 x 161	4-way Square (γ = 840.0)	3.5269	13.8729	6.7340
11 x 161	8-way Compass (γ = 280.0)	3.5269	13.8729	9.6699
11 x 161	4-way Diagonal (γ = 420.0)	3.5269	13.8729	9.6930
81 x 81	4-way Square (γ = 840.0)	13.9589	27.2029	8.3928
81 x 81	8-way Compass (γ = 280.0)	13.9589	27.2029	13.1755
81 x 81	4-way Diagonal (γ = 420.0)	13.9589	27.2029	13.3467

Table 14: Estimated and observed mean first passage times (single point corner absorption) of 10000 T on a plane with $R_{vox} = 0.005\mu m$, and plane size, connectivity, and γ as shown. Again although our ballpark figures are mostly correct, we are unable to reproduce the expected t_{SPA} with any degree of accuracy.

The results of this section show that, although we believe that Spatial Kappa is accurately modelling the diffusion of agents across a lattice-like voxel structure, we find it unable to accurately reproduce the expected parameters of a narrow escape problem on the microscopic scale. Partitioning a space into channel-connected voxels appears to be insufficient for recreating the complex internal dynamics of biophysical structures (in particular narrow escape through an aperture).

3.3 In Three Dimensions

Derivations for three-dimensional formulae for t_{FBA} are similarly derived to those for the two-dimensional case. Again we refer to canonical formulae regarding the narrow escape problem to calculate t_{SPA}. Our intent in this section is to see whether we observe the same problems in three dimensions what we have encountered in two dimensions, for commonly modelled structures such as spheres.

3.3.1 Connectivity in Three Dimensions

We investigate the two examples of three-dimensional connectivity pre-provided in Spatial Kappa. Again using Equation 3.13 and Equation 3.14, we calculate the covariance matrices and hence the stochastic translocation rates of (1) 6-way cardinal lattice, or FaceNeighbour connectivity (Equation 3.31), and (2) 26-way compass lattice, or Neighbour connectivity (Equation 3.32).

$$
\begin{gathered}
(X^1,X^2,X^3) = \tfrac{1}{6}((-h,0,0)+(h,0,0)+(0,-h,0)+(0,h,0)+(0,0,-h)+(0,0,h)) \\
\Sigma(\varepsilon) = \begin{pmatrix} (\frac{2h^2}{6})-(0)(0) & 0-(0)(0) & 0-(0)(0) \\ 0-(0)(0) & (\frac{2h^2}{6})-(0)(0) & 0-(0)(0) \\ 0-(0)(0) & 0-(0)(0) & (\frac{2h^2}{6}) \end{pmatrix} = \begin{pmatrix} \frac{h^2}{3} & 0 & 0 \\ 0 & \frac{h^2}{3} & 0 \\ 0 & 0 & \frac{h^2}{3} \end{pmatrix} \\
\gamma^* \cdot \Sigma(\varepsilon) = 6\gamma \cdot \Sigma(\varepsilon) = 2h^2\gamma I = 2DI \\
\gamma = \tfrac{D}{h^2}
\end{gathered}
\tag{3.31}
$$

$$
\begin{gathered}
(X^1,X^2,X^3) = \tfrac{1}{26}((-h,-h,-h)+(-h,-h,0)+(-h,-h,h)+(-h,0,-h)+(-h,0,0) \\
+(-h,0,h)+(-h,h,-h)+(-h,h,0)+(-h,h,h)+(0,-h,-h)+(0,-h,0)+(0,-h,h) \\
+(0,0,-h)+(0,0,h)+(0,h,-h)+(0,h,0)+(0,h,h)+(h,-h,-h)+(h,-h,0) \\
+(h,-h,h)+(h,0,-h)+(h,0,0)+(h,0,h)+(h,h,-h)+(h,h,0)+(h,h,h)) \\
\Sigma(\varepsilon) = \begin{pmatrix} (\frac{18h^2}{26})-(0)(0) & 0-(0)(0) & 0-(0)(0) \\ 0-(0)(0) & (\frac{18h^2}{26})-(0)(0) & 0-(0)(0) \\ 0-(0)(0) & 0-(0)(0) & (\frac{18h^2}{26})-(0)(0) \end{pmatrix} = \begin{pmatrix} \frac{9h^2}{13} & 0 & 0 \\ 0 & \frac{9h^2}{13} & 0 \\ 0 & 0 & \frac{9h^2}{13} \end{pmatrix} \\
\gamma^* \cdot \Sigma(\varepsilon) = 26\gamma \cdot \Sigma(\varepsilon) = 18h^2\gamma I = 2DI \\
\gamma = \tfrac{D}{9h^2}
\end{gathered}
\tag{3.32}
$$

3.3.2 Three Dimensions: Spheres

Once again let us assume a regular sphere of radius $n = 10$ voxels and $0.1\mu m$, such that $R_{vox} = 0.005\mu m$. We observe the translocation of our travelling agent T with diffusion coefficient $D = 0.084\frac{\mu m^2}{s}$, towards a fully absorbing boundary and a boundary with a single absorbing voxel located directly 'north' of the origin at the centre of the sphere. With regards to the single point absorption, our question is whether we can better our performance in two dimensions in three, or whether we encounter the same problems.

The t_{FBA} for an agent T to travel in three dimensions from its origin to the absorbing border, according to Equation 3.3, is given in Equation 3.33.

$$
\begin{aligned}
t_{FBA} &= \tfrac{R^2}{2dD} \\
&= \tfrac{(0.1)^2}{2(3)(0.084)} \\
&= 0.0198
\end{aligned}
\tag{3.33}
$$

To calculate the t_{SPA} we once again turn to equations presented by Singer, Schuss, and Holcman [46, 76, 77] and verified by Cheviakov et al. [11]. In particular, the narrow escape problem on a sphere may be approximated by Equation 3.34, where $|\Omega|$ is the volume of the sphere and ε the radius of the absorbing circle on the sphere boundary. The first term is highly dominant in the equations for three dimensions, unlike in the two-dimensional case in which the second term plays a large moderating role.

$$
\begin{aligned}
t_{SPA} &= \tfrac{|\Omega|}{4\varepsilon D}[1+\tfrac{\varepsilon}{\pi R}ln\tfrac{R}{\varepsilon}+o(\tfrac{\varepsilon}{R}ln\tfrac{R}{\varepsilon})] \\
&= \tfrac{\frac{4}{3}\pi(0.1)^3}{(4)(0.005)(0.084)}[1+\tfrac{(0.005)}{\pi(0.1)}ln\tfrac{(0.1)}{(0.005)}+o(\tfrac{(0.005)}{(0.1)}ln\tfrac{(0.1)}{(0.005)})] \\
&\sim 2.6122
\end{aligned}
\tag{3.34}
$$

We continue to observe the general trend of our results so far in Table 15. Our Spatial Kappa simulations match well with the expected times derived for full border absorption. On the other hand the formulae underestimate our observed results for single point absorption.

Connectivity	γ	t_{FBA}	t_{SPA}
6-way Cardinal (1)	840.0	0.0196	3.4909
26-way Compass (2)	93.3	0.0203	4.1882

Table 15: Mean first passage times of 10000 T on a sphere with radius $n = 10$ voxels and $R_{vox} = 0.005\mu m$; connectivity, absorbing borders, and γ as shown. We see that the t_{FBA} match the expected result from Equation 3.33, but that the t_{SPA} deviate largely from Equation 3.34.

Radius	Connectivity (γ)	t_{FBA}	t_{SPA}
20 voxels (0.2 μm)	6-way Cardinal (γ = 840.0)	0.0780	3.4909
20 voxels (0.2 μm)	26-way Compass (γ = 93.3)	0.0798	4.1882
40 voxels (0.4 μm)	6-way Cardinal (γ = 840.0)	0.3121	10.8701
40 voxels (0.4 μm)	26-way Compass (γ = 93.3)	0.3197	12.4533

Table 16: Mean first passage times of 10000 T on a sphere with $R_{vox} = 0.005\mu m$; connectivity, absorbing borders, sphere radius, and γ as shown. We see that the t_{FBA} match the expected results from Equation 3.35 and Equation 3.37, but that the t_{SPA} deviate considerably from Equation 3.36 and Equation 3.38.

Building on these results, we once again compare full border absorption and single point absorption over different sphere diameters.

For spheres with 20 voxel radius:

$$\begin{aligned} t_{FBA} &= \frac{(0.2)^2}{2(3)(0.084)} \\ &= 0.0794 \end{aligned} \tag{3.35}$$

$$\begin{aligned} t_{SPA} &= \frac{\frac{4}{3}\pi(0.2)^3}{(4)(0.005)(0.084)}[1+\frac{(0.005)}{\pi(0.2)}ln\frac{(0.2)}{(0.005)}+o(\frac{(0.005)}{(0.2)}ln\frac{(0.2)}{(0.005)})] \\ &\sim 20.5321 \end{aligned} \tag{3.36}$$

For spheres with 40 voxel radius:

$$\begin{aligned} t_{FBA} &= \frac{(0.4)^2}{2(3)(0.084)} \\ &= 0.3175 \end{aligned} \tag{3.37}$$

$$\begin{aligned} t_{SPA} &= \frac{\frac{4}{3}\pi(0.4)^3}{(4)(0.005)(0.084)}[1+\frac{(0.005)}{\pi(0.4)}ln\frac{(0.4)}{(0.005)}+o(\frac{(0.005)}{(0.4)}ln\frac{(0.2)}{(0.005)})] \\ &\sim 162.3552 \end{aligned} \tag{3.38}$$

The results in Table 16 indicate that the t_{FBA} match the Stokes-Einstein relation well even in three dimensions. Once more, however, the results for t_{SPA} deviate considerably from the target

Radius	Connectivity (γ)	t_{FBA}
10 voxels (0.4 μm)	6-way Cardinal (γ = 52.5)	0.3146
10 voxels (0.4 μm)	26-way Compass (γ = 5.83)	0.3279
20 voxels (0.4 μm)	6-way Cardinal (γ = 210.0)	0.3101
20 voxels (0.4 μm)	26-way Compass (γ = 23.3)	0.3200
40 voxels (0.4 μm)	6-way Cardinal (γ = 840.0)	0.3121
40 voxels (0.4 μm)	26-way Compass (γ = 93.3)	0.3197

Table 17: Mean first passage times of 10000 T on a sphere with $R_{vox} = 0.005\mu m$; connectivity, absorbing borders, sphere radius, and γ as shown. The t_{FBA} scale properly across different sphere radii and for different connectivities.

result. Given that the error terms in Equation 3.34 are likely to be smaller than in the two-dimensional case, we believe that the reason for this is that the Spatial Kappa approximation of circular spheres in a lattice-like layout of voxels with channels and channel connectivities between them is not amenable to the analysis of narrow escape from their confines, especially given the abstractions such a structure makes near its surface.

Finally, we again ask whether scaling the number of voxels and translocation rate for a sphere with a given radius produces similar results. Given the computational time required to run single point absorption simulations in three dimensions, as well as the lack of success in our two-dimensional experiments and the three-dimensional t_{SPA} of Table 15 and Table 16, we observe this for full border absorption only. Thankfully, the results in Table 17 indicate that the t_{FBA} scales properly.

3.3.3 Three Dimensions: Planoids

We examine diffusion on a three-dimensional $a \times b \times c$ planoid by extending the formulae for full border absorption from the two-dimensional planar case. Again we assume an origin voxel that distorts the Spatial Kappa definition of the planoid as $(a+1) \times (b+1) \times (c+1)$, such that $a < b < c$ and distance is measured by the number of 'hops' (translocations) the travelling agent T is forced to make before encountering a boundary. Once again we take N_{vp} to indicate the number of voxels on the outside faces of the plane and $h = 2R_{vox}$ the diameter of a voxel (or the length of a 'hop' along the perimeter), and multiply the weight of the voxel (minimum voxel distance to origin, or actual voxel distance to origin) by the expected time of arrival at each voxel (as calculated by the Stokes-Einstein relation). Equation 3.39 simply enumerates this over each voxel in the $a \times b \times c$ planoid.

Unfortunately, no corresponding benchmark formula is yet available for narrow escape (single point absorption) on three-dimensional cuboid domains.

Planoid Size	Connectivity (γ)	Est. t_{FBAmin}	Est. t_{FBA}	Obs. t_{FBA}
21 x 21 x 21	6-way Cardinal (γ = 840.0)	0.0198	0.0253	0.0269
21 x 21 x 21	26-way Compass (γ = 93.3)	0.0198	0.0253	0.0269
41 x 41 x 41	6-way Cardinal (γ = 840.0)	0.0794	0.1016	0.1079
41 x 41 x 41	26-way Compass (γ = 93.3)	0.0794	0.1016	0.1078
11 x 81 x 81	6-way Cardinal (γ = 840.0)	0.0050	0.0339	0.0151
11 x 81 x 81	26-way Compass (γ = 93.3)	0.0050	0.0339	0.0151
21 x 41 x 81	6-way Cardinal (γ = 840.0)	0.0198	0.0587	0.0542
21 x 41 x 81	26-way Compass (γ = 93.3)	0.0198	0.0587	0.0551
21 x 21 x 161	6-way Cardinal (γ = 840.0)	0.0198	0.0890	0.0354
21 x 21 x 161	26-way Compass (γ = 93.3)	0.0198	0.0890	0.0351
11 x 41 x 161	6-way Cardinal (γ = 840.0)	0.0050	0.0452	0.0148
11 x 41 x 161	26-way Compass (γ = 93.3)	0.0050	0.0452	0.0153
81 x 81 x 81	6-way Cardinal (γ = 840.0)	0.3175	0.4066	0.4218
81 x 81 x 81	26-way Compass (γ = 93.3)	0.3175	0.4066	0.4278

Table 18: Estimated and observed mean first passage times (full border absorption) of 10000 T on a planoid with $R_{vox} = 0.005\mu m$, and planoid size, connectivity, and γ as shown. The formula given in Equation 3.39 is not as accurate as its two-dimensional counterpart, but for balanced structures still gives a reasonable approximation of the t_{FBA}

.

$$
\begin{aligned}
t_{FBA} = \; & \tfrac{1}{N_{vp}\cdot 6D}\Big(2(\tfrac{a}{2})^2 + 2(\tfrac{(\frac{a}{2})}{(\frac{b}{2})})(\tfrac{b}{2})^2 + 2(\tfrac{(\frac{a}{2})}{(\frac{c}{2})})(\tfrac{c}{2})^2 + 8(\tfrac{(\frac{a}{2})}{(\sqrt{(\frac{a}{2})^2+(\frac{b}{2})^2+(\frac{c}{2})^2})})((\tfrac{a}{2})^2+(\tfrac{b}{2})^2+(\tfrac{c}{2})^2) \\
& +4(\tfrac{(\frac{a}{2})}{(\sqrt{(\frac{a}{2})^2+(\frac{b}{2})^2})})((\tfrac{a}{2})^2+(\tfrac{b}{2})^2) + 8\textstyle\sum_{i=1}^{\frac{c}{2h}-1}(\tfrac{(\frac{a}{2})}{(\sqrt{(\frac{a}{2})^2+(\frac{b}{2})^2+(ih)^2})})((\tfrac{a}{2})^2+(\tfrac{b}{2})^2+(ih)^2) \\
& +4(\tfrac{(\frac{a}{2})}{(\sqrt{(\frac{a}{2})^2+(\frac{c}{2})^2})})((\tfrac{a}{2})^2+(\tfrac{c}{2})^2) + 8\textstyle\sum_{i=1}^{\frac{b}{2h}-1}(\tfrac{(\frac{a}{2})}{(\sqrt{(\frac{a}{2})^2+(\frac{c}{2})^2+(ih)^2})})((\tfrac{a}{2})^2+(\tfrac{c}{2})^2+(ih)^2) \\
& +4(\tfrac{(\frac{a}{2})}{(\sqrt{(\frac{b}{2})^2+(\frac{c}{2})^2})})((\tfrac{b}{2})^2+(\tfrac{c}{2})^2) + 8\textstyle\sum_{i=1}^{\frac{a}{2h}-1}(\tfrac{(\frac{a}{2})}{(\sqrt{(\frac{b}{2})^2+(\frac{c}{2})^2+(ih)^2})})((\tfrac{b}{2})^2+(\tfrac{c}{2})^2+(ih)^2) \\
& +4\textstyle\sum_{i=1}^{\frac{b}{2h}-1}(\tfrac{(\frac{a}{2})}{(\sqrt{(\frac{a}{2})^2+(ih)^2})})((\tfrac{a}{2})^2+(ih)^2) + 4\sum_{i=1}^{\frac{c}{2h}-1}(\tfrac{(\frac{a}{2})}{(\sqrt{(\frac{a}{2})^2+(ih)^2})})((\tfrac{a}{2})^2+(ih)^2) \\
& +8\textstyle\sum_{i=1}^{\frac{b}{2h}-1}\sum_{j=1}^{\frac{c}{2h}-1}(\tfrac{(\frac{a}{2})}{(\sqrt{(\frac{a}{2})^2+(ih)^2+(jh)^2})})((\tfrac{a}{2})^2+(ih)^2+(jh)^2) \\
& +4\textstyle\sum_{i=1}^{\frac{a}{2h}-1}(\tfrac{(\frac{a}{2})}{(\sqrt{(\frac{b}{2})^2+(ih)^2})})((\tfrac{b}{2})^2+(ih)^2) + 42\sum_{i=1}^{\frac{c}{2h}-1}(\tfrac{(\frac{a}{2})}{(\sqrt{(\frac{b}{2})^2+(ih)^2})})((\tfrac{b}{2})^2+(ih)^2) \\
& +8\textstyle\sum_{i=1}^{\frac{a}{2h}-1}\sum_{j=1}^{\frac{c}{2h}-1}(\tfrac{(\frac{a}{2})}{(\sqrt{(\frac{b}{2})^2+(ih)^2+(jh)^2})})((\tfrac{b}{2})^2+(ih)^2+(jh)^2) \\
& +4\textstyle\sum_{i=1}^{\frac{a}{2h}-1}(\tfrac{(\frac{a}{2})}{(\sqrt{(\frac{c}{2})^2+(ih)^2})})((\tfrac{c}{2})^2+(ih)^2) + 4\sum_{i=1}^{\frac{b}{2h}-1}(\tfrac{(\frac{a}{2})}{(\sqrt{(\frac{c}{2})^2+(ih)^2})})((\tfrac{c}{2})^2+(ih)^2) \\
& +8\textstyle\sum_{i=1}^{\frac{a}{2h}-1}\sum_{j=1}^{\frac{b}{2h}-1}(\tfrac{(\frac{a}{2})}{(\sqrt{(\frac{c}{2})^2+(ih)^2+(jh)^2})})((\tfrac{c}{2})^2+(ih)^2+(jh)^2)\Big)
\end{aligned}
\tag{3.39}
$$

Table 18 shows that our formula again successfully approximates t_{FBA} for regular planoids,

Sphere Radius (R)	Absorbing Radius (a)	Est. t	Obs. t
10 voxels (0.1 μ m)	0.5 voxels (0.005 μ m)	0.7132	3.1405
10 voxels (0.1 μ m)	1.5 voxels (0.015 μ m)	0.4517	1.5313
20 voxels (0.2 μ m)	0.5 voxels (0.005 μ m)	3.5132	11.6258
20 voxels (0.2 μ m)	1.5 voxels (0.015 μ m)	2.4669	7.3482
40 voxels (0.4 μ m)	0.5 voxels (0.005 μ m)	16.6934	26.1905
40 voxels (0.4 μ m)	1.5 voxels (0.015 μ m)	12.5082	20.8944

Table 19: Estimated and observed mean first passage times of 10000 T on the surface of a sphere with $R_{vox} = 0.005\mu m$, 26-way Compass (Neighbour) connectivity, $\gamma = 93.3$, and sphere radius and absorbing radius (circle centred at south pole) as shown. Unfortunately, we find that we are unable to duplicate the estimated mean via simulation.

but for irregular structures the MFPT is likely to tend towards t_{FBAmin}. The greater the irregularity of the structure, the greater this tendency.

3.3.4 Traversal on a Three Dimensional Object: Sphere Surfaces

We now take a quick look at an interesting subclass of the two- and three-dimensional diffusion explored previously. The question we ask is: can we verify a model of transmembrane protein movement along the surface of a membrane (i.e. two-dimensional traversal on the surface of a three-dimensional object)?

We find in Holcman and Schuss [46] a formula (Equation 3.40) describing the MFPT for Brownian motion on the surface of a sphere of radius R to an absorbing circle centred on the south pole with small radius $a = R\sin\frac{\delta}{2}$. θ is the angle between the origin voxel and the south-north axis of the sphere. Since we locate the origin on the north pole in our simulations, $\sin\frac{\theta}{2} = 1$. As always, the diffusion coefficient D of our travelling agent T is $0.084\frac{\mu m^2}{s}$.

$$\begin{aligned} t &= \frac{2R^2}{D} ln \frac{\sin\frac{\theta}{2}}{\sin\frac{\delta}{2}} \\ &= \frac{2R^2}{D} ln \frac{R}{a} \end{aligned} \tag{3.40}$$

We investigate Equation 3.40 for a variety of sphere and absorbing radii in Table 19. Unfortunately, we find that the formula consistently and considerably underestimates the observed simulation value for the MFPT. Again we posit that this is due to the approximation of the spherical structure in the lattice-like Spatial Kappa grid, given that the results obtained deviate greatly from the expected MFPT should T be travelling on a flat two-dimensional surface (at a first approximation for a sphere with radius 10 voxels, $t_{2D} = \frac{(\pi R)^2}{2dD} = 0.2937$) as well.

Sphere Radius	Column Length (L)	Column Radius (a)	Est. t	Obs. t
10 voxels (0.1 μ m)	10 voxels (0.1 μ m)	1 voxel (0.01 μ m)	1.3062	11.0476
10 voxels (0.1 μ m)	10 voxels (0.1 μ m)	5 voxels (0.05 μ m)	0.3089	1.1791
10 voxels (0.1 μ m)	20 voxels (0.2 μ m)	1 voxel (0.01 μ m)	1.4848	20.6357
10 voxels (0.1 μ m)	20 voxels (0.2 μ m)	5 voxels (0.05 μ m)	0.4874	2.1123
10 voxels (0.1 μ m)	40 voxels (0.4 μ m)	1 voxel (0.01 μ m)	2.1990	39.0726
10 voxels (0.1 μ m)	40 voxels (0.4 μ m)	5 voxels (0.05 μ m)	1.2017	4.1926
20 voxels (0.2 μ m)	10 voxels (0.1 μ m)	1 voxel (0.01 μ m)	10.0328	56.8358
20 voxels (0.2 μ m)	10 voxels (0.1 μ m)	5 voxels (0.05 μ m)	2.0542	7.5651
20 voxels (0.2 μ m)	20 voxels (0.2 μ m)	1 voxel (0.01 μ m)	10.2114	79.7333
20 voxels (0.2 μ m)	20 voxels (0.2 μ m)	5 voxels (0.05 μ m)	2.2328	12.5316
20 voxels (0.2 μ m)	40 voxels (0.4 μ m)	1 voxel (0.01 μ m)	10.9257	110.9491
20 voxels (0.2 μ m)	40 voxels (0.4 μ m)	5 voxels (0.05 μ m)	2.9470	22.9462

Table 20: Estimated and observed mean first passage times of 1000 T on the surface of a sphere with $R_{vox} = 0.005\mu m$, 26-way Compass (Neighbour) connectivity, $\gamma = 93.3$, and sphere radius and absorbing radius (circle centred at south pole) as shown.

3.3.5 Three Dimensions: Spines

For our final investigation of three-dimensional structures in Spatial Kappa, we approach the neuronal dendritic spine. A dendritic spine may be approximated [73] by a spherical head with volume V connected to a cylindrical shaft of length L and radius a, where the radius of the neck is small in comparison to that of the spine head. The MFPT for a diffusing agent T to travel from the head to the tail of such a dendritic spine may be decomposed into the mean time to find the escaping window from the head plus the mean time to escape the shaft, by making the simplifying assumption that upon entering the shaft T never returns to the head. We formalise this escape time in Equation 3.41.

$$t = \frac{V}{4aD} + \frac{L^2}{2D} \tag{3.41}$$

We investigate Equation 3.41 for a variety of sphere radii and column sizes in Table 20. Our obtained results grossly exceed the target formulae, and even results that should bear some resemblance to each other (for example simply varying the length of the cylinder whilst keeping its radius and the sphere radius equal) lack coherence. We posit that the assumption made in the equation – that upon entering the shaft T never returns to the head – is not a biologically plausible assumption to make. Even when we hardcode this assumption into the model, however, the equations are not a good approximation of the overall time due to the discrepancy in

narrow escape times from the sphere portion to the cylindrical aperture of the spine.

Throughout this section, we have shown that Spatial Kappa's approximation of a grid-like layout of voxels with channels and channel connectivities between them handles simple omni-directional diffusion well. On the other hand, we have time and again encountered the issue that it is not amenable to the analysis of complex spatial problems such as the narrow escape problem. We look at overcoming this issue in the next chapter.

Chapter 4

Extensions and Further Definitions for Spatial Kappa

In this chapter we...

- ❖ Review historical methodologies for defining location and compartment in formal languages for biology.
- ❖ Identify the strengths of the current Cartesian voxel definition of compartments in modelling non-homogeneous compartments.
- ❖ Propose a further extension to refine these strengths with the notion of voxel occupancy.
- ❖ Propose an alternative definition of space as a Kappa-like complex (site graphs with attributes stored as state).
- ❖ Demonstrate the strengths of space as a Kappa-like complex in theoretical examples that show the automated derivation of rates and the re-writing of space on the fly.
- ❖ Propose a second alternative definition of space, the automated generation of compartments and channels using Cayley graphs.
- ❖ Demonstrate the strength of Cayley graphs in a theoretical example where we walk polymeric agents through complex, automatically generated space.
- ❖ Outline provisional operational semantics for the alternative definitions, and consider how we might automate their derivation.

Spatial Kappa [81, 83, 84] supports spatial concepts in the context of rule-based modelling. Currently it defines space in terms of voxel-composed compartments, channels of diffusion between these compartments, and rules for agents to translocate along these channels. We made

efforts in the previous chapter to verify the soundness of Spatial Kappa with respect to well-established results in diffusivity theory and narrow escape problems. We had varying success, observing that Spatial Kappa is capable of imitating reaction-diffusion for the Stokes-Einstein relation, but also that its approximation of space (as a lattice structure of voxels with channels and channel connectivities) is insufficiently precise to recreate the complex internal dynamics (for example narrow escape through an aperture) of cells and other biophysical structures.

Recent, further efforts have focused on the future sustainability of Spatial Kappa and improvement of its computational efficiency. As seen in the previous chapter, models with large numbers of voxels or large numbers of agents moving across said voxels take an inordinately long time to simulate, and we wish to alleviate this problem via algorithm refinement and an optimised implementation. We have also made progress on revamping and evolving the syntax and semantics of Spatial Kappa to reflect the needs of the community utilising it. This chapter addresses our work in streamlining the process of defining spatial structures. The work presented, in particular the sections regarding Cayley graphs and operational semantics, was undertaken under the close supervision of Vincent Danos.

The main consideration for our syntactic choices is a *decoupling principle* specifying that the syntax should make transparent both the Kappa projection $K(M)$ of a model M and its spatial projection $S(M)$. In other words, there exists a need to cleanly separate the transformation (reaction) semantics of the model from the translocation (diffusion) semantics. One of our future goals is the automatic derivation of an SDE model (with relations between parameter sets) corresponding to the spatial projection of a model $S(M)$, which will allow us to perform numerical calibration and verification of the Spatial Kappa model against its SDE counterpart.

We begin by reviewing existing methodologies for defining locations and compartments in formal languages for biology, including particle-based methodologies and constructive solid geometry (CSG). Alongside this, we consider the current implementation of the spatial projection $S(M)$ of a Spatial Kappa model M, identifying its strengths and weaknesses. We then provide alternative methodologies, breaking down $S(M)$ into a set of voxels V and a set of channels C, including a closer look at how best either to generate these sets automatically or to provide a methodology allowing modellers to define structures suiting their particular modelling purpose. Although we have yet to conclude as to the best choice or the efficiency of these methodologies, it is our hope that they provide some insight as to the future of the language. We finish off by providing a provisional updated operational semantics for Spatial Kappa – given a specification for the spatial projection $S(M)$ of a model M, how may we make use of this in a standard Kappa simulation?

4.1 Space in Formal Languages for Biology

Spatial homogeneity is a convenient assumption in the modelling of biological processes, but not always a realistic one. From the literature of the last decade, we can identify three major approaches to simulating biochemical systems with spatial and stochastic detail.

The abstract notion of biological compartments as an object has existed since the creation of ontologies for genome and pathway databases; these compartments are easily organised into hierarchical structures, and may be either static or dynamic depending on the formal language. Examples of such definitions of space in formal languages for biology include the process calculi (BioAmbients, Brane calculi, and Bio-Pepa) described below.

In lattice methods, an explicit computational mesh (generally in two or three dimensions) is used to represent a cellular compartment. This mesh is populated with the molecular species that comprise the system, which are allowed to diffuse and react at user-defined rates. The lattice may represent microscopic (i.e. each lattice site may host at most one molecule) or mesoscopic domains. The latter may suffer from lattice-based artifacts such as the 'volume exclusion' effect, in which a domain is populated by a large number of molecules that would not physically fit. An example of a formal language utilising the lattice method for representing space is SpatialKappa; alternately, simulators such as MesoRD [42] also incorporate such a lattice representation of space in implementing the reaction-diffusion master equation.

In off-lattice or particle-based methods, individual molecules in the system have explicit spatial coordinates and are allowed to diffuse (in a random walk fashion) to new positions in the continuous space at each timestep. Such methods provide detailed simulations of complex systems and avoid lattice-based artifacts, but are computationally intensive since they track individual molecules rather than aggregating over a discretised space. An example of a simulator incorporating the particle-based method is Smoldyn [2].

In this section, we track the evolution of spatial descriptions in formal languages for biological modelling, beginning with early process calculi and progressing to modern extensions for rule-based modelling (including Spatial Kappa).

4.1.1 BioAmbients and Related Process Calculi

The BioAmbients calculus [70] was developed as a methodology for representing the localisation and compartmentalisation of molecules, including: the movement of molecules between compartments, the dynamic rearrangement of cellular compartments, and interaction between molecules in a compartmentalised setting. Notably it was the first attempt at a formal representation of biological compartment dynamics, building on hierarchical ontologies and data schemas (non-dynamic) and extended Petri net abstractions (either informal or non-biological). In BioAmbients, a system is represented as a hierarchy of nested ambients representing the

boundaries of compartments, which contain communicating processes whose actions specify the system dynamics. These actions might include location-specific reactions, complex formation, and the transport of small molecules between compartments. BioAmbients offers intuitive modelling of dynamic compartment hierarchies, but lacks explicit detailed methodologies for modelling (for example) membrane proteins, and also requires all molecules to be enclosed within an ambient for diffusive transport.

The closely related Brane calculi [8] thus focuses on precise membrane operations: membranes are not simply containers but also active entities responsible for coordinating specific activities. A system is represented as a set of nested membranes, which in turn are represented as sets of actions. Such actions might include the transport of small molecules across membranes, or the entry into, exit from, merging with, or splitting of membranes with other membranes. On the other hand, Brane calculi do not include a description of biochemical reactions.

Later efforts in spatial representation of process calculi [13] combine aspects of both of the above methodologies, enriching the Bio-Pepa process algebra with notions of compartments and explicit definitions of their enclosing membranes. This abstracts away from changing the structure of compartments over time, reasoning that current models did not require such a level of detail, but does allow for growth (i.e. changes in size).

4.1.2 Formal Languages Based on Bigraphs

Building on the process calculi described above, further work investigates bigraphs as a framework for developing rule-based languages for molecular biology, and for incorporating spatial localisation into these descriptions [20]. The C-calculus [18, 19] focuses on the modelling of proteins and membranes, thus capturing a class of spatial constraints within an idealised cell. In particular it describes compartments and proteins by using local membranes without committing to a specific global structure, and specifies a complete set of molecular interactions via a minimal set of generators. It also introduces the concept of a channel to connect topologically related volumes.

The distinct bigraphical approach taken here, of representing compartment structure separately from agent complexes, is carried forth in much of the theoretical basis for abstract rule-based spatial conventions explored below.

4.1.3 Spatial Conventions in Rule-Based Modelling

In both process calculi and bigraphs approaches, space in the system of interest is defined as an abstract object. Such a definition has also been applied to rule-based modelling, for example in cBNGL [41] and BioCham [7]. cBNGL also introduces the notion of volumetric scaling of reaction rates, allowing the effects of compartmentalisation to be naturally modelled by rules.

A similar approach of providing dynamic 'containment' structures in Kappa was introduced in [87].

Recent work has attempted instead to assign a more explicit structure to the cell under study. For example, the Stochastic Simulation Compiler (SSC) allows static compartments specified using constructive solid geometry (CSG), as well as diffusion between them and modelling of spatial and geometric effects [57]. Specialised membrane reactions, inspired by the Brane calculi, are incorporated into Kappa as the bioKappa calculus [55]. Klann et al. propose a coarse-grained spatial simulator combining particle-based reaction-diffusion with (non-spatial) rule-based modelling in [50], although they incur a high computational cost and further encounter a bottleneck in the efficiency of rule-based plug-ins for existing particle simulators.

For now we focus on low-cost and accessible alternative spatial definitions attempting to incorporate the most useful features of the specialised variants described above (i.e. volumetric scaling of reaction rates and the intuitive automated definitions of constructive solid geometry). We thus concentrate on the current lattice-based implementation of space as a Cartesian grid, and propose two abstract alternatives in the form of space as a Kappa-like complex and space as a Cayley graph.

4.2 The Existing Methodology: Space as a Cartesian Grid

The existing solution, adopted when first designing Spatial Kappa [83], defines a compartment as an array of voxels with Cartesian coordinates. The result is a lattice approximation of the specified shape providing a natural grid-like linkage structure between the individual voxels. Commonly used compartment shapes are pre-defined in both two and three dimensions and with both solid and hollow variants (for example circles, cuboids, spheres, and cylinders). Pre-defined channels (for example Neighbour and FaceNeighbour connectivity) are provided in a similar manner.

The advantages of this representation include ease of visualisation and of locating translocation and transformation rules, and the convenience of declarative channel definition. Its detailed and regular division of space is intuitive to both understand and use, and Cartesian grid coordinates make it equally easy and intuitive to access individual voxels for the purpose of locating agents and rules. The utility of the representation is clear when we examine non-homogeneous compartments in which we wish to track the explicit changes within each voxel.

However, the inherent rigidity and the implicitly unclear assumptions the representation makes for non grid-like (i.e. spherical) structures make it unsuitable for all applications. In particular, the Cartesian approximation of circular and spherical structures impacted on our calculation of the narrow escape problem in Spatial Kappa in Chapter 3.

Thus, although Cartesian grids remain a viable default option for modelling spaces that nat-

urally have a lattice structure, we present in the following sections two alternatives for defining V and C in a Spatial Kappa model M. Each option described has unique advantages and disadvantages; we do not attempt to arrive at a consensus as to the best method but instead present all of them for consideration. First, however, let us look at an extension to better take advantage of the strengths of the Cartesian representation.

4.2.1 Extension: Voxel Occupancy

The greatest strength of the Cartesian representation is that it methodically confronts the assumption of spatial homogeneity within a compartment that underlies much of Kappa semantics. For example, the question of whether *in vitro* observations accurately reflect *in vivo* behaviour is strongly linked to the issue of crowding within a cell, in which macromolecules jostle and compete with each other to carry out their biological functions [28, 64]. We address this disconnect between the conditions under which biological agents may be usefully studied and those under which they actually live, by proposing a definition of agents in existing Spatial Kappa that restricts how many of them may occupy a single voxel at a particular point in simulation time. The aim is to counteract the artefacts of lattice-based representation and to prevent a single large scaffold-based complex (consisting of many multiples of agents bound together perhaps in a way that would be infeasible *in vitro*) from being present in one spatially constrained voxel.

Let us assume that the voxel size is predefined by the user when they consider the spatial structure of the model. This definition may be by calculation, albeit not necessarily made explicit; for example, we might assume a default volume of 1.0 units and define agents in relation to this, with both voxels and agents having equivalent size across compartments. In this case, we need only specify in the agent definition the size of the agent in relation to the voxel in fractional terms to describe how much of a voxel an agent occupies. The volume here may be of arbitrary unit as long as it is kept consistent throughout the model. An agent without a defined volume in its declaration may be assumed to have negligible size in comparison to its fellows.

As an example, let us suppose a `Kinase` with volume 0.4, a `Phosphatase` with volume 0.2, and a `Target` with volume 0.3, as shown in Figure 33. Furthermore, let us assume a voxel with volume 1.0.

In this case, we wish for it to be physically impossible for a `Kinase` to enter a voxel already occupied by a `Kinase` and a `Target`. This might have tangible consequences for the model, as it would prevent `Kinase`s from entering voxels where their presence would not affect signal propagation, and instead efficiently funnel them towards unattended `Target`s elsewhere. Notice also that we have not explicitly defined how we might calculate the voxel or agent size here: it would be good practice to comment this within the model, although this is not necessary for practical implementation.

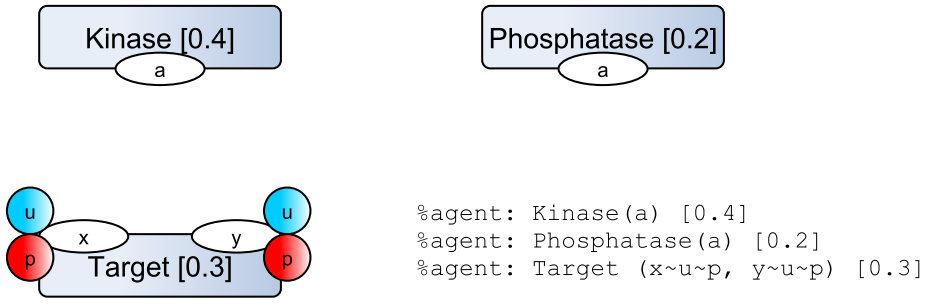


Figure 33: Updating the agents involved in the basic kinase-phosphatase model with volumes. Again, we show sample textual code (including a proposed method of definition for the agent volumes) corresponding to the visualisation at bottom right.

As possible semantics to enforce these voxel occupation constraints, we might modify the Next Subvolume method described by Johan Elf and Mans Ehrenberg [27] when it checks on translocation events and creation or deletion transformation events.

- When handling a translocation event, only update the occupancy states of the target and source voxels if the diffusion is valid (i.e. it does not cause an update to breach the target voxel's volume). If the diffusion is invalid, ignore current translocation and choose another translocation event.

- When handling a transformation event, update the occupancy state of the voxel if the transformation involves either creation or deletion effects. In particular, if the transformation involves a creation effect, then only proceed if it does not cause the update to breach the voxel's volume. If invalid, ignore current transformation and choose another transformation event.

Although we have only outlined a possible extension to the updated Spatial Kappa semantics, already we are able to begin asking further questions of spatially-aware models. For example, is it really necessary to enact such constraints? Is there any experimental evidence for the formation of large scaffold-based structures ('machines') in the first place, or should we instead minimise the rule assumptions and adopt a transient signalling ('ensemble') hypothesis [85]? Even if it is the case that 'machines' are observed, should limitations on their size emerge from the rates of diffusion rather than by assuming arbitrary size constraints?

We might ask further questions regarding the optimal voxel size to simulate a given model [48]. Too large, and we fall back beneath the umbrella assumption of spatial homogeneity which we are trying to work around; in essence, there is little point to the simulation in constraining voxel volume because each voxel may still contain an abundance of agents. Too small, and

translocation events dominate to the detriment of transformation events, meaning that simulation complexity increases; in the infinite limit, transformation events never occur and thus nothing of note ever happens in the simulation. Striking the optimal balance between the two, thus, is paramount.

We emphasise that we have not yet implemented the above, but we can begin devising a theoretical test suite of models. For example, to show translocation (diffusion) events in such a system we might consider a 'Funnel' in which all travelling agents are attracted towards the central voxel of a large disk and over time should accumulate like a liquid in a blocked funnel, or a 'Labyrinth' in which we seed a two-dimensional structure with large static molecules and aim to diffuse a travelling agent from entrance to exit. To show transformation (reaction) events we might consider a 'Black Hole' that adapts the funnel to delete agents at the centre voxel at a certain rate (consistently forcing the simulator to check if diffusion can occur), or a 'Bubbler' in which the central voxel of a 'Funnel' continuously creates agents instead until the structure is completely occupied.

We might eventually build practical models. One such model might be based on McGuffee and Elcock's model of the cytoplasm of *E.coli* [64], in which they take 51 macromolecules of known molecular weight including GFP introduced to the system, and observe the diffusion of the GFP through the other large macromolecules. This would not only have concrete biological application, but would also observe the timescales involved in the exchange of individual agents in comparison to actual *in vitro* data. From here we might extend to examples in which located (spatially non-homogeneous) scaffold proteins are thought to play a large role in transferring signals through a system, for example calcium diffusion in the synapse or the HOG web model introduced in Chapter 2.

In the future, we might even refine this methodology to include the possible automated derivation of model compartments and channels from experimental observations, i.e. from multiple images of the actual system over time. Proportioning space as a Cartesian grid allows us to easily (in theory, at least) limit the movement of agents based on image analysis. However, it may also prove to be too rigid and too inflexible for our purpose, and for our first alternative for defining space we look at a more abstract and more flexible method.

4.3 Space as a Kappa-like Complex

We now begin looking at further options for generating the voxel set V and the channel set C of a spatial projection for a Kappa model. Given that we have covered the lattice-based methodology above, and the computational costs and complexity of integrating Kappa with particle-based off-lattice representations of space, we thus concentrate on abstract alternatives that incorporate desired usability features such as inclusion of geometric and biophysical con-

straints and the automated derivation of diffusion rates and compartments.

We propose one such alternative given by visualising the spatial structure of the model as a linked complex of Kappa-like agents. This proposal shares many features with the approach adopted by the multi-level rule-based language ML-Rules [63], although it explicitly defines only one higher level of abstraction as the compartments of the model and makes use of cBNGL-like automated volumetric scaling of reaction rates. This hierarchical representation is also related to the reactive bigraphs of [18].

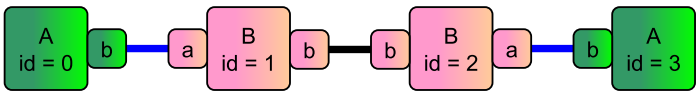
In this representation, voxels are defined as named agents with an identifier and further optional attributes such as voxel size. These attributes are represented as sites, with the state of the site the value of the attribute. The identifier is used to specify in a rule a subset of named voxels, thus defining the activity that occurs within them. Compartments can be thought of implicitly as connected regions of voxels with the same name, although we encode no explicit definition in the representation. Channels are 1-1 links between sites of voxel agents, tacitly typed by their endpoints. Directionality of the channel is provided by the translocation rules defined upon the channel, and thus all channels are considered inherently reversible (directionless).

Defining the spatial projection as a Kappa-like complex relies heavily on the modeller to correctly and explicitly structure the voxel set V, especially given that we currently lack any methodology for automating the generation of V from our underlying biological knowledge. Writing out complicated Kappa complexes by hand remains an error-prone process even in pure Kappa, and we propose in this thesis an alternative means by which to give potential users this capability (graphical definitions as shown in Figure 34). These may alleviate this problem considerably, but are only realistically applicable to one- and two-dimensional structures. Another possible solution is to pre-define commonly used structures as Kappa variables as is done already in the Cartesian approach, and to allow users to modify them according to need. Ideally we would be able to provide a more user-friendly methodology for the definition of complex spatial structures.

Based on this definition of V and C, translocation over the set of channels can easily be expressed in terms of pattern-matching over the connectivity structure. As an example, we might define the translocation rule shown in Figure 35, combining multiple 1-1 channels with a transformation (binding) effect.

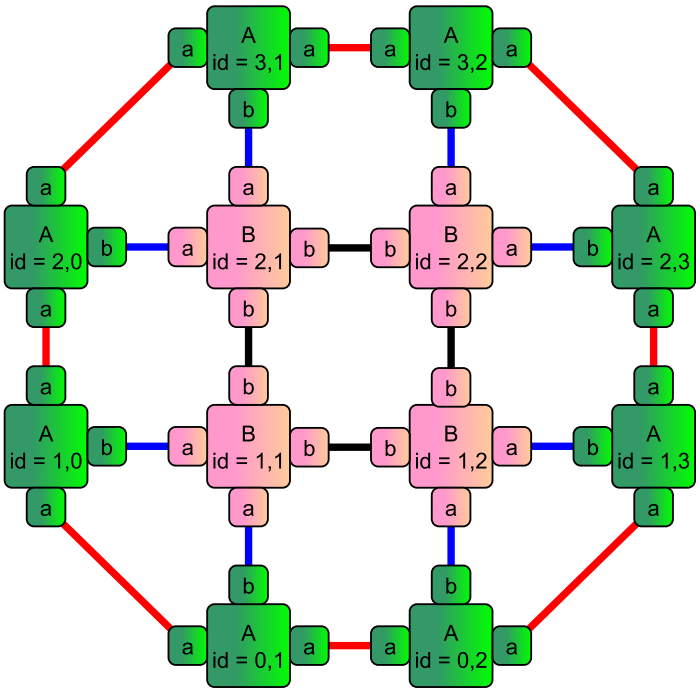
The advantages of such a representation include its intuitive definition and the ease by which we can associate physical and geometric attributes as states of sites belonging to the individual voxels (agents). On the other hand, as mentioned above, a heavy dose of modeller input is required to define the spatial structures V and C. We also abandon the fine-grained detail provided by the Cartesian methodology by choosing to work instead in a more abstract frame, in particular the coordinate system with which to access individual voxels.

A



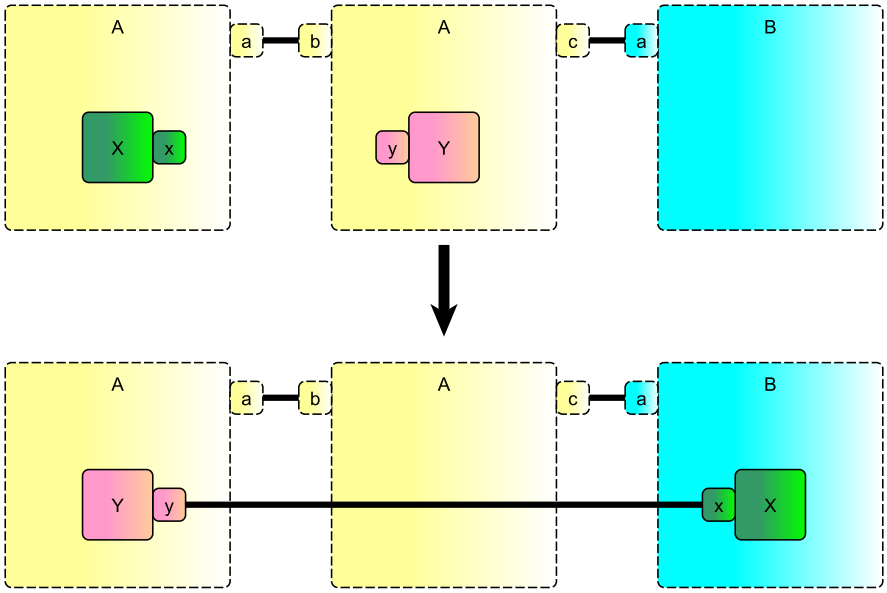

```
A(b!1),B(a!1,b!2),B(b!2,a!3),A(b!3)
```

B




```
A(a!1,a!8,b!9),A(a!1,a!2,b!10),A(a!2,a!3,b!11),A(a!3,a!4,b!12),
A(a!4,a!5,b!13),A(a!5,a!6,b!14),A(a!6,a!7,b!15),A(a!7,a!8,b!16),
B(a!9,a!16,b!17,b!20),B(a!10,a!11,b!17,b!18)
B(a!12,a!13,b!18,b!19),B(a!14,a!15,b!19,b!20)
```

Figure 34: (A) The graphical and textual definitions of a linear one-dimensional spatial structure defined as a Kappa-like complex. (B) A more complex two-dimensional structure; note that for the purposes of this visualisation we use a non-standard Kappa notation allowing for sets of sites with the same name. Note also that in all cases, the Kappa agent would be expected to keep track of an '`id`' site with appropriate state; we omit this from the textual definitions shown for the sake of clarity and of concentrating on the abstract connectivity. It is apparent that the textual definition of such spatial structures quickly becomes intractable.




```
A(a!1[X(x)]),A(b!1,c!2,[Y(y)]),B(a!2) -> A(a!1[Y(y!3)]),A(b!1,c!2[]),B(a!2[X(x!3)])
```

Figure 35: A sample two-channel combined translocation and binding rule upon a spatial structure defined as a Kappa-like complex. In the textual notation the top level of the rule represents the spatial structure (i.e. the connected voxels), whilst agents translate and interact within the square brackets attached to each voxel.

We now look at two modelling concepts of interest arising from defining the spatial structure as a Kappa-like complex.

4.3.1 Extrapolating Translocation Rates from Spatial Attributes

So far in this section, we have endeavoured to maintain separation between the geometric and biophysical attributes of the biological and spatial entities involved in the model and their abstract connectivity structure. In other words, although we have spent much time so far investigating how best to arrange voxels and channels to describe a particular spatial configuration, we have not yet given thought to how they might be used in an actual biophysical model.

Regardless of how the spatial projection is defined, quantitative geometric and biophysical attributes are implicitly included in a Spatial Kappa model via the translocation rates of various agents along channels between voxels. Modifying these rates manually is prone to error and miscalculation, and thus it would be desirable to automatically compile them from declared agent and voxel attributes. Users would not necessarily be tied to the rates thus derived, however, and would be given the option to adjust or override them as necessary (representing, for example, active transport mechanisms lacking detailed knowledge). They would serve as a baseline to represent unmodified diffusion under the constraints provided.

The question we now ask is as follows. Suppose we are given a set of voxels V and a set of compartments C, the attributes (i.e. voxel volume and channel width) thereof, and an agent population A with respective diffusivities. Defining space as a Kappa-like complex allows us to provide these attributes easily as states of the voxel agents. Can we adopt a well-defined diffusivity theory that allows us to infer provisional rates for the translocation rules within the model?

A simple example of such a theory (shown in Figure 36) involves calculating the typical time for an agent $\mathtt{A} \in A$ with diffusivity D to escape a voxel V with volume ω through a channel C with width ε. Such theories exist for the case where, for example, we assume that ε is small in comparison to the radius of V. In such a case, Rayleigh's formula [4] allows us to approximate the problem as a narrow escape from the voxel with typical time $\frac{\omega}{4D\varepsilon}$ and thus rate $4D\varepsilon/\omega$.

However, such well-defined diffusivity theories are hard to derive for more complicated examples, in particular for channels involving the complex translocation of agents to or from more than two voxels. Further work is necessary to clarify these cases.

Assuming that we are given the above we may now further outline methodologies for verifying the model rates thus obtained. We envisage two directions in which this may be possible: verification against an equivalent simulation process such as a linear SDE, and verification against approximate closed formulae. The former involves setting up and running a second simulation in addition to the primary Spatial Kappa simulation, with a corresponding expenditure in time, effort, and computing power as well as direct comparison of two stochastic processes.

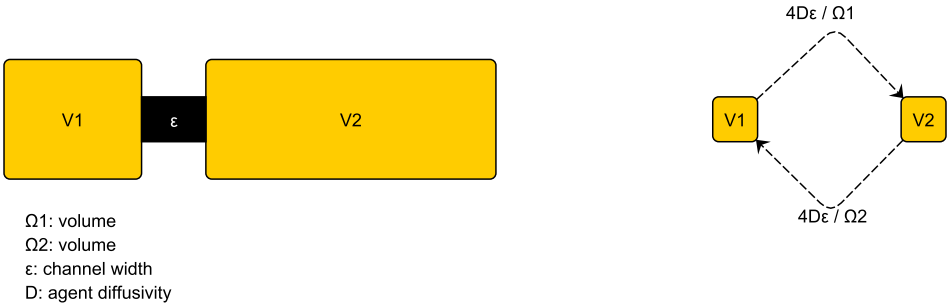


Figure 36: Two voxels of unequal size, connected by a 'relatively small' channel (aperture). On the right we display the corresponding connectivity structure, now reflecting geometric information (voxel volume ω, channel width ε) and physical attributes (agent diffusivity D) in the rates. Note that there is a corresponding loss of information – we are unable to recover ω, ε, and D from the rates.

The previous chapter shows that verification against closed formulae presumes the existence of analytic mean first passage time formulae such as the well-known Stokes-Einstein relation, or narrow escape times outlined by Singer et al. in [75, 76, 78] and related works [11, 69].

Automated extrapolation of translocation rates would be useful in, for example, the model of the mammalian circadian clock as presented in Chapter 2. The model here not only depends greatly on translocation of its agents for its dynamics, but also is derived from a differential model making use of actual biophysical constants. Hence we would also be able to use the above methodology to verify the rates used, and if necessary, to run the simulations based on newly derived parameters to see if the assumptions made still hold.

4.3.2 Dynamic Space: Growing, Shrinking, and Modifying during Simulation

A second modelling concept making best use of space defined as a Kappa-like complex is the modification of spatial structures over time. Since we can grow and prune the structures that define model space just as we would a standard Kappa complex (via the addition and deletion of binding links or channels), we are able to simulate the dynamic evolution of space, for example in the growth cycle of a cell model. Similarly, simply by modifying state of attribute sites, we can alter the physical constraints of the encoded spatial structure. This would not require complicated rewriting of the simulator, as we would simply reuse existing Kappa syntax and semantics on the level of the spatial projection rather than the Kappa projection. Hence, we would dedicate a separate section of the model to rules that act on the spatial structure of the model rather than on the low-level agents.

Eventually we might also consider extending this to the concept of multi-scale modelling [71].

This would be accomplished by defining a hierarchy in which we embed layers of agents within agents, for example proteins within cells within tissue. Given that each level of agents would essentially be running separate simulations with inter-dependent sets of rules and agents, the computational demands of such models would be immense and possibly beyond the current capabilities of Spatial Kappa.

One model we have already introduced that lends itself very well to this paradigm is the cooperative assembly model of chemoreceptors in *E.coli*, as described in Chapter 2. If we envision the agents interacting in this model as the spatial structure of some underlying system of protein interactions, we can imagine agent aggregation within the voxels – for example, the transmission of a signal agent – causing modification to the spatial complex. For example, given a population of `A` and `B` spatial agents, a configuration change in the agents within the `A` model is reflected back to its spatial parent (i.e. modification of a state on the spatial agent by a signalling end product), allowing it to bind to `B` and to begin construction of a large cluster. Unbinding would be caused by the counter-effect. Such a model would demonstrate the ability to rewrite space dynamically on an existing pool of spatial agents.

Alternatively we might consider a cell cycle model, as touched upon briefly above. In this case we would actually create additional spatial agents to increase the initial population, representing the growth of the cell based on events occurring within the voxels. Cell division would cause the loss (deletion) of half of these spatial agents as they split off to form a new cell. In contrast to the previous model, this would also demonstrate the creation and deletion of space.

We do not solve any of the aforementioned problems with defining space as a Kappa-like complex, in that the models described above all require the modeller to explicitly define the spatial structure of the model. With regards to the automated derivation of model compartments and channels from experimental observations, defining space as a Kappa-like complex gives us the ability to make more abstract specifications of model space as well as better use of macro-observables such as geometric and biophysical constraints. The ability to grow and shrink the spatial structure of the model over time is also a major benefit to this form of analysis. On the other hand, we again leave as an open question the exact methodology, likely based on cell imaging, by which we might accomplish this.

We next consider an alternative in which a model structure is procedurally generated without need for explicit user definition.

4.4 Space as a Cayley Graph

A fundamental method to provide a geometric structure for a group G is by specifying a list of generators S on G. The geometry of a generated group may then be visualised by its labelled Cayley graph: a directed coloured graph with edges coloured by the elements s of S and vertices

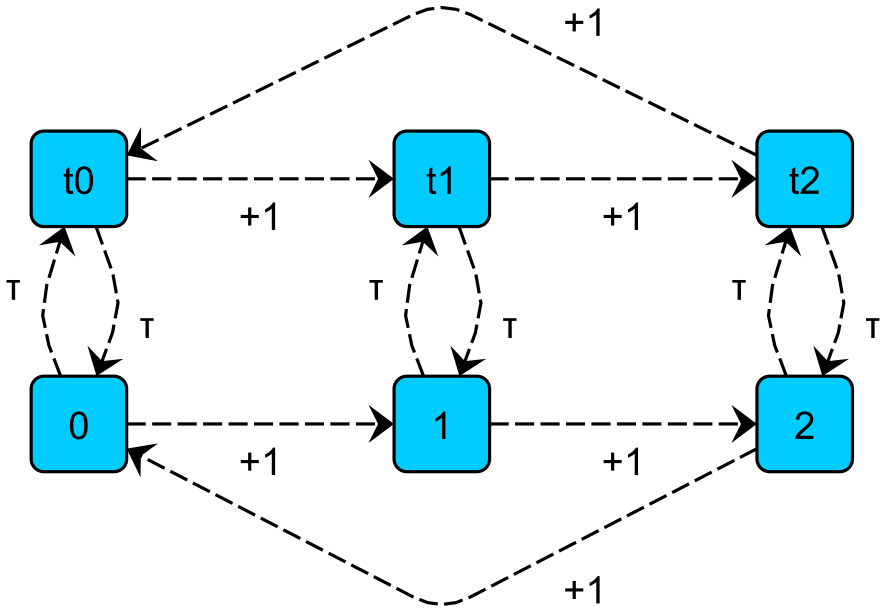


Figure 37: The Cayley graph generated from the group S_3 with generators transposition τ and $+1$. Six voxel compartments are formed, with twelve directed channels between them.

labelled by elements of x of G, with a directed edge of colour s from x to sx for each $x \in G$ and $s \in S$. For example, given the group S_3 with two generators, transposition (τ) and rotation ($+1$), we generate the Cayley graph shown in Figure 37.

In the context of providing a spatial projection $S(M)$ to a Spatial Kappa model M, we might consider defining V as the vertex set of such a Cayley graph. The channel set C can then be defined upon the connectivity structure provided by the generators of the Cayley graph.

An example rule on the above might be:

```
x[X(a)], x+1[X(b)] -> x[], x+1[], t(x+1)[X(a!1),X(b!1)] @ k
```

In this rule, the `X` agent at location `x` 'catches up' with its compatriot at location `x+1` and translocates to the other layer τ. If any other `X` agents are connected to the two `X` agents that match the rule above, we might either translocate them all (i.e. the whole connected closure) or simply leave them behind. In the former view, voxels are large and connected complexes travel as one unit; in the latter, voxels are small and limited in occupancy (although we don't make explicit provisions against crowding as we have previously proposed for Cartesian representations).

An advantage of such a representation is that for each colour $s \in S$ in a Cayley graph, every vertex `x` has a single s-edge leading out of `x` and a single s-edge leading in. The vertex-transitive

graph thus created has a regularity property useful for verifying its simulation behaviour [1]. A second advantage of Cayley graphs is that they allow the elegant procedural generation (as opposed to the explicit description) of V and C. One might even think of using Cayley graphs for the automatic synthesis of computationally interesting spatial projections S without an underlying biological structure based on analysis of visual experimental evidence. Finally, if the number of agents is small compared to the number of locations, a Cayley graph representation of space is much more memory-efficient since it tracks the location of agents upon the space rather than the occupancy of the individual voxels. This could be very useful in the computational simulation of sparse systems.

On the other hand Cayley graphs themselves are highly inflexible objects based on a rigorous mathematical definition, and it may prove problematic to find a group and a set of generators that suits the particular connectivity structure that a modeller wishes to enforce upon his/her model.

4.4.1 Procedural Generation of Complex Sparse Geometries

The greatest advantage of defining space as a Cayley graph is that, given a group G and a set of generators S, the simulator will automatically generate the set of voxels and channels via a well-defined mathematical procedure. Furthermore, as we have seen, the user simply needs to know S to define rules for translocation and located transformation. In this way, the user can concisely define rules on an arbitrarily large spatial landscape.

An example of a model in which such definitions would prove useful builds on the sample 'catch up' rule described earlier. In essence we can walk polymers – complexes of agents extended over multiple voxels – on a Cayley-generated spatial structure. Simple diffusion across the generator channels, translocating only the agents specified in the rule, is augmented by 'catch up' rules that pull the trailing complex to join its translocated brethren. By playing with different 'catch up' rules and their rates, we can simulate different elasticities of the polymers, and observe their progression across complex geometries.

On the other hand, we see it as a difficult task to derive the definition of space as a Cayley graph from experimental data. For example, it would be a difficult task to derive appropriate generators for neuron / axon growth or capillary structure. We believe that the aforementioned two methodologies, space as a Cartesian grid and space as a Kappa-like complex, are more amenable to such analysis at the current time.

4.5 A Provisional Operational Semantics

Given (V,C) via one of the methods described above, how do we make use of this in a Kappa simulation? We wish the operational semantics defined here to be maximally general for a

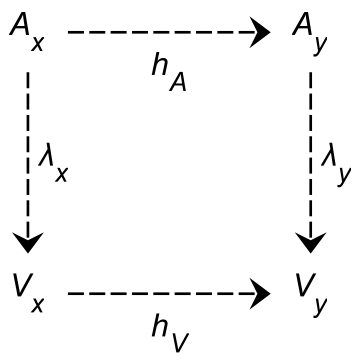


Figure 38: Decomposing a translocation rule on a set of agents A_x in a site graph x located in voxel $V_x \in V$.

given efficiency of the implementation.

A site graph x located in voxel $V_x \in V$ is a map λ from $A_x \to v$ where A_x represents the set of agents involved in x. The intuitive idea is that each agent of x is located at a voxel in V. The notion of a rule (morphism) on A_x and V_x extends as expected (Figure 38).

A rule need not necessarily specify the full connected closure of the complexes involved in its left hand side precondition. Although it is possible to define semantics in which the parts of the complex not specified in the precondition are 'left behind', we are first interested in a semantics for translocating the entire connected complex. As a prerequisite, we require that the connected closure of the complexes matched to the precondition $(m(x)*)$ does not spill over to new compartments not already specified. This ensures that when the connected component is translocated, we do not cause unintended side-effects such as dragging complexes from locations not specified in the precondition or breaking connections that spill over to other locations.

Given that we do not have an explicit definition of how we should translocate agents not directly specified in the rule precondition, we require an additional translocation map α to assign target locations to the 'surplus' nodes discovered in $m(x)*$. We can then extend Figure 38 as shown in Figure 39 to compute the new locations of the agents in the match $m(x)*$, and thus provide a working operational semantics for a generic translocation rule.

If we were interested only in translocating those agents explicitly specified in the rule (and thus possibly allow parts of the complex to be 'left behind' in the original voxel), $m(x)*$ in Figure 39 is simply $m(x)$. In this way we have a provisional operational semantics for the translocation rules of both connected and local components.

In the previous chapter, we approached Spatial Kappa from a verification viewpoint. In this,

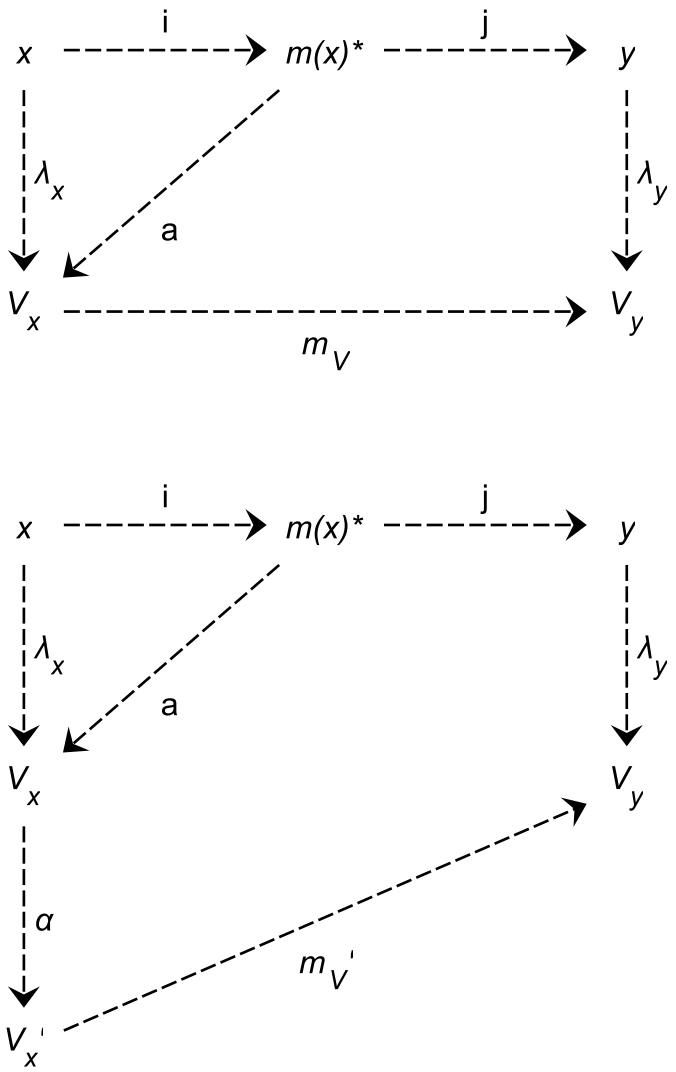


Figure 39: The operational semantics of a site graph translocation rule $\mathrm{x} \rightarrow \mathrm{y}$. From x we discern the connected closure $m(x)*$, where $ji = m$ is a match of x in y. By the prerequisite condition, $m(x)*$ must map to V_x with a given map a. Given the translocation map α, we may then determine a new mapping m'_V from V_x to V_y.

we instead approached the redefinition of Spatial Kappa from a user's point of view. We have proposed alternative methodologies for defining the spatial components of a Kappa model, along with simulation semantics and rate derivations from physical and geometric constraints. Finally, we have started to explore the new questions that arise only in the spatial dimension of rule-based modelling.

We now move on to defining and developing such models over time.

Chapter 5

Rule-Based Models of Complex Biological Webs

In this chapter we...

- ❖ Address the issues inherent to working with large-scale, complex rule-based models in a collaborative, iterative context.
- ❖ Develop the concept of a model, defined for example in standard Kappa or in Spatial Kappa, as a long-lasting, flexible, and maintainable data structure.
- ❖ Propose a means for graphically representing such a model defined in Kappa or Spatial Kappa.
- ❖ Illustrate these visualisations via a simple example of an extended kinase-phosphatase model.
- ❖ Explore the development and structure of a large-scale, complex rule-based model of the high osmolarity glycerol web in yeast.
- ❖ Identify further issues inherent to current working practices and the visualisations as specified.

In the preceding chapters we have explored the world of Spatial Kappa, particularly in relation to how we define spatial structures and how we might verify the activity of diffusing agents within them. We now return to our original set of models in Chapter 2, and to the other question posed at the beginning of our thesis: how can we ease the job of the modeller in defining, developing, and analysing a biological model (i.e. defined in Kappa) over time?

Domain experts working with the textual code of Kappa experience difficulties with the large and unwieldy nature of complex Kappa models such as the high-osmolarity glycerol (HOG) web. For example, to take advantage of the latest simulation and analysis algorithms we

may need to manually convert between obsolete GUI-capable syntax and up-to-date representation, but this adds an unwanted layer of time and effort to the modelling process. Furthermore, in modelling the HOG web we identify the need to focus further on complex interconnected and evolving 'webs' of biological information in addition to more traditional 'pathways', to better understand how biological interactions and associated experimental data may be reflected *in silico*. Improvements in tools and methods are necessary to aid and assist in the collaborative development of these large, highly interconnected biological models.

We believe that the development of graphical visualisation methodologies are an important step towards allowing domain experts to craft complex biological models without the need to worry about precise modelling (Kappa) syntax. We emphasise now, however, that the visualisations we propose in this chapter are drawn out by hand (according to the rules describe) and that we yet lack both appropriate graphical data structures and layout algorithms to automate the process.

We begin this chapter by reviewing existing methodologies for visual interfaces and graphical languages for the creation and presentation of biological models. We follow by sketching a preliminary design of graphical representations of rule-based models.

5.1 Visualising Formal Languages in Biology

Tools such as CellDesigner [35] employ visual interfaces and graphical languages for model creation, interpretation, and revision. The establishment and usage of standardised graphical notations has played an important role in facilitating the efficient and accurate transmission of biological knowledge. First and foremost amongst these notations is SBGN, the Systems Biology Graphical Notation [56], building on previous work such as Kohn's Molecular Interaction Maps (MIM) [52].

MIMs attempt to provide a comprehensive visualisation of all reactions, binding sites, and molecular conditions. However, this is only possible for limited, small networks, since inferring reaction requirements necessitates the simultaneous tracking of multiple edges in the visualisation, which can become cumbersome for complex diagrams. Although MIMs encapsulate interactions in a global context, they suffer from ambiguities in representation and lack the ability to express domain-level combinatorics that Kappa uses to reduce complexity.

SBGN defines three orthogonal and complimentary projections of the underlying complex information in a biological model: the process diagram representing all molecular processes and interactions taking place between biochemical entities and their results, the entity relationship diagram placing emphasis on the influences that entities have upon transformations rather than the transformations themselves, and the activity flow diagram displaying abstract influences between the activities displayed by the molecular entities. However, none of the three are

fully suitable for visualising rule-based models. Process diagrams suffer from information bloat and are unsuitable for the large (often infinite) reaction networks present in rule-based models. Activity flow diagrams are too simplistic to unambiguously specify the knowledge bases we are interested in. Entity relationship diagrams most closely correspond to our requirements, in that we may describe agents as entities and the rules between them as relationships, but even so are insufficient to fully capture the information contained in a Kappa model. In particular they lack an unambiguous mapping between model rules and the information displayed in the visualisation. We believe that it would be useful to have the capability to translate from our proposed visualisation to the SBGN-ER (and possibly the SBGN-AF) format for compatibility with the formalisation, but this would involve a loss of information preventing the recovery of the full model.

Efforts have also been made for the visualisation and annotation of models in Kappa's sister rule-based biological modelling language, BioNetGen (BNGL). The extended contact map representation thus chosen [12] is specialised towards displaying the purely biological features of a model, much in the same way that BioNetGen is tailored to biological models in particular rather than adopting Kappa's more generalist approach. In particular, the authors approach the problem by defining a visualisation based on biological features of interest, whereas we instead wish to derive a visualisation based on the agents and rules described in a model. Similarly, the visualisation tool RuleBender [79] links a BioNetGen contact graph with elements of the full rule set, but the visualisation itself only contains part of the information encoded in the model. In contrast, our aim is to define a formal relationship between a model and its corresponding visualisation that does not exist in the BioNetGen visualisations.

One further effort of note that visualises rule-based models is the Simmune NetworkViewer [10] where an attempt is made to simplify the network visualisation and to integrate the display of rules with their conditions without resorting to displaying model code on the visualisation (as is done in the previously described BioNetGen visualisations, for example). We do not go this far, instead believing that a modicum of Kappa syntax and pseudo-syntax in fact makes the visualisation more understandable if presented in the correct manner. We do, however, eventually wish to adapt a similar approach to their "overview first, zoom and filter, then details-on-demand" principle in which the visualisation "first generates an overview of the global network and, upon user request, presents more detailed views of local sub-networks and the underlying reaction rules for selected interactions" [10]. We believe that this approach, combined with modular rule-based model development methodologies and iterative revisions on the model code, will help to overcome the issue that purely visual approaches to model creation do not scale well with model size.

Our proposal below is also closely related to the idea of executable biological model visualisations, in which visualisations are not only used to provide a standardised means of document-

ing and communicating information but are also utilised as artefacts of the model development process, including automatic export of changes made to the visualisation to mathematical models. An early example of such work is [39], in which high-level visual formalisms are used to tie together decompositions of biological systems, which are then reassembled and simulated concurrently to form comprehensive models. We attempt to gain a similar level of compositionality through the use of Kappa semantics, focusing on how best to display this to the user and how to allow them to modify the information contained within.

We also borrow ideas from a recent 'second-generation' approach [58] to support high-level modelling (for example in Kappa) through the structured programming features of Python. In particular, we similarly attempt to provide support for multiple modelling languages through the concept of 'views' on an underlying data structure. We share the need for programmatic methods to support the development process as biological models and their modelled interactions become more realistic and complex (as shown in Chapter 2), and identify in a similar manner the advantages of programmatic abstractions when models must be shared and collaboratively modified. We differ, however, in that we still wish to present a non-programmatic interface for the development and iterative modification of biological models, which is where the visualisations presented to follow are hoped to come in useful.

5.2 Visualising Complex Models in Kappa

The solution we thus propose is to extend the basic static contact graph of Kappa with richer model knowledge and annotated dynamic information, by mapping the set of rules encoded in the model upon the agents displayed in the contact graph. We work with rules rather than reactions, unlike MIMs, and explicitly associate edges with rule effects.

We follow a two-fold process in visualising this information. Initially we derive a simple contact graph, modified to include agent states and compartments. Then we encode the set of rules in the model upon the elements of a spatially-extended contact map. The eventual aim of the visualisation is to be simple to understand and modify, but also to contain the comprehensive detail allowing for precise and unambiguous recovery of the structure (agents, tokens, compartments, channels, rules, variables, perturbations, and initial conditions) of a corresponding knowledge base from any modifications made.

5.2.1 Entities

Our basis for visualising models is a Kappa contact graph (as described in Chapter 1) modified to represent the Compartments present in the model, their initial populations, the Channels within and between them, and an explicit representation of the States available for each Site (Figure 40).

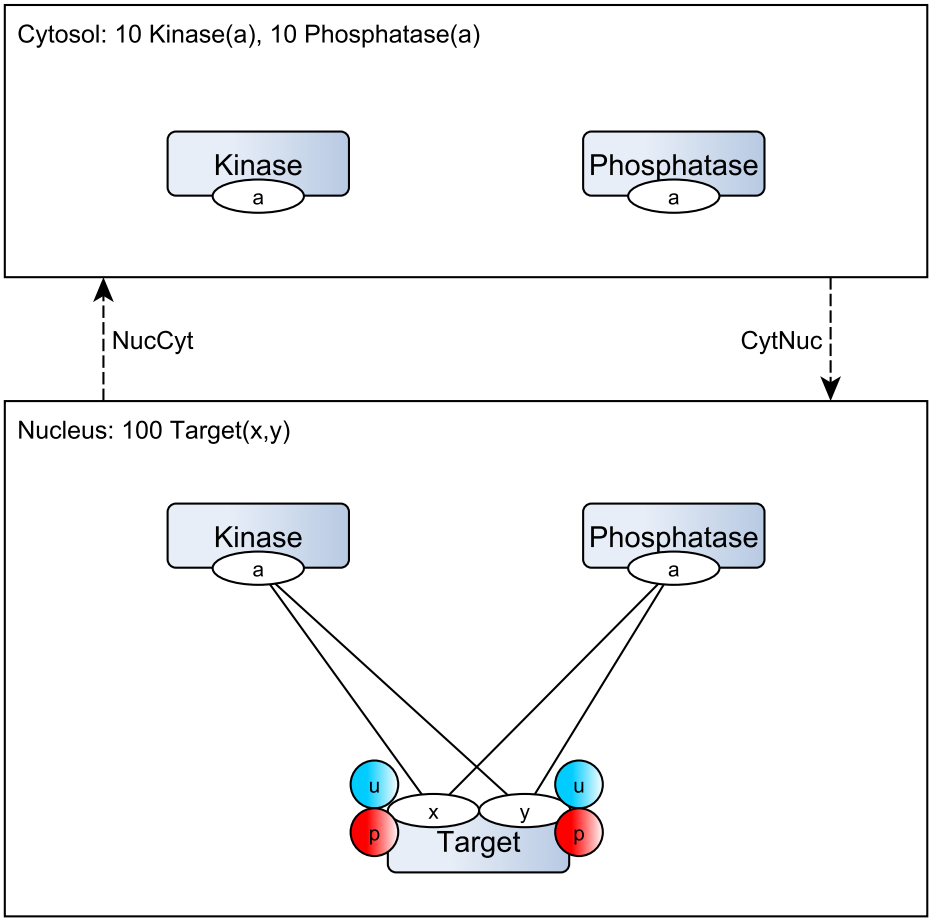


Figure 40: A contact graph for the extended kinase-phosphatase model, enhanced to display Compartments, Channels, and States alongside the usual Agents and Sites. The Compartments shown are single-voxel Compartments, and thus we have no need to explicitly state their structure (similar to the fact that no extra definition is needed in the actual Spatial Kappa model). We also indicate the initial populations of each Compartment. The links between Agents still represent how their Sites may bind. In this particular model the `Target` is unable to escape the nucleus, whereas the `Kinase` and `Phosphatase` start in the cytosol and may translocate back and forth via the Channels provided.

A Compartment is represented by an enclosed area, labelled with its name (and if necessary its explicit voxel structure), followed by a list of number-complex pairs representing the initial population. Any other attributes such as compartment or voxel volume are also written in this label, in a form similar to how the `%compartment` definition is represented in the textual model.

Channels may be either intra-compartmental channels from voxel(s) to voxel(s) within a Compartment, or inter-compartmental channels between two Compartments. We represent the former with a dashed arrow looping back upon the parent Compartment, and the latter with a dashed arrow between source and target Compartments indicating the direction of the Channel. We label the arrow with the Channel name and any attributes that may be present in the `%channel` definition.

We then populate the Compartment with the Agents and Tokens that may be present in interactions that take place there, as we would a standard contact graph. The only difference in our representation of Agents is that we explicitly represent all spatially plausible States in our enhanced visualisation, whereas in a traditional contact graph we would simply colour Sites with modifiable States. For a given Site, if its States are immutable (i.e. not changed by any Modification rule within the model) then we colour them yellow; if they are mutable, then we colour them in red, with the default state (as determined by their order in the agent declaration of the model) in light blue instead. Tokens are shown as smaller labelled circles, marked with any Perturbations upon them.

In this way, the static elements of a model (entities and initial conditions) form a tree-like branching hierarchy with Compartments at the top and States at the bottom. In the next section, we see how these are then linked together by relations that encapsulate the dynamic elements of a model: rules and perturbations.

The positional relationship between Compartments must be inferred from the Channels between them, but this may be difficult in a complex multi-dimensional structure such as those theoretically supported by Spatial Kappa. On the other hand, such structures are difficult enough to define manually that we believe they would be catered for in a complete implementation by a syntactic abstraction of voxels (i.e. the Sphere and Spine abstractions in the current implementation of Spatial Kappa), represented in this visualisation as a structure attribute of a single Compartment. Thus there should rarely if ever arise the need to visually represent such complex structures. For now we leave this as an unresolved question, requiring us to revisit it at a later stage should it be necessary.

5.2.2 Rules and Rule Effects

Rules are named combinations of six distinct rule effects (Creation, Destruction, Binding, Unbinding, Modification, and Translocation) on a certain precondition, occurring at a given stochastic rate. With the exception of Translocation effects, these are equivalent to the notion of

atomic rules comprising a composite rule in the Scala implementation framework LMS-Kappa by Ricardo Honorato-Zimmer (unpublished work). We add Translocation effects to define rules upon the spatial aspects of a model.

We map these effects to the entity graph described above, labelling each effect with the rule name, precondition, and rate. A single Rule may thus be represented multiple times on the same visualisation, once for each of its effects; the combination of all these effects upon the precondition is the sum of the Rule's actions.

A consequence of this is that choosing a long rule label will make the visualisation difficult to read. For this thesis we thus choose short rule labels to compensate; in many of our examples, we omit them altogether where no confusion is caused (i.e. for simple rules with only one effect). In an actual implementation, we might envision mapping the long rule label to a shortened variant in order to represent it on the visualisation.

The preconditions to a Rule are recorded in a syntactic abstraction similar to that outlined by Michael Pedersen [67], in which large rules with many bindings are simplified such that the agent brackets only specify modification sites. Take for example the following complex Kappa rule:

```
A(b~p!1),B(a!1,c!2),C(b!2,d),D(d~p) ->
            A(b~p!1),B(a!1,c!2),C(b!2,d!3),D(d~p!3)
```

The rule expresses the creation of a link between `C` in a certain complex to a free phosphorylated `D`, but this effect is not immediately apparent in the verbose code due to the elaborate specification of binding sites and bond labels in the precondition. Our solution is to classify this as a Rule with a Binding effect between `C(d)` and `D(d p)` and to simplify the precondition as follows:

```
A(b~p)-B-C, D(d~p)
```

Essentially we remove as many numbered links as possible in the precondition complex and replace them with hyphens. This annotation allows us to get away without writing bulky and incomprehensible preconditions whilst still preserving Kappa's "don't write don't care" policy, via trivially automated deduction of appropriate binding sites. In particular, we do this where there is no ambiguity as to how two agents may be bound. An alternate representation of this, where some ambiguity might exist, is as follows:

```
A:b~p!a:B:c!b:C, D:d~p
```

In this representation, we explicitly associate the site at which the agent Binding occurs. For the rest of this thesis we use both *implicit* and *explicit* notations interchangeably, except where we are unable to simplify the precondition to such a form and are forced to write out the

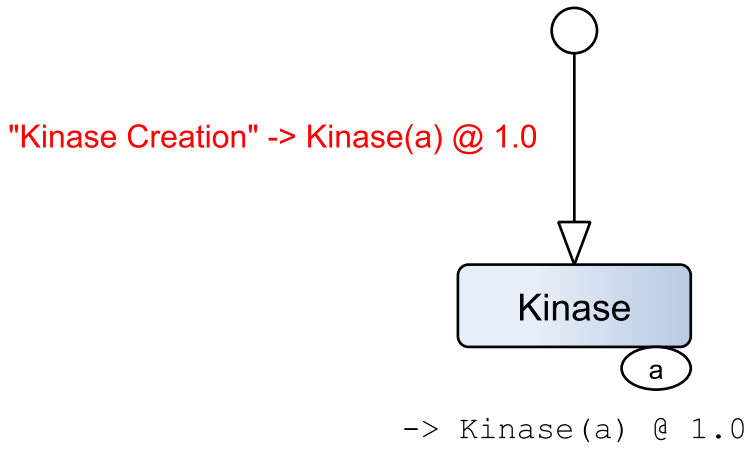


Figure 41: Creation of a new `Kinase(a)` Agent.

precondition explicitly (for example, when the precondition is branching or circular rather than linear, an inherent weakness to such notations that we do not address any further for now).

Bidirectional Rules – rules with both forwards and backwards effects – are simply treated as two unidirectional rules with opposite effects, each with their own precondition and rate.

The Kappa notation for ambiguous molecularity, which allows for modellers to distinguish between contexts in which seemingly disconnected agents (i.e. a presumed binary complex) in the precondition may actually be connected (i.e. are actually unary), is also treated as a separate instance of the same effect and precondition. We separate the Rule into the unary and binary cases:

```
A(b),B(a) -> A(b!1),B(a!1) @ 'k2' ('k1')
```

thus becomes two binding rules with preconditions and rates `A,B @ 'k2'` and `A-B @ 'k1'` respectively. Our notation for the second case may safely leave the binding between `A` and `B` unspecified.

Tokens may be destroyed and created by each application of a Rule. Each Rule must reference the Tokens involved. Unfortunately, there is no easy way of denoting the number of Tokens associated at each stage (precondition and postcondition) of a Rule, hence we take the solution of forcing each Rule to keep track of the appropriate number of references to Tokens both created and destroyed.

We now visit each rule effect and its representation upon our visualisation in turn.

We visualise the base Creation rule effect in Figure 41. The rate of the rule is represented on the link, as is the created agent explicitly (allowing us to unambiguously declare, for example, the State of its Sites or any complex binding). Any preconditions are annotated on the link

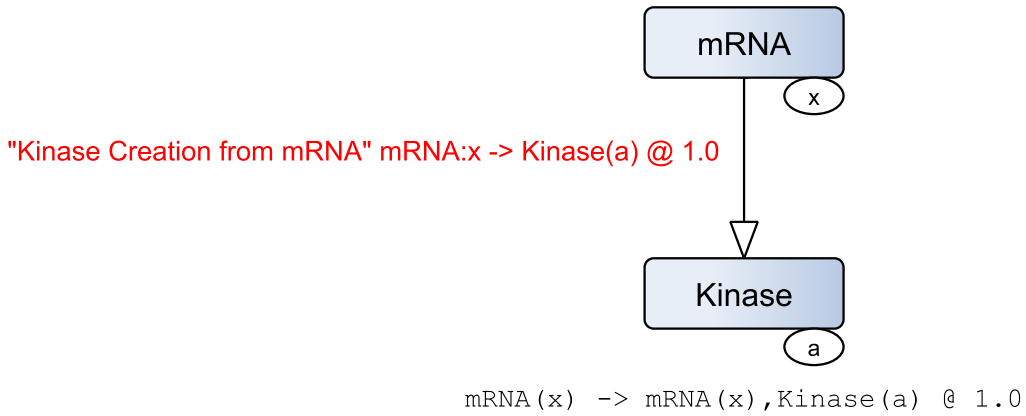


Figure 42: Creation of a new `Kinase(a)` Agent preconditioned on the presence of unbound `mRNA`.

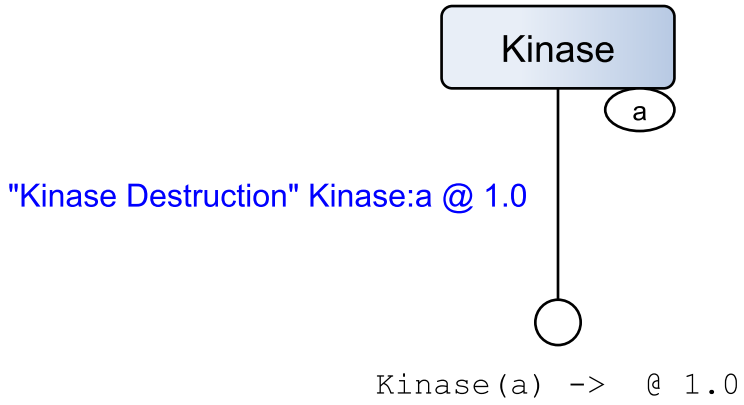


Figure 43: Destruction of a `Kinase`, preconditioned on the fact that the `a` site is unbound.

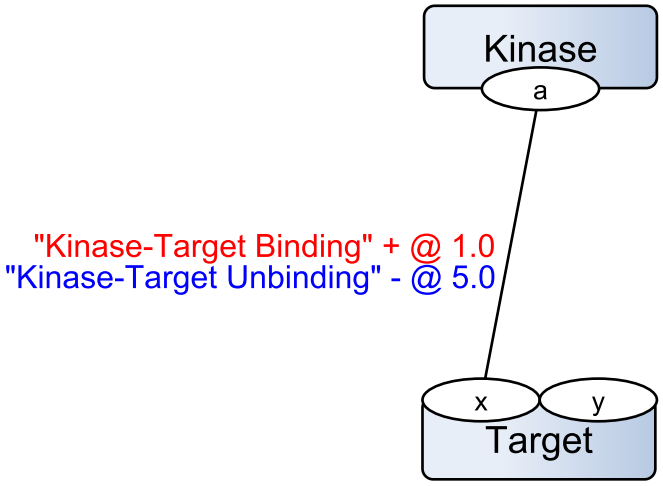
shown as in Figure 42 – for a Creation effect with a precondition, we also change the link from an empty node to the first Agent in the precondition.

We visualise the base Destruction rule effect in Figure 43. Again, should the rule have a precondition (in this case that the Kinase Site is unbound), this is annotated on the link.

Binding and Unbinding effects are shown in Figure 44. Binding rules are shown in red with a positive prefix, and unbinding rules in blue with a negative prefix.

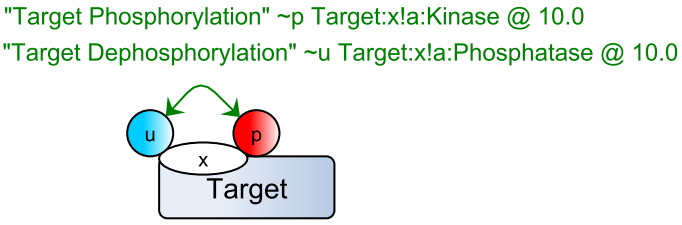
Modification rules control the change of State within a Site (Figure 45). We indicate the direction of the modification with an arrow between the States, and annotate this link with directionality, precondition, and rate.

Finally, Translocation rules are annotated upon the Channels where they occur, as shown in Figure 46.



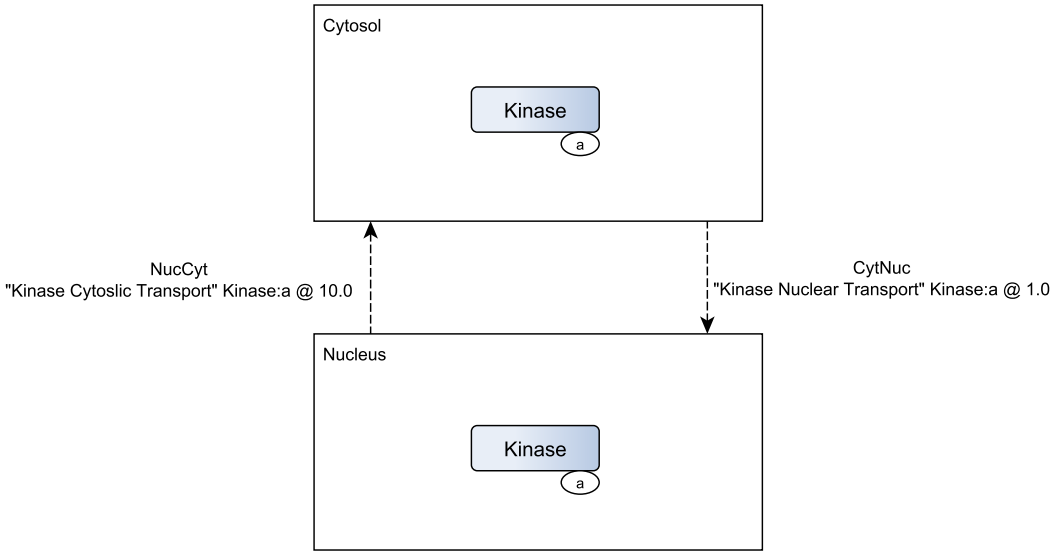

```
Kinase(a),Target(x) -> Kinase(a!1),Target(x!1) @ 1.0
Kinase(a!1),Target(x!1) -> Kinase(a),Target(x) @ 5.0
```

Figure 44: Agents interacting (binding and unbinding) with each other.



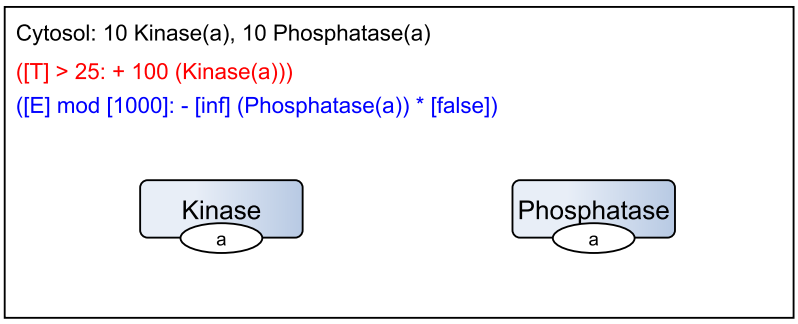

```
     Target(x!1~u),Kinase(a!1) -> Target(x!1~p),Kinase(a!1) @ 10.0
Target(x!1~p),Phosphatase(a!1) -> Target(x!1~u),Phosphatase(a!1) @ 10.0
```

Figure 45: Modifying Agent States. Note the notations on the edge that indicate the outcome of the modification (i.e. in which direction the modification occurs). Note also the explicit declaration of Site-Site binding in the preconditions.



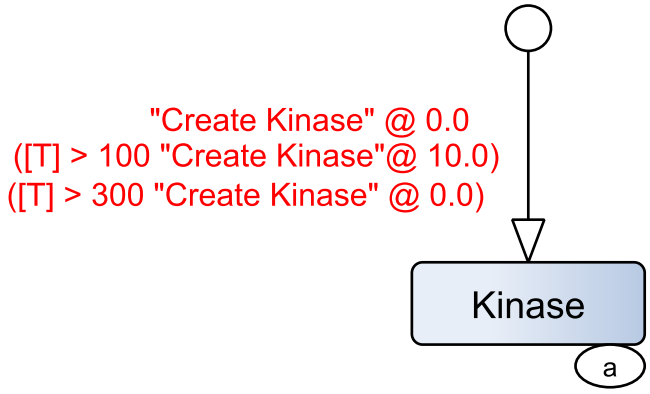

```
Kinase(a) ->:cytnuc Kinase(a) @ 1.0
Kinase(a) ->:nuccyt Kinase(a) @ 10.0
```

Figure 46: Translocating Agents across Compartments via Channels.



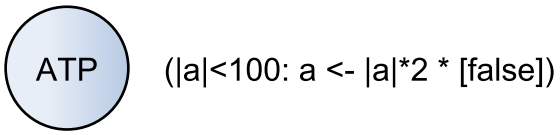

```
                [T]>25 do $ADD 100 Kinase(a) :Cytosol
repeat ([E] [mod] 1000)=0 do $DEL [inf] Phosphatase(a) :Cytosol until [false]
```

Figure 47: Visualising $ADD and $DEL Perturbations upon Agents within a Compartment.




```
[T]>100 do $UPDATE 'Create Kinase' 10.0
[T]>300 do $UPDATE 'Create Kinase' 0.0
```

Figure 48: Visualising $UPDATE Perturbations upon named Rules.




```
repeat (|a|<100 do a <- |a|*2) until [false]
```

Figure 49: Visualising a Perturbation that resets the value of a Token.

We consider the three main types of dynamic *Perturbation*s upon a Kappa model: perturbations that induce a change in rate of pre-existing Rules (`$UPDATE`), perturbations that reset the value of a Token (`t < - x`), and perturbations that add (`$ADD`) and delete (`$DEL`) specified Agents, both according to predetermined preconditions. This allows us to associate perturbations to Rules, Tokens, and Compartments respectively (for example, an `$ADD` or `$DELETE` perturbation associates with the corresponding Compartment in which the Agents are created or destroyed).

Perturbations upon a model are visualised as parenthesised addendums to the appropriate entity or rule, detailing the condition and effect of the perturbation. The perturbation effect is demarcated from its condition by a colon. We represent the repeat-until construct after the perturbation effect by a Kleene star, followed by the termination condition.

The first type of Perturbation, adding and deleting agents from Compartments within the reaction mixture, is represented in the visualisation as a parenthesised addendum to the initial conditions of the Compartment describing the conditions at which the changes are made. For example, in Figure 47 we show a Perturbation adding 100 `Kinase` agents to the `Cytosol` Compartment at simulation time `T > 25`, followed by a Perturbation that acts every `E = 1000` simulation events to remove all `Phosphatase`s from the `Cytosol`.

The second type of Perturbation, altering the rates of existing rules, is represented in the visualisation as a parenthesised addendum to the Rule in question. In Figure 48 we show updates upon a constitutively inactive Rule (consisting of a single Creation rule effect) to switch it on at simulation time `T > 100` at rate 10.0, and to switch it off again at simulation time `T > 300`.

The third type of Perturbation, modifying the concentration of a Token, is represented in the visualisation as a parenthesised addendum to the Token in question. In Figure 49 we show a Perturbation that will double the value of `ATP` Tokens within the model each time it falls below a certain threshold `|a| = 100`.

5.2.3 Variables

Models may define entity and rule attributes as *Variable*s and functions upon said Variables (for example taking into account bio-time or token concentration). Variables embedded in the visualisation may for example be used to derive stochastic rates or initial copy numbers. We allow the variable labels to be displayed in the model visualisation proper, and associate the variables to concrete values in a separate frame. A user of the visualisation may achieve the same degree of control he might have in the textual model simply by editing these values.

In future iterations of these visualisations we can envision expanding this section with further information regarding the model, such as references to pathway databases or publications.

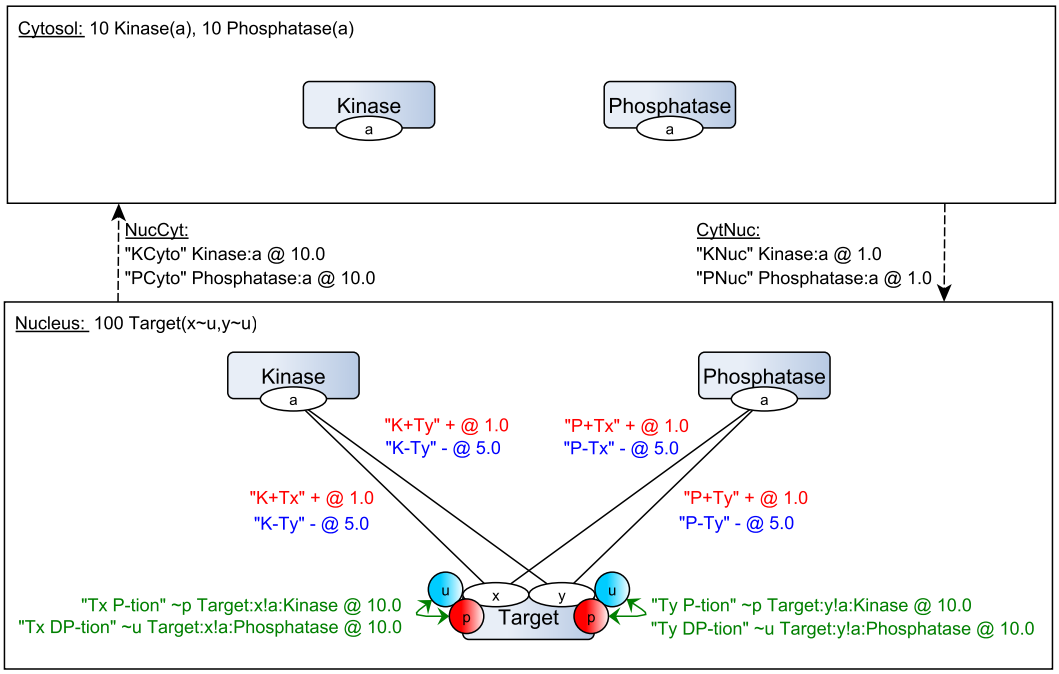


Figure 50: Visualising all entities and rules of an extended Kinase-Phosphatase model with two Compartments and two Channels. The model consists of 16 single-Effect rules. Four of these govern Translocation of the `Kinase` and `Phosphatase` between the Compartments, four each control Binding and Unbinding from the `Target`, and the final four modify the `Target`'s phosphorylation States.

5.2.4 Example: Extended Kinase-Phosphatase

An example visualisation is shown in Figure 50, combining model features including entities and rules but excluding perturbations and variables. The model shown is a variant of the Kinase-Phosphatase model introduced in Chapter 1, extended with two Compartments and two Channels between them. We take the existing Kinase-Phosphatase model as occurring in the `Nucleus` of a two Compartment model with each Compartment (`Cytosol` and `Nucleus`) consisting of a single voxel of unspecified volume each. We then provide Translocation Rules for diffusion of the `Kinase` and `Phosphatase` in and out of the `Nucleus`.

The model consists of 16 Rules composed of a single Effect each. Four Rules (labelled '`KCyto`', '`PCyto`', '`KNuc`', and '`PNuc`' respectively) govern the Translocation of `Kinase` and `Phosphatase` Agents between the two Compartments `Cytosol` and `Nucleus`. In the `Nucleus` the `Kinase` and `Phosphatase` may both Bind and Unbind from the two Sites of the `Target`; this is described in another eight rules. A final set of four rules allow the `Kinase` and `Phosphatase` to modify the State of the `Target` Site to which they are bound. Figure 51 shows a sample trajectory for the model, and the stochastic equilibrium thus achieved.

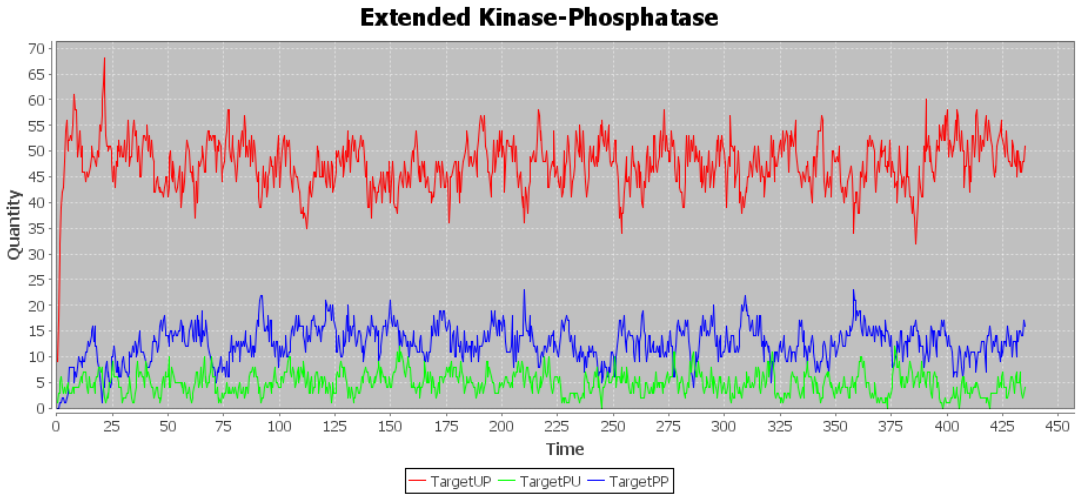


Figure 51: A sample stochastic trajectory for the extended kinase-phosphatase model in Spatial Kappa. Similar to its simple counterpart (Chapter 1), the model very swiftly achieves a stochastic equilibrium in which roughly half of the total `Target` population is fully phosphorylated. Simulation conducted in Spatial Kappa tool version 2.1.1.

Note that we use the explicit notation for depicting rule preconditions as introduced above. For each phosphorylation or de-phosphorylation rule, we must resolve the ambiguity at which site the `Target-Kinase` or `Target-Phosphatase` binding occurs for us to reconstruct the model from the visualisation. Were we to use the implicit notation, it would not be clear to which site the `Kinase` or `Phosphatase` should be bound to effect the modification.

5.3 Use Case: The High-Osmolarity Glycerol Web in Yeast

To explore these visualisations in the context of large complex models, we revisit the most complex model introduced in Chapter 2: the high-osmolarity glycerol web in yeast. We begin by highlighting the fact that this model was created via an exhaustive literature curation methodology, and displays a large number of rule refinements based on minor hypothetical differentiations in binding and phosphorylation states of the preconditions (which are only exacerbated by the number of such states possible). Although the original modeller has made an effort to limit these to the core areas of interest (i.e. the `SHO1` interactions) at the expense of clarity in others (i.e. the use of dummy agents in the feedback process), we still see the effects of this decision upon the model in the visualisations below.

The contact map of the overall model is shown in Figure 52, but for purposes of discussion we split the rules of the model into eight separate chapters to better parse the interconnected web of interactions between the agents involved. Again, to preserve space and readability, we keep rule labelling on the visualisation to a minimum. We also utilise the explicit notation of

rule preconditions so as to eliminate unambiguity; although there is no more than one direct binding path between any two agents in the model, there are multiple indirect paths that may be the cause of confusion (for example, PBS2 and HOG1 may be bound either directly or via the PTC1 or PTC23 phosphatase complexes).

Figure 53 displays the abstract overview of the high osmolarity glycerol web, in particular the relationships between the eight chapters. In rough order corresponding to their response to osmotic stress, they are as follows: the osmosensing system (Figure 54), the SLN1 pathway (Figure 55), the SHO1 response (Figure 56) and activity (Figure 57), the activity of CDC42 (Figure 58), the STE11-STE50 dimer (Figure 59), and OPY2 (Figure 60) respectively, the MAPK cascade proper (Figure 61), and the proteomic and genetic feedback response (Figure 62).

In the following subsections, we take a closer look at the chapters in turn, including how their visualisations (as defined above) fit into the overall model scheme. We hope that graphical representation of the rules make the model ease their understanding.

5.3.1 Osmosensing

The osmosensing component of the model represents the mechanisms of osmotic shock upon the virtual yeast cell - in particular, the detection and absorption of osmolytes (Figure 54). The key agent in this chapter, Osm, is capable of existing either internal to or external of the cell via the '∼i' and '∼e' states of its localisation site. It also has a single theoretical binding site, which it uses to interact with the membrane components of the SLN1 and SHO1 pathways.

When external of the cell, the Osm agent may bind to the sensory domain of the SLN1 protein, or to HKR1 or MSB2 (both upstream agents in the SHO1 pathway). This deactivates the first and activates both of the others, thus initiating signal propagation.

When internal to the cell, the Osm agent may again bind to the above membrane proteins. Its action now is opposite to the above: it may activate the sensory domain of SLN1 or deactivate HKR1 / MSB2, thus terminating the downstream signal.

Finally, internal Osm may also bind to the FPS1 channel protein. This is a representation of an active export mechanism, which in reality is predicted to control the influx and efflux of osmolytes to encourage internal accumulation. In the model, we use FPS1 to destroy any excess internal Osm agents, indicating that the cell has achieved an osmotic equilibrium.

The Osmosensing chapter thus represents competitive binding between internal and external osmolytes on the three different categories of sensor and the two underlying pathways. When external osmotic pressure is applied, the cell swells (as represented by a build-up of internal osmolytes) and efflux of osmolytes (via the FPS1 mechanism) occurs. We note that this representation (explicit binding of osmolytes to the membrane sensors of the cell) is unlikely to correspond to the exact mechanism existing in a yeast cell *in vivo*. In particular, it fails to duplicate experimental results in which a subsequent osmotic shock reactivates the pathway,

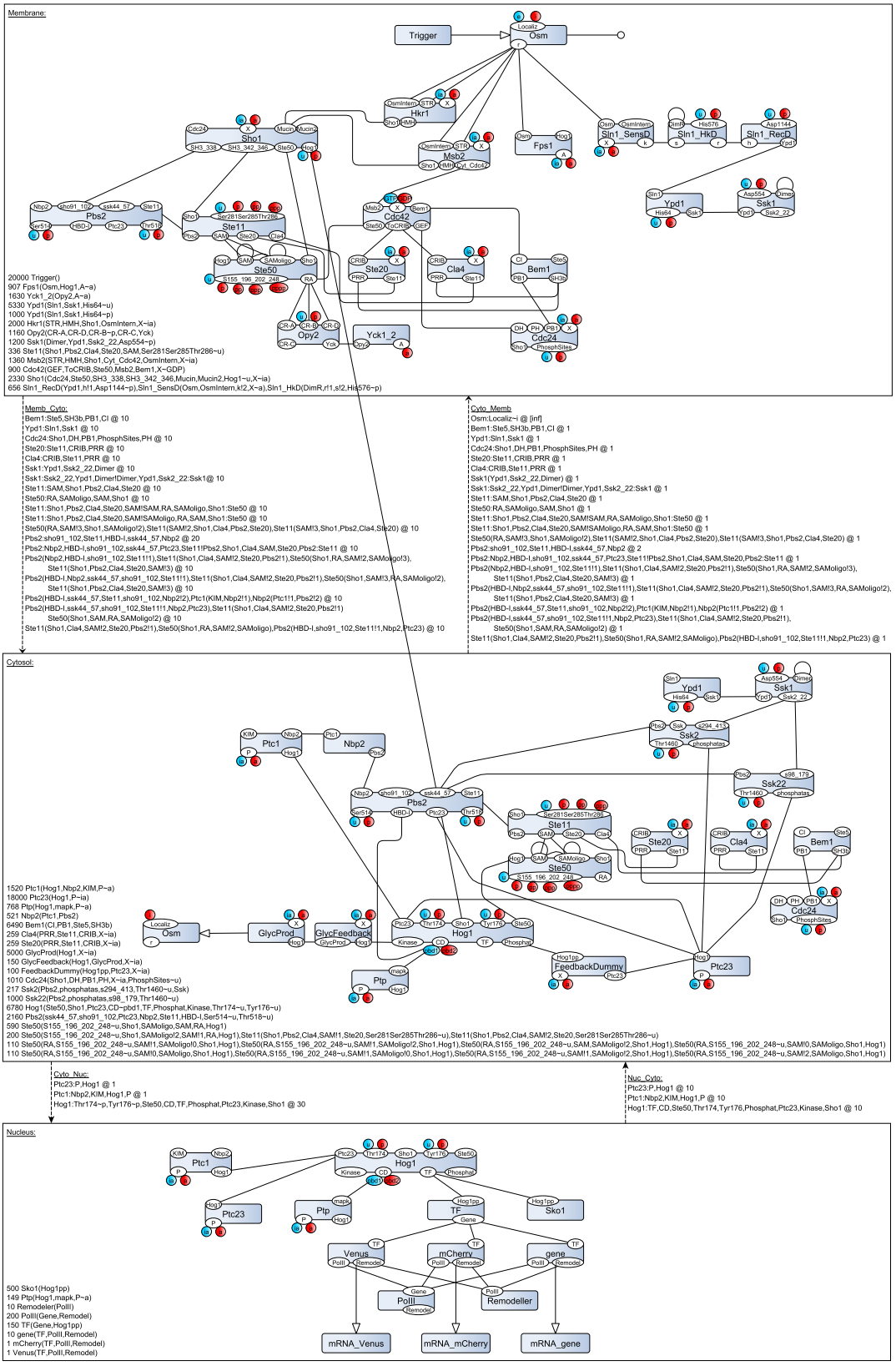


Figure 52: The extended contact map of the high osmolarity glycerol web in yeast showing spatial structures (compartments and channels) as well as initial conditions. Note also the presence of a inter-compartmental link indicating the possible binding of cytosolic `HOG1` with membrane `SHO1`.

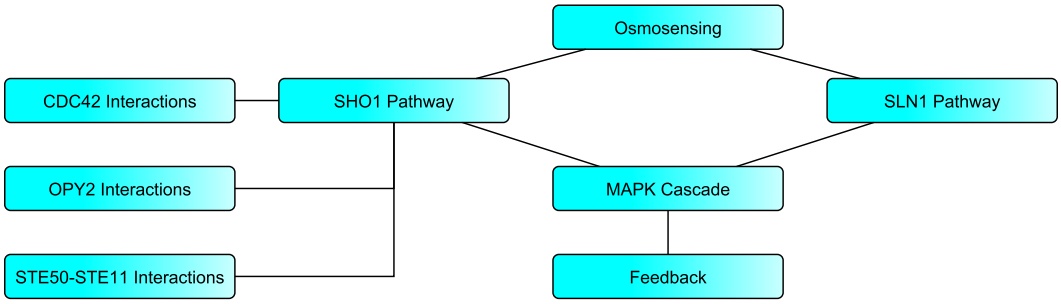


Figure 53: An abstract overview of the high osmolarity glycerol web in yeast, showing the nine chapters of the Kappa model and their interrelation with one another. The Osmosensing module is responsible for detecting the signal for osmotic shock, and passing it on to the fast response (the `SLN1` Pathway) and the slow response (the `SHO1` Pathway). The slow response is mediated by CDC42, OPY2, and STE50-STE11 Interactions, but eventually both feed into the MAPK Cascade and the Feedback mechanisms.

due to the lack of a 'reset' button to clear the effects of the first. Instead, the system builds up internal osmolyte such that the available sensor proteins are mostly bound and the effects of a subsequent shock do not propagate.

We propose an alternative modelling mechanism in the form of a 'perfectly adaptive' sensor as adapted from previous unpublished exercises by Vincent Danos and Walter Fontana, based on methylation states in bacterial chemotactic sensors. In such a sensor, newly injected `Trigger` agents capture inactive receptors and remove them from the equilibrium between external and internal osmolytes, thus rebalancing the system in favour of signal propagation. Feedback from the `HOG` response in turn deactivates the `Trigger`, feeding captured receptors back into the system and driving it once more towards a new equilibrium.

We test this alternative both in isolation (with highly simplified feedback) and observe that in both cases the model remains competent in the face of a second osmotic stimulation. In particular it matches further experimental results [51], in that if we immediately apply a second osmotic shock to follow a first, the response profile of the system resembles that of a single larger shock. Again, we note that we have no evidence that this alternative representation actually exists within the yeast cell. However, it does reproduce experimental behaviour in a way that the naive approach was unable to.

Filtering the effect from in conjunction with the full high-osmolarity glycerol web proves to be more difficult, especially given that the increased number of rapid binding and unbinding interactions at the head of the pathway makes it time-consuming and difficult to obtain time series results. We are able to once again observe a qualitative effect on downstream output similar to the case in isolation, which suggests that the 'perfectly adaptive' sensor is a better fit to observed data than the competitive binding sensor. But we are at the same time confronted by the

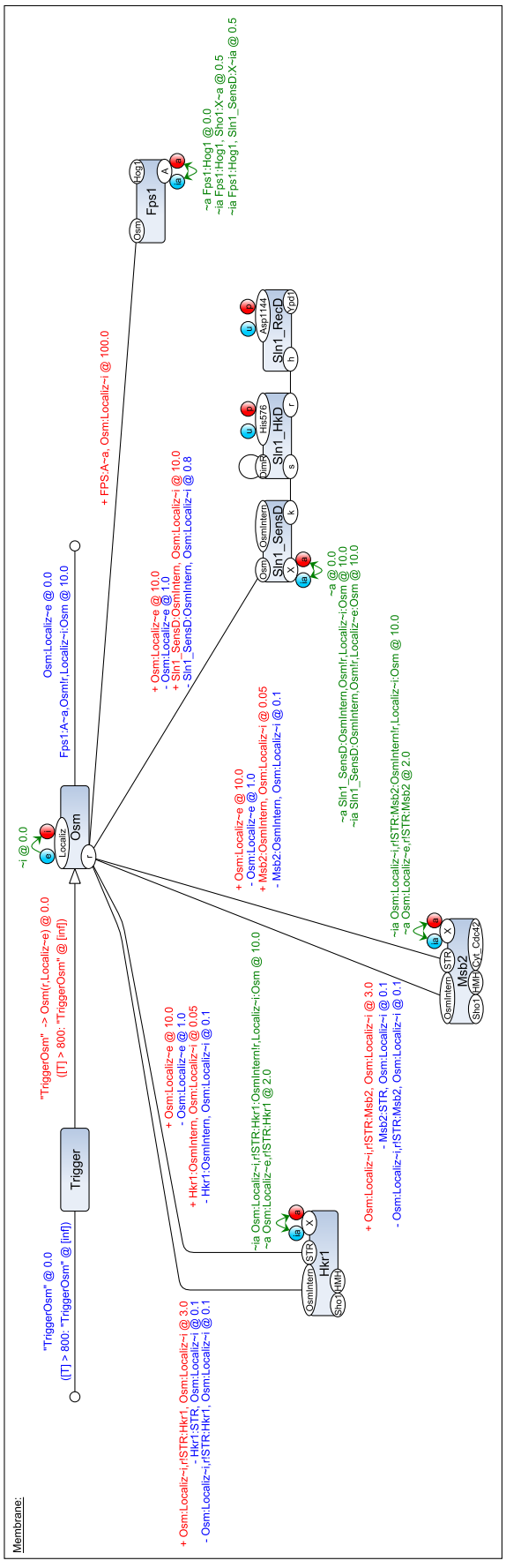


Figure 54: Osmotic sensing module of the high osmolarity glycerol web in yeast, showing external osmotic shock as well as the balancing effect of the internal osmotic response. Note also the perturbation upon the `Trigger` agent that creates external `Osm`.

difficulty of quantitatively observing refinements to a large-scale complex web of interactions

5.3.2 SLN1 Phosphorelay

The two-component SLN1 phosphorelay (Figure 55) is the fast-acting pathway of the high-osmolarity glycerol response in yeast. We choose to depict the SLN1 protein as an immutable (i.e. there are no rules that destroy the bindings present in the initial conditions) complex of three agents according to function: the sensory domain SLN1_SensD, the histidine kinase (transmitter) domain SLN1_HkD, and the response regulatory (receiver) domain SLN1_RecD. This choice clearly delineates the functionality of the three domains by modelling them each as separate agents; of course, should we wish not to emphasise this so much, it is an inconsequential matter to model the SLN1 protein as a single agent with multiple sites instead, much as we do with the PBS2 and HOG1 agents described later.

The phosphorelay involves the sequential transfer of a phosphate group (the presence of which is represented in the model as a pair of states, unphosphorylated '∼u' and phosphorylated '∼p') from the transmitter domain to the receiver domain. This occurs between two SLN1 complexes (meaning that they must form a dimer on their 'DimR' sites before signal propagation) rather than intramolecularly.

The phosphate group is then further transferred to the downstream YPD1 and SSK1 agents to complete the phosphorelay. Finally, the latter translocates to the cytosolic compartment, where it then binds with either SSK2 or SSK22 in preparation to activate the MAPK cascade.

We see the benefits of the visualisation methodology in how we can follow the phosphorylation events from the head of the SLN1 phosphorelay to its tail. On the other hand, we note that the explicit description of binding and phosphorylation events in the visualisation makes it more complicated than it otherwise might be, a theme repeated in the chapters to follow.

5.3.3 SHO1 Response

The SHO1 pathway is the slower of the two osmotic responses in yeast, and its activity is not as easily categorised as the two-component SLN1 phosphorelay. The SHO1 protein at the core of this pathway is a complex osmosensor that responds to heat and oxidative stress in addition to osmotic stress, and is also thought to be involved in regulation of the mating and pseudohyphal growth pathways as well. The reason for its slow and long-lasting response is thus largely due to this unwieldy and highly interconnected complex structure, which also involves components of the three chapters that follow this one (the activity of CDC42, STE11 / STE50, and OPY2 / YCK1_2).

We consider this response in two stages due to the complexity of the interactions surrounding STE11 in particular. We begin with the upstream component of the response in Figure 56.

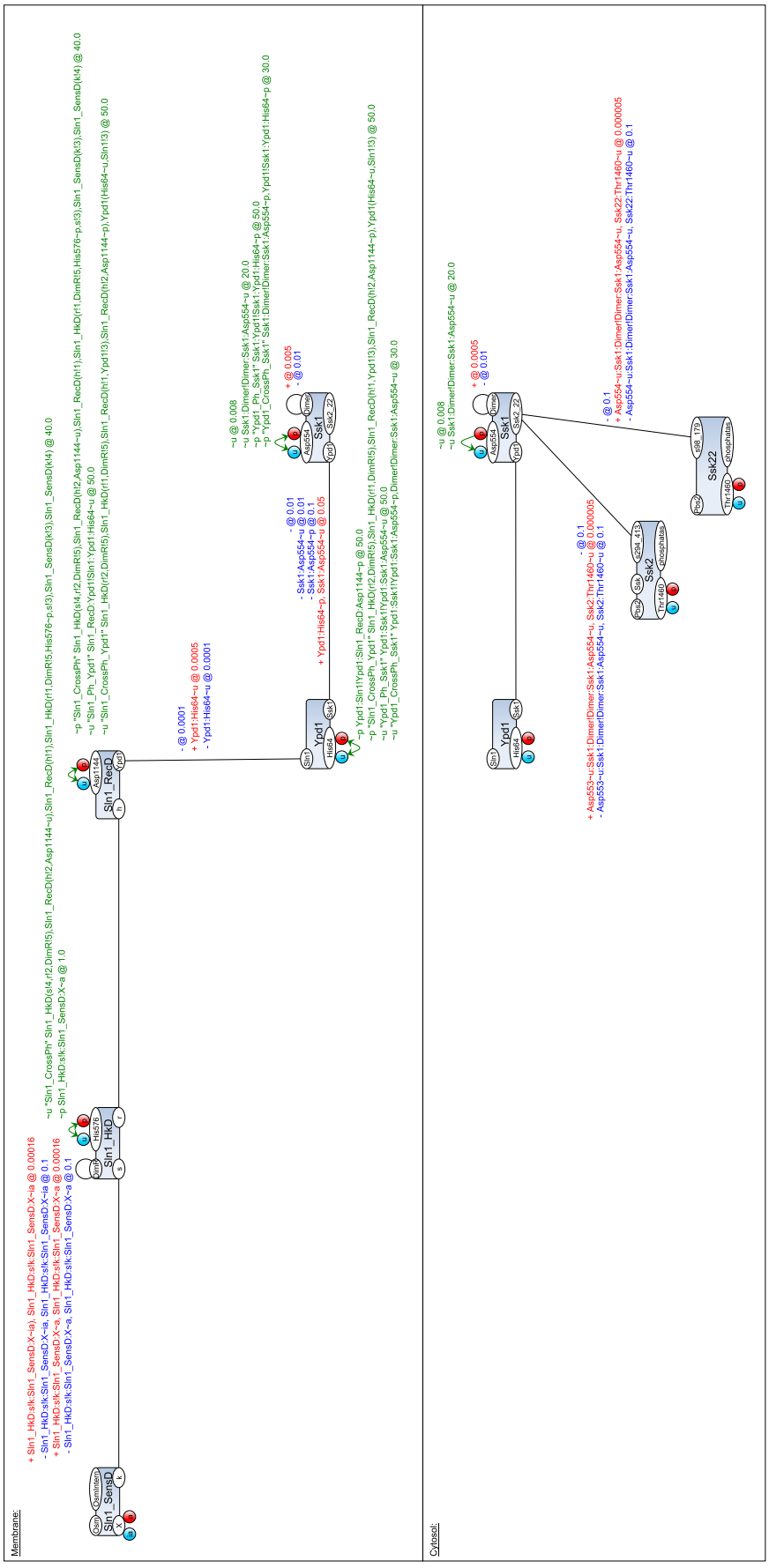


Figure 55: The two-component SLN1 phosphorelay of the high osmolarity glycerol web in yeast. The signal flows from SLN1 to SSK1 via YPD1, then travels to the cytosol where it acts upon the MAPKKKs SSK2 and SSK22.

Here we see the previously introduced HKR1 and MSB2 sensor proteins transfer their phosphorylation signal to SHO1. We also see the possibility of trans-compartmental binding between doubly-phosphorylated (fully active) HOG1 in the cytosol and SHO1 in the membrane, in which the downstream MAPK may (at a low rate) interact with SHO1 as one of the pathway's feedback mechanisms.

In this upstream component we also consider the complex interplay between STE11, STE20, CLA4, BEM1, and CDC24 in both membrane and cytosol. STE20 and CLA4 act to triply phosphorylate STE11.

STE11 is also the key participant in the downstream component of the membrane SHO1 response (Figure 57), where it binds to PBS2 with the scaffolding assistance of SHO1, mediated by STE50.

We see here an example of how even the systematic modularisation of a large complex model can be insufficient to decompose it into easily visualised and understood chapters, especially when the model involves multiple compartments (and an unclear delineation of activity between them) and complicated rule preconditions. We attempt to solve this here by splitting it further, but this is a stop-gap measure at best; in Chapter 7 we take a look at more permanent solutions, including multi-scale visualisations based on the "overview first, zoom and filter, then details-on-demand" approach mentioned above.

5.3.4 CDC42 Activity

The chapter on CDC42 activity is the first of three chapters that support the SHO1 response to osmotic stress and are also involved in various other yeast pathways including the pheromone response. As shown in Figure 58, CDC42 binds with many important players in the upstream SHO1 pathway. In particular it activates the STE20 and CLA4 kinases, and provides a means of triply phosphorylating STE11 (via STE50) in preparation for propagating the signal to PBS2.

5.3.5 STE11 and STE50 Activity

The second of the three chapters supporting the SHO1 response to osmotic stress, the multiple binding configurations of STE11 and STE50 play a major role in regulating the activity of STE11. Many of the rules in this chapter (Figure 59) concern the cytosolic oligomerisation of STE50, in which one STE50 agent may bind with another on a combination of its 'SAM' and 'SAMoligo' sites. The remainder concern the binding of STE11 with STE50, mediated in the membrane by OPY2.

We note that a large proportion of the complexity in this chapter occurs because the SAM and SAMoligo sites have the same activity but must be specified separately in the rules. We also note that although the large number of refinements on the base binding rules (an artefact of making a literal translation from the previous version of Kappa to the current) make the

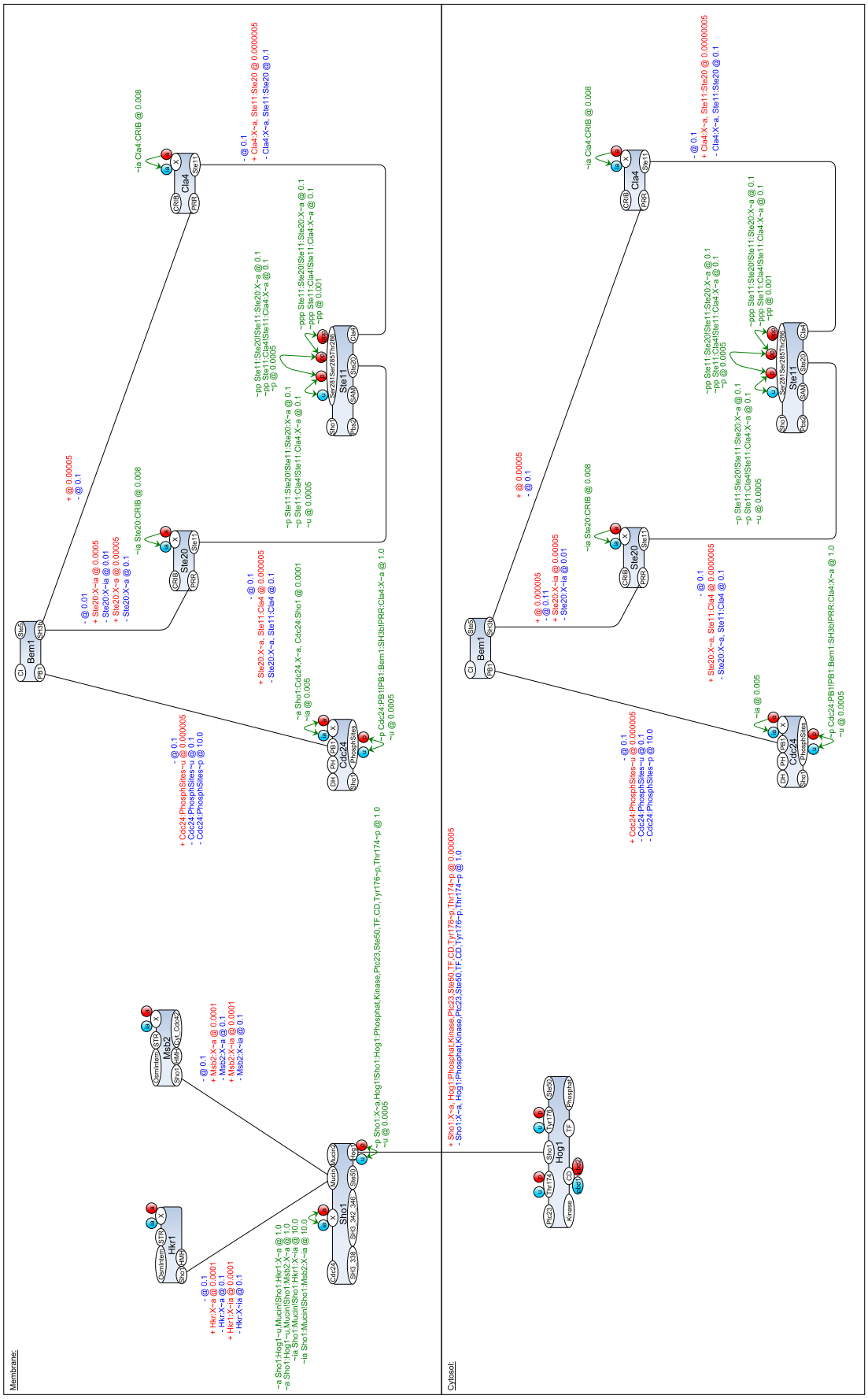


Figure 56: The initial SHO1 response of the high osmolarity glycerol web in yeast. HKR1 and MSB2 transfer the osmotic signal to SHO1. Also included in this chapter are the MAPKKK STE11, and the actions upon it of its close neighbours STE20 and CLA4, mediated by BEM1 and CDC24.

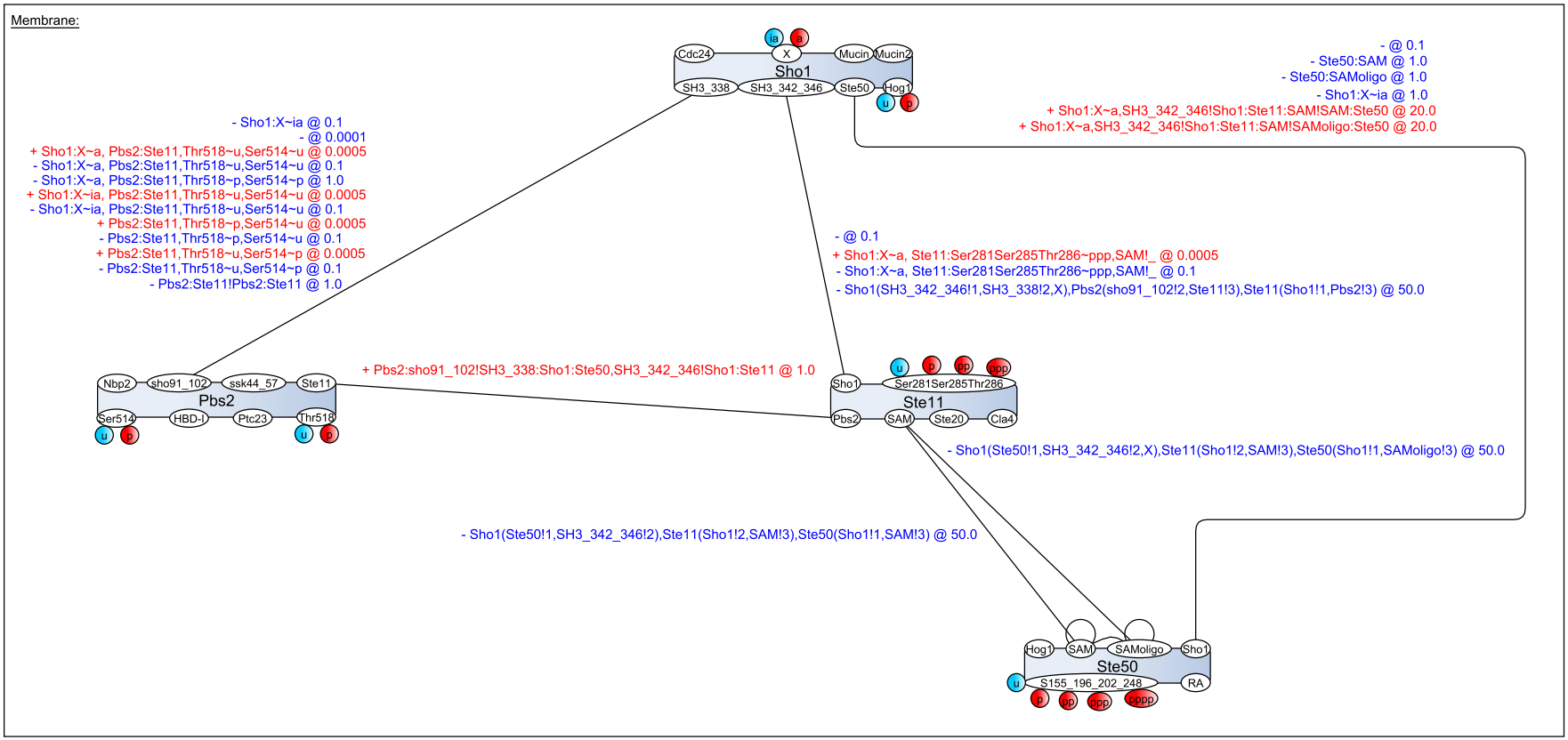


Figure 57: The second part of the SHO1 response of the high osmolarity glycerol web in yeast. Here we show the direct interaction between MAPKKK STE11 and MAPKK PBS2, mediated by STE50 and the osmotic signal transferred via SHO1.

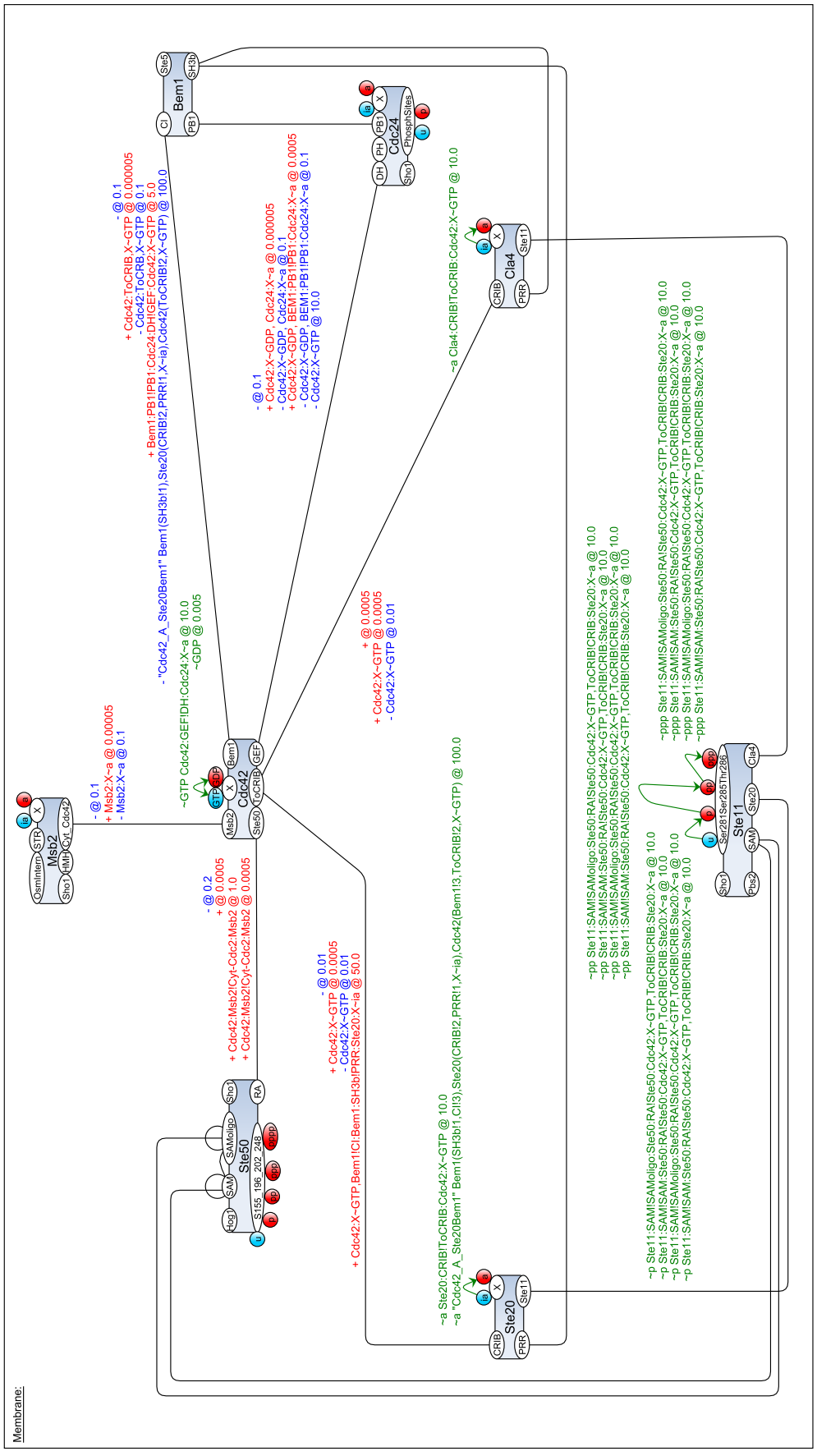


Figure 58: The activity of CDC42 in the high osmolarity glycerol web in yeast. CDC42 bridges many components of the SHO1 response, including CDC24, STE11, and STE50.

visualisations verbose, the number of species described by the binding states of the oligomer in combination with the possible phosphorylation states of STE11 is in itself a good reason to use rule-based modelling to define the module (and the system as a whole).

5.3.6 OPY2 and YCK1_2 Activity

The function in yeast of OPY2 and its associated kinases, YCK1 and YCK2, is not fully understood. It is thought that they participate in both the high-osmolarity glycerol and pheromone response pathways as a membrane scaffold for the STE11-STE50 complex. We represent them in our model as the third chapter mediating the SHO1 response to osmotic stress (Figure 60), where OPY2 can bind to STE50 (with preference for its unphosphorylated form) and can aid the binding of STE11 and STE50 shown in the previous chapter.

5.3.7 MAPK Cascade

In Figure 61 we take a closer look at the MAPK cascade chapter of the high-osmolarity glycerol web, in which the majority of the signal propagation rules are encoded. In many respects this follows the same design philosophy as the simple MAPK cascade introduced in Chapter 2, with a core of cascading kinases (SSK2, SSK22, STE11; PBS2; HOG1) counteracted by a handful of phosphatases. This illustrates how the simple cascade forms a biological motif that is replicated and refined throughout more complex pathways and webs, a fact that is not apparent from the model code but is made clearer when we explore the structure of the visualisations.

We rejoin the SLN1 pathway with phosphorylated SSK2 and SSK22, which bind with and phosphorylate the MAPKK PBS2 in the cytosol. In contrast, SHO1 may either act in the membrane as a scaffold for STE11 to phosphorylate PBS2, or the STE11-STE50 complex may translocate to the nucleus and act directly on the MAPKK. Once doubly phosphorylated from either source, PBS2 binds and transfers the signal to the MAPK HOG1, which then may translocate to the nucleus where it eventually acts as a transcription factor.

Counteracting this signal propagation are the phosphatases PTC1, PTC23, and PTP. These are constitutively present in the cytosol and nucleus of the virtual yeast cell, and act to shut down the MAPK cascade once feedback mechanisms have dampened the upstream signal. Less is known of these interactions than of the primary MAPK cascade, and thus the model uses placeholder agents such as FeedbackDummy to insert an artificial delay between the MAPK and phosphatase deactivation.

Once again we encounter the difficulties involved in visualising complex refined preconditions and interleaving interactions across multiple compartments, and emphasise the importance of layout of causal flow in such a way that the user can naturally trace the dynamics of the system from head to tail. Lacking automation of this vital process, we currently consider it infeasible to step away from manual visualisations.

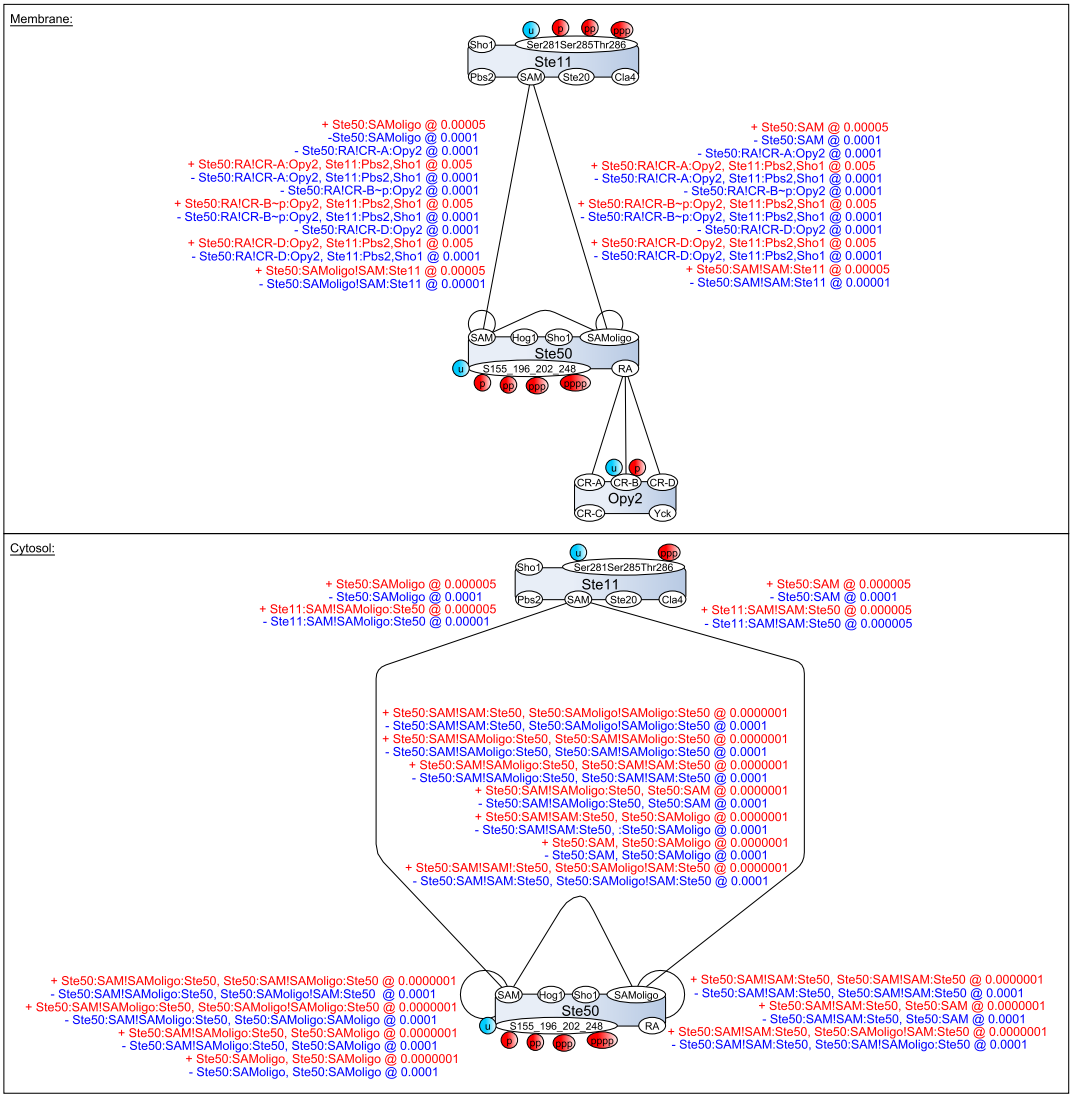


Figure 59: The interactions of STE11 and STE50 in the high osmolarity glycerol web in yeast. The multiple binding configurations of these two proteins play a major role in regulating the activity of the MAPKKK STE11 in the SHO1 response to osmotic stress. They are also involved in multiple other pathways of the yeast cell, including its pheromone pathway, and thus are prime candidates for further study.

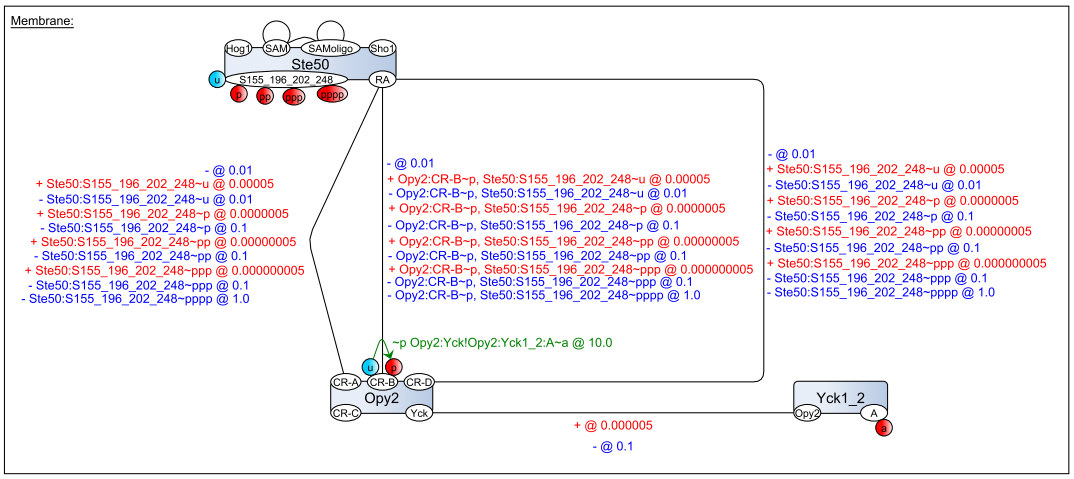


Figure 60: The activity of OPY2 and YCK1_2 (a composite agent combining the activity of YCK1 and YCK2) in the high osmolarity glycerol web in yeast. These three proteins play a minor mediating role on STE50.

5.3.8 Feedback Mechanisms

In Figure 62 we display the two main downstream feedback mechanisms in our model of the high-osmolarity glycerol web. In the cytosol, doubly phosphorylated HOG1 creates (via unknown mechanisms that we currently represent using placeholder agents) internal Osm, which then may travel to the membrane compartment to counteract the effects of external Osm (as shown in the first Osmosensing chapter).

In the nucleus we recreate a number of standard genetic reporter mechanisms (as well as a generic placeholder). The timescales involved in translation and transcription delay this response in comparison to the proteomic response in the cytosol. For ease of modelling the original modeller has chosen to keep the rates equivalent between reporters, although an obvious and easy extension to the model would be to vary these to observe the different responses.

5.4 Conclusions

Although we manually produce no less than ten separate visualisations (nine constituent chapters and the overall spatially-extended contact graph for compartment translocations) to capture the whole of the high osmolarity glycerol web, we find it a refreshing difference to working with over five hundred rules – many differing in little more than a single phosphorylation or binding state – in model code.

Furthermore, we are able to highlight certain modelling assumptions and possible errors made in the process that may not have been immediately evident from the code: in the HOG web described above, the original modeller has made use of bidirectional rule syntax in the

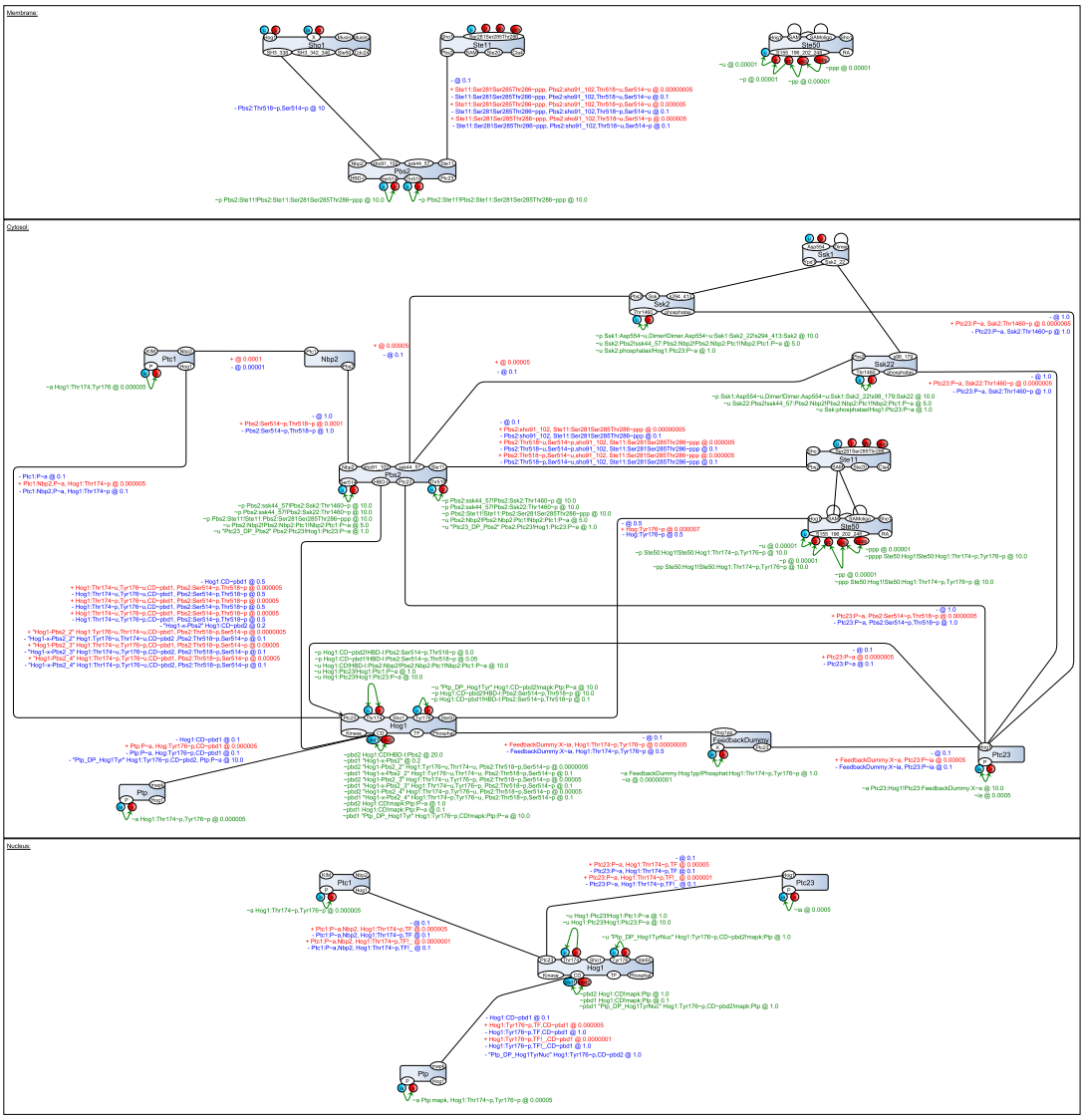


Figure 61: The MAPK cascade contained within the high osmolarity glycerol pathway in yeast. MAPKKKs STE2, STE22, and STE11 act on MAPKK PBS2 in both membrane and cytosol, allowing signal to propagate to MAPK HOG1. Counteracting this is the phosphatase action of PTC1, PTC2 and PTC3 (combined as PTC23), and PTP, mediated by other agents some of which we know little about.

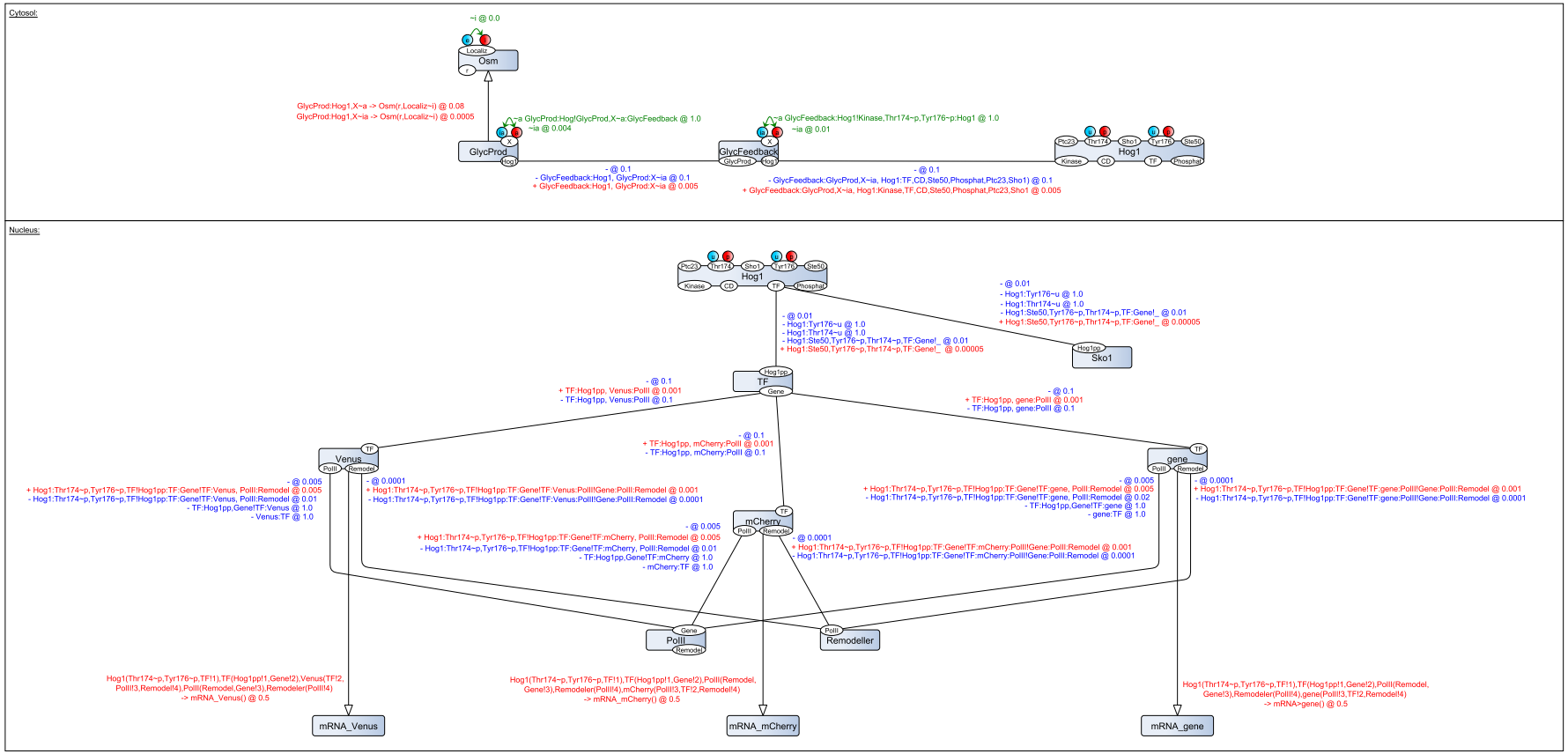


Figure 62: Feedback from the high osmolarity glycerol pathway in yeast. In the cytosol, doubly phosphorylated `HOG1` creates a cascade of reactions eventually resulting in internal osmolarity levels rising. In the nucleus, we show typical genetic reporters (`Venus` and `mCherry`). These operate at different timescales, and the model allows us to observe the lag from the proteomic response to the genetic.

underlying model without care for the reverse condition, meaning that (for example) many unbinding rules are over-specified in their preconditions and likely can be simplified. This is in large part an artefact of the previous version of Kappa used in the development of the model, but illustrates an important aspect of working with biological knowledge bases over time: the backwards compatibility of the methodologies used, and if possible, the need to automate the transfer of knowledge as the underlying techniques evolve.

We see in the HOG web the use of numerous refinements on binding and unbinding rules, which complicate the visualisation labelling process and degrade both its readability and (to a far worse extent, since we are unable to automatically associate a refined rule with its parent) the readability of the underlying model. Although some of these refinements are essential to the dynamic activity of the system (for example, differing unbinding rates of `STE11` from `STE50` depending on the configuration of the latter), others have minimal effect and may be highlighted as artefacts of the modelling process followed. The visualisations enable us to bring these to the attention of the users, listing the offending preconditions in one place for convenience and easing the process of iterative model development. Also in a collaborative context, our interrogation of the Osmosensing module asks how we might best define a workflow to test modules in isolation and to reflect the results of this testing back to the underlying knowledge base.

Development of the HOG web and its associated graphics have also highlighted numerous shortcomings in our visualisation methodology. In particular, the complexity of the interactions described prevents us from displaying the mechanistic interactions in one A4-sized graphic, necessitating the need to split them into modular components. Although this in itself is not an inferior methodology for modelling large complex systems, it does raise the question of how best to define these modules – the examples above show that a simple decomposition based along intuitive mechanistic lines does not always suffice in simplifying the module to the extent where it can be easily visualised.

Finally, we further raise the question of how best to convey dynamic information flow on these static visualisations. We accomplish this in the HOG web through the manual layout of the agents and rules involved from the head to the tail of the pathways in question, but this is not a sustainable solution in models of such size and complexity. To present information to prospective users of the model, and to facilitate discussion and refinement through the process of iterative development, we wish for a better way to convey how the rules of a model tend to affect its agents over time.

We address these issues in further detail in Chapter 7: proposing a model development framework for an iterative testing and refinement workflow as well as to translate between different versions of code and visualisations; developing the idea of multi-scale visualisations to present graphical information in a clear and concise manner; and answering how we might

visualise model dynamics on a static model representation.

For now, we turn in Chapter 6 to complex rule-based modelling over time in the context of functional genetic parts.

Chapter 6

Rule-Based Models Structured to Synthetic Parts

In this chapter we...

- ❖ Address the issues inherent to working with complex genetic parts, including promoter structure and models of transcription.
- ❖ Introduce the current iteration of the Kappa BioBrick Framework (KBBF), which provides a structured framework for modelling BioBrick parts in Kappa.
- ❖ Illustrate the KBBF with a well-known repressilator model.
- ❖ Adapt the model visualisations introduced in Chapter 5 to the KBBF, using them to provide the necessary inputs to a model defined in the framework and simultaneously proposing the usage of the KBBF to overcome the shortfalls of the visualisations in defining and displaying genetic interactions.
- ❖ Introduce the International Genetically Engineered Machine competition, and take a closer look at our award-winning University of Edinburgh iGEM 2010 model of light-based communication in *E. coli*.
- ❖ Provide a further example of these adapted model visualisations in the light-mediated repressilator.
- ❖ Investigate iterative development within the context of modelling synthetic biology using the KBBF.
- ❖ Identify further issues inherent to the current framework, and how we might improve upon it in the future.

Modular methodologies for modelling structured synthetic biological systems, based on

the BioBrick standard [74] and formalised by systems of ordinary differential equations, were first explored by Marchisio and Stelling in 2008 [61]. Other tools such as TinkerCell [9] facilitate modellers looking to incorporate important principles such as stochasticity and analysis paradigms including parameter scanning into the modelling of BioBrick parts.

The Kappa BioBrick Framework, originally laid down by Ty Thomson in 2009 [88], differs from the above by incorporating the advantages of rule-based modelling in its description of dynamic BioBrick parts. Given a specification of a device constructed from BioBrick parts, the Kappa BioBrick Framework provides rules describing how these gene constructs are processed by the transcription and translation machinery of the cell (RNA polymerases and ribosomes). Such a framework allows the modular formalisation of individual functional units within the system and their composition into more complicated devices and systems. It also meshes well with incremental strategies for modelling synthetic biological systems. Finally, the structure of the Kappa BioBrick Framework corresponds with efforts to standardise the characterisation of BioBrick parts, utilising measures such as Polymerases Per Second (PoPS, the rate of transcription defined as the number of times that an RNA polymerase molecule passes a specific point on DNA per second) or Ribosomal Initiations Per Second (RiPS, the level of translation as the number of ribosome molecules that pass a point on mRNA each second). The above features bode well for automation of the modelling process, which we discuss later in this chapter.

We defined an updated implementation of the framework in LMS-Kappa in 2013 [96], from which much of this chapter is taken. Here we present a minor improvement of this updated implementation, before defining data structures and visualisations based on the framework, and then some promising future directions. We then present an award-winning model from the 2010 International Genetically Engineered Machine competition to illustrate the use of this framework.

Our approach is similar to later work by Marchisio et al. [60] regarding the use of the BioNetGen language in a framework for the design of complex eukaryotic gene circuits. Unlike this, the Kappa BioBrick Framework does not specifically cater for synthetic biology based on mRNA regulation using small RNAs, or eukaryotic issues such as compartmentalisation or RNA interference, although these features may of course be added by the modeller. An additional difference is that Marchisio et al. utilise the Model Description Language to provide an interface between modules; the Kappa BioBrick Framework relies instead on the inherent compositionality of the rules in Kappa.

6.1 The Kappa BioBrick Framework

We begin describing the Kappa BioBrick Framework by considering the functional categories of BioBrick parts: promoters, coding sequences, ribosome binding sites (RBS), and termina-

tors. We extend these categories in this thesis to include shims, or spacer bricks with no functional purpose beyond their transcription, and protein domains used for the creation of fusion proteins.

A BioBrick part in the framework consists of one or more DNA agents connected in a chain and tagged with a unique identifier, for example adopted from the Registry of Standard Biological Parts (http://partsregistry.org/). Each BioBrick part also has an RNA representation defined in a similar manner. Given a set of BioBrick parts, the framework will automatically generate these DNA and RNA agents, along with RNA polymerases (RNAP) and Ribosomes involved in transcription and translation, and placeholder Transcription Factor and Protein agents to be manually refined by the user (Figure 63). Organising the agents in this manner allows a modular representation of the transcriptional and translational interactions, without placing any limitations on protein behaviour (which may vary a great deal more) beyond activity as a transcription factor.

The framework generates a concise and complete set of rules, with associated kinetic rates, necessary to describe the activity of these BioBrick parts in an idealised space-homogeneous chassis. Once the rules for a particular virtual part have been established, it can be composed with other virtual parts in a modular manner analogous to the use of actual BioBrick parts in synthetic biology. The paragraphs that follow may be thought of as a recipe with which to create a set of Kappa rules for a certain part.

The crux of the framework lies in three basic rule templates: 'docking', 'sliding', and 'falloff', shown in Figure 64. Every rule generated by the framework, bar the universal RNA degradation rule, is instantiated from one of these templates. At the transcriptional level:

- The 'docking' rule template (Figure 64A) defines the mechanism of transcription factor and RNA polymerase binding to BioBrick promoter parts. The regulatory effect of transcription factors on BioBrick promoters are simply refinements of these 'docking' rules (Figure 65).
- All BioBrick DNA agents require associated rules that describe the transcription of the part from DNA to RNA caused by RNAP; these are the 'sliding' rules (Figure 64B).
- 'Falloff' rules (Figure 64C) deal with the detachment of RNA polymerase and transcription factors from the chain of DNA agents representing BioBrick parts. Although this may in theory happen on any BioBrick part, BioBrick terminators have a higher falloff rate due to their function in preventing further transcription downstream.

Similarly at the translational level, ribosomes may 'dock' at ribosome binding sites, may 'slide' across protein coding sequences to translate the appropriate protein, and may 'falloff' from the RNA chain. The framework also describes the potential degradation of RNA agents at a uniform rate, unlike DNA agents which are assumed to be immutable barring user-defined rules.

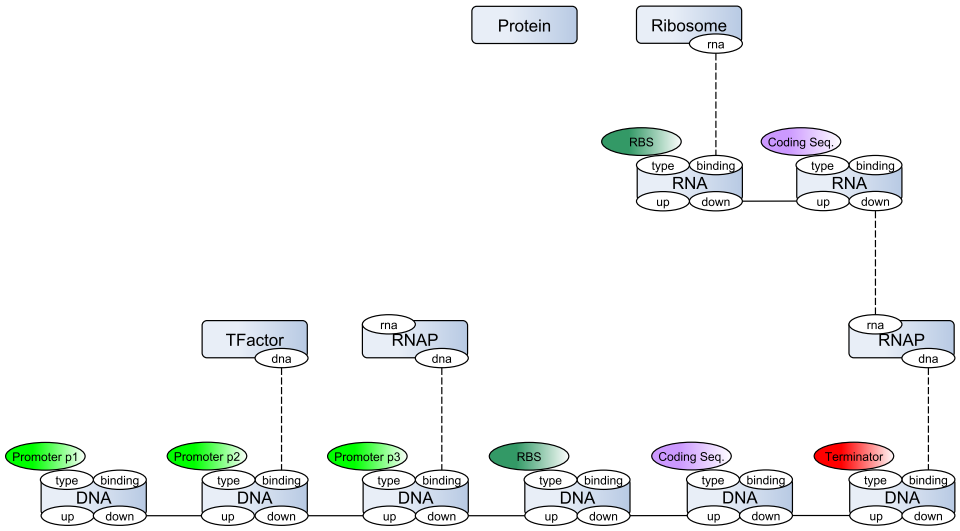


Figure 63: The agents involved in the Kappa BioBrick Framework. At bottom, a chain of `DNA` agents (linked by their upstream and downstream sites) representing the BioBrick parts; each part would have a unique identifier contained as a state to the 'type' site. `RNA` is also represented as such a chain, transcribed from the `DNA` by `RNAP` and subject to translation by Ribosomes. The `TFactor` and `Protein` agents are placeholders automatically generated by the framework, and must be manually refined into the corresponding protein.

The rate constants associated with these framework rules correspond to values that would ideally be known to a rich database of synthetic BioBrick parts (and thus would ideally be automatically derived from said database). For example, the effective rates of 'sliding' rules would correlate with measurements of PoPS for transcriptional activity and RiPS for translational activity. 'Docking' rules would relate to such measures as promoter and RBS strength: transcription factor affinity, RNAP attraction as a function of transcription factor binding, and ribosome affinity. The kinetic rate for 'falloff' rules is determined by such factors as RNAP error rate and terminator efficiency.

It must be clearly stated that the above rules do not take into account the actions of the proteins after they are synthesised or any pathways that they may be involved in, bar their possible effect as a transcription factor. Such considerations (for example, protein degradation or kinase activity) are up to the individual modeller to incorporate separately in addition to rules generated by the framework. Of course, at this point the advantages of modelling in Kappa that we have already explored apply.

The rules of the Kappa BioBrick Framework may thus be categorised along three dimensions: the BioBrick part they are associated with, whether they are involved at the transcriptional or translational level of cellular machinery, and which of the three templates they are based on. We denote these clearly in Table 21.

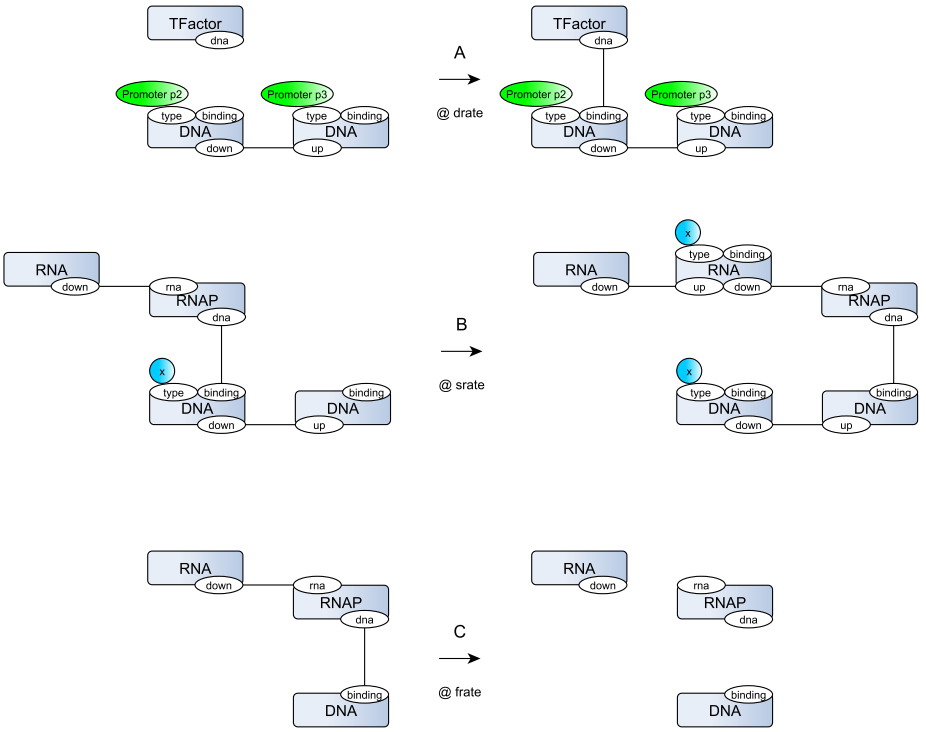
Current levels of implementation allow the skeleton of the framework – the unrefined 'docking', 'sliding', and 'falloff' rules described above – to be automatically generated via LMS-Kappa.

6.2 Example: The Repressilator

To illustrate further the Kappa BioBrick Framework, we now present a model of the hallowed Elowitz repressilator [29] created from BioBrick parts. The code for this model is provided in Appendix 1.

The repressilator combines three simple synthetic genes connected in a loop (Figure 66), such that the product of each gene represses the next gene in the loop and is in turn repressed by the product of the previous gene. The output is oscillatory as the repression of one protein by the second allows the third to build up to repress the second, which in turn allows the first to accumulate to repress the third, and so forth (Figure 67).

Our model consists of seven types of agents (`DNA`, `RNA`, `Ribosome`, `RNAP`, and the three repressor proteins involved) and 61 rules. The agents and rules are composed as described previously for eight BioBrick parts: three protein coding sequences (`LacI`, `TetR`, and `λ-CI`), their corresponding promoters, and one each of a terminator and an RBS. Each promoter is designed to be cooperatively bound by up to two of its associated transcription factors (in this




```
'A (Docking)' DNA(type~p2,binding,down!1), DNA(type~p3,binding,up!1), TFactor(dna) ->
                    DNA(type~p2,binding!2,down!1), DNA(type~p3,binding,up!1), TFactor(dna!2)    @ drate
'B (Sliding)' DNA(type~x,binding!2,down!1), DNA(binding,up!1), RNAP(dna!2,rna!3), RNA(down!3) ->
                    DNA(type~x,binding,down!1), DNA(binding!2,up!1), RNAP(dna!2,rna!3),
                    RNA(type~x,binding,down!3,up!4), RNA(down!4)                                @ srate
'C (Falloff)' DNA(binding!1), RNAP(dna!1,rna!2), RNA(down!2) ->
                    DNA(binding), RNAP(dna,rna), RNA(down)                                      @ frate
```

Figure 64: Instances of the trinity of rules in the Kappa BioBrick Framework: ‘docking’, ‘sliding’, ‘falloff’. (A) The ‘docking’ rule illustrates ‘transcription factor binding’ rule for a promoter from Table 21: the binding of a transcription factor to the correct binding region on the promoter BioBrick part, given that there is no `RNAP` present (note that the binding region on the downstream DNA agent is free). (B) The ‘sliding’ rule illustrates a ‘translation’ rule for a protein coding sequence: an `RNAP` agent moving along the DNA chain and transcribing the appropriate `RNA` as it does so. (C) The ‘falloff’ rule illustrates both universal `RNAP` falloff and the falloff for a terminator: the dissociation of `RNAP` from an unspecified `DNA` part (the terminator falloff rule would have a higher rate constant than the universal rule), and the simultaneous release of the constructed RNA chain (there may be any number of `RNA` agents upstream of the one shown).

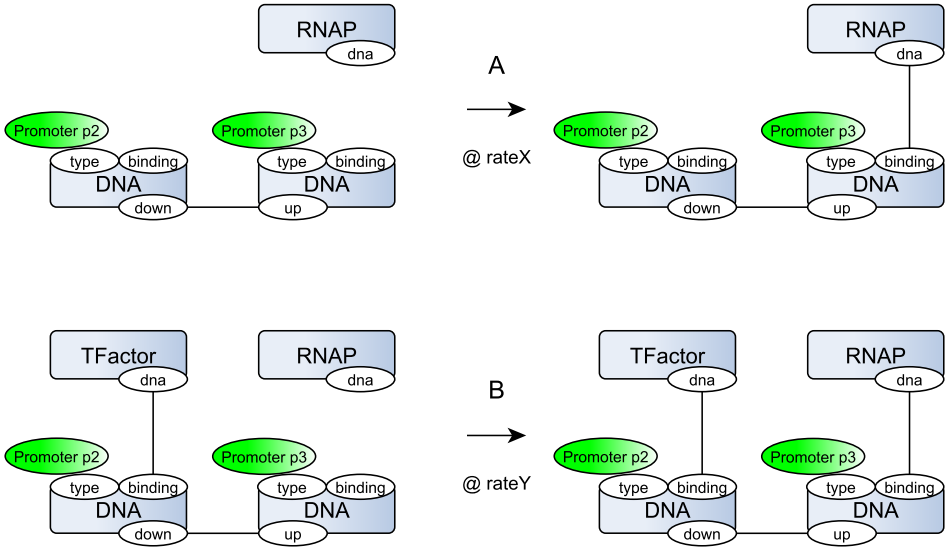


Figure 65: Adapting the 'docking' rules to model transcription factor regulation upon promoter activity. (A) The binding of `RNAP` to a promoter whose regulatory site is free, governed by a rate constant rateX. (B) The binding of `RNAP` to a promoter whose regulatory site is bound to a generic transcription factor, similarly governed by a rate constant rateY. The transcription factor is an activator in rateX is negligible and rateY is significant; conversely, if rateX is significant but rateY negligible, then the transcription factor is a repressor. By manipulating the exact values of rateX and rateY we are able to define leaky promoters or promoters of varying strengths. Finally, we can represent promoters with multiple regulatory binding sites, for example cooperative regulation, simply by elongating the promoter part (increasing the number of `DNA` agents involved) and specifying the effect of multiple transcription factors upon `RNAP` attraction.

	Transcription	Translation
Promoter	Transcription factor binding (*) Transcription factor unbinding (*) RNA polymerase binding (*) Transcription initiation Transcription readthrough	
Shim	Transcription	
Ribosome Binding Site	Transcription	Ribosome binding
Protein Coding Sequence	Transcription	Translation initiation Translation (*)
Protein Domain - Head	Transcription	Translation initiation Translation (*)
Protein Domain - Internal	Transcription	Translation (*)
Protein Domain - Tail	Transcription	Translation (*)
Terminator	Transcription termination Transcription readthrough	
Other	RNA polymerase falloff	Ribosome falloff RNA degradation

Table 21: The rules generated by the Kappa BioBrick Framework at both transcriptional and translational levels. 'Docking' rules are denoted in red, 'sliding' rules in blue, and 'falloff' rules in green. The marked promoter rules need to be manually refined according to promoter structure (repressor or activator, number of transcription factor binding sites); the protein coding sequence translation rule similarly requires user input of the specific protein agent created.

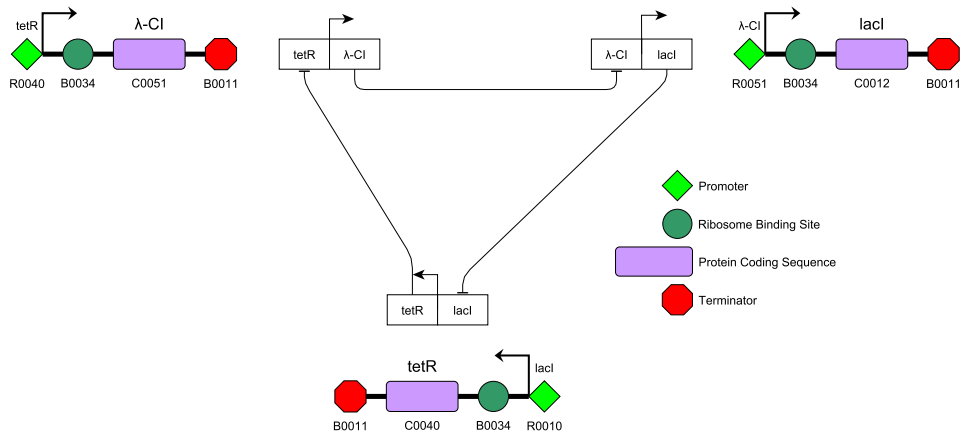


Figure 66: The Elowitz repressilator constructed from BioBrick parts, with each repressor inhibiting production of the next repressor in the loop through the action of the appropriate promoter. Each part may be represented as a modular component (a set of agents and associated rules) in the Kappa BioBrick Framework, and the composition of a number of these parts creates a full circuit as shown. The central representation hides the RBS and terminator components of the BioBrick device, since we reuse the same parts throughout the model.

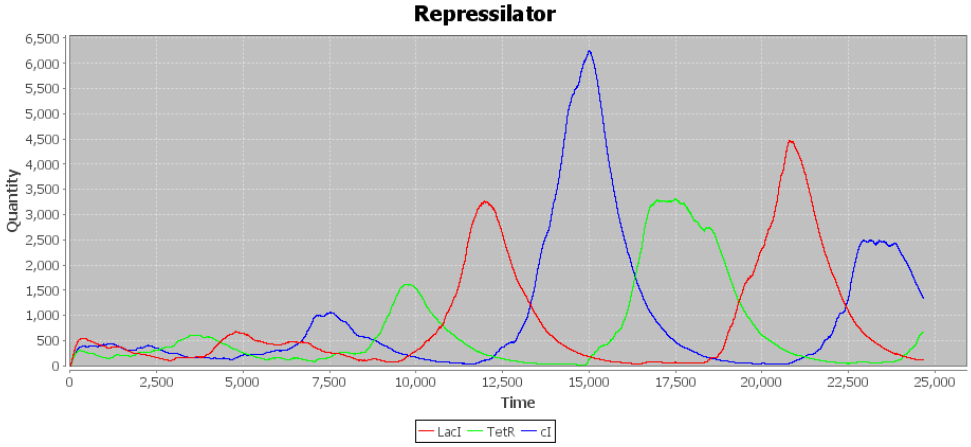


Figure 67: Simulation results for the Elowitz repressilator modelled using the Kappa BioBrick Framework. The system stutters to begin with due to the initial conditions of the model favouring none of the three repressors, but as soon as one begins to assert dominance the system falls into its familiar oscillating pattern. Simulation conducted in Spatial Kappa tool version 2.1.1.

case a repressor), and thus is modelled as a concatenation of four DNA agents (two dedicated to repressor binding, one for RNAP binding, and a linker or spacer separating the promoter from any preceding part). Every other BioBrick part is modelled as a single DNA agent. The terminator and RBS are reused across all three devices.

In the paragraphs to follow, we describe the rules of the model according to the BioBrick parts they are associated to. Readers may find Table 21 useful in keeping track of the parts and their associated rules.

The first three rules of the model are RNAP and Ribosome 'falloff' rules that apply to all DNA agents (BioBrick parts), and an equally universal RNA degradation rule.

Each promoter requires up to $2n(2^{n-1})+2n+2$ rules, where n is the number of transcription factor binding sites, (2^{n-1}) represents the number of occupancy contexts in which a binding or unbinding can occur, and $2n$ is the number of transcription factor binding configurations that the RNAP may depend on. These rules decompose to:

- up to $n(2^{n-1})$ rules to describe the possibly cooperative binding of the transcription factor(s).
- up to $n(2^{n-1})$ rules more to describe unbinding.
- up to 2^n rules to describe the binding of the RNAP.
- a singleton rule describing initiation of the transcription process.
- a further singleton rule to deal with transcription readthrough by transporting any RNAP that arrives at the linker to the RNAP binding site (thus allowing the action of further rules).

In the repressilator model, $n = 2$ for all promoters, hence each promoter requires 14 rules to describe its behaviour. All of the transcription factor binding and unbinding rules are combinatorial in the number of binding sites, in this case two, since (for example) the binding of a transcription factor at one site is conditional on whether or not the other site is already bound. Similarly, we model four rules to describe the binding of RNAP: one for when no transcription factors are bound, one each for when either of the sites are bound, and one for when both sites are bound. The eight rules describing the binding of various agents to the part are all variations on the 'docking' base rule, the four rules describing unbinding are variants on the 'falloff' base rule, and both transcription initiation and transcription walkthrough are based on the 'sliding' base rule.

Of course, if the combinatorics of the promoter are known not to fully affect its activity, for example the rate of RNAP binding, enumerating all occupancy contexts is possibly an overly detailed solution. Currently it is unclear how one might benefit from the potential regularities in promoter structure by saving on the cost of describing them.

The protein coding sequences each require three rules, all based on the 'sliding' template:

- a rule for its transcription.
- a rule for initiating translation.
- a rule for the actual translation of the protein.

As may be surmised from the rule descriptions, only the first rule is active at the transcriptional level; the other two operate at the translational level. Protein domains, although not included in this particular model, are defined in a similar manner. Head domains require all three of the above rules, whilst internal and tail domains need not specify translation initiation. Shims, also not included in this model, require only the rule for their transcription.

The RBS is described via two rules:

- a transcription rule ('sliding').
- a ribosome binding rule ('docking').

Transcription of the RBS `DNA` agent creates an equivalent `RNA` agent, upon which the `Ribosome` binding rule may then fire as the first step of the translation process.

The terminator also requires two rules:

- a 'falloff' rule with enhanced rate to represent its efficiency at terminating transcription.
- transcription readthrough ('sliding') if its terminator function fails.

The first of these terminator rules overrides the generic `RNAP` 'falloff' rule, thus making the terminator much more efficient at removing `RNAP` from the `DNA` chain than any accidental falloff.

The proteins created in the model act as transcription factors as described in the promoter rules, but have no other activity beyond three rules describing their degradation. These final three rules are the only rules in the model not specified by the Kappa BioBrick Framework, but are required to produce the oscillations characteristic of the repressilator.

The initial conditions of the model contain one of each of the BioBrick devices shown in Figure 66, as well as a population of RNA polymerase and ribosomes. By identifying the `LacI`, `TetR`, and `λ-CI` protein agents as the observables of interest, simulation of the agents and rules described above generates a time course similar to that shown in Figure 67.

Note, of course, that the framework as described above is not the only way to model BioBrick parts in Kappa; other, less modular formalisms may be of equal use to the modeller. But in comparison to the ad hoc modelling of synthetic biology described in the nitrosylase induction model in Chapter 2, we gain structure and traceability by defining synthetic parts in

the manner described above. We also achieve guaranteed composability of parts, and stronger links to online part repositories such as those described later in this chapter.

By no means does the framework completely capture every interaction necessary in fine detail. As an example, the RNA degradation process makes no distinction between exo- and endonucleases; it simply destroys `RNA` with free binding and downstream sites, which is roughly equivalent to half the activity of an exonuclease. A simple modification to the base framework would be to differentiate between the two. As another example, the transcription factors in the framework associate specifically with their binding site upon the promoter, whereas in reality they search for their appropriate binding sites by 'sliding' along the DNA chain; a possible extension of the framework would be to take this action into account. A further extension might be to make use of the linker portion of the promoter and the promoter's transcription readthrough rule to represent the length of non-coding region between BioBrick devices. We might even refine the `RNA` agent to model the ability for multiple `Ribosome`s to exist on a single mRNA, thus fully distinguishing between PoPS and RiPS.

6.3 Visualising the Kappa BioBrick Framework

Our main theoretical contribution of this chapter, in addition to the minor updates of the Kappa BioBrick Framework described above, is to refine the previously developed model data structure and its visualisations to incorporate the KBBF. Simultaneously, we introduce methodologies for graphically defining user inputs (promoter architecture and combinatorics, association between the protein coding sequence and the produced protein) to the framework. We see these visualisations as providing a standardised 'specification' for the definition and activity of a rule-based model in the KBBF.

The visualisations introduced in Chapter 5 are suitable for visualising protein-protein interactions in Kappa for synthetic biology. However, we found via the examples explored that complex genetic preconditions rapidly overwhelm the capabilities of the visualisation to provide a clear and easily understood graphical representation. In particular, additional visualisation constructs are required for the definition of the genetic element of the KBBF. We focus on giving the user the means to define promoters and operators according to their behaviour in the context of the current KBBF implementation in LMS-Kappa.

Spatial definitions would be useful for the definition of synthetic circuits in eukaryotes such as yeast. We believe that it is a simple matter to integrate them into visualisations for the protein-protein interaction level of the model, by making the assumption that the genetic circuits defined below are located in the nuclear compartment. For now we concentrate on prokaryotic circuits such as those present in *E.coli* for ease of representation. Our example visualisation (Figure 68) is the repressilator model defined in the previous section of this chapter.

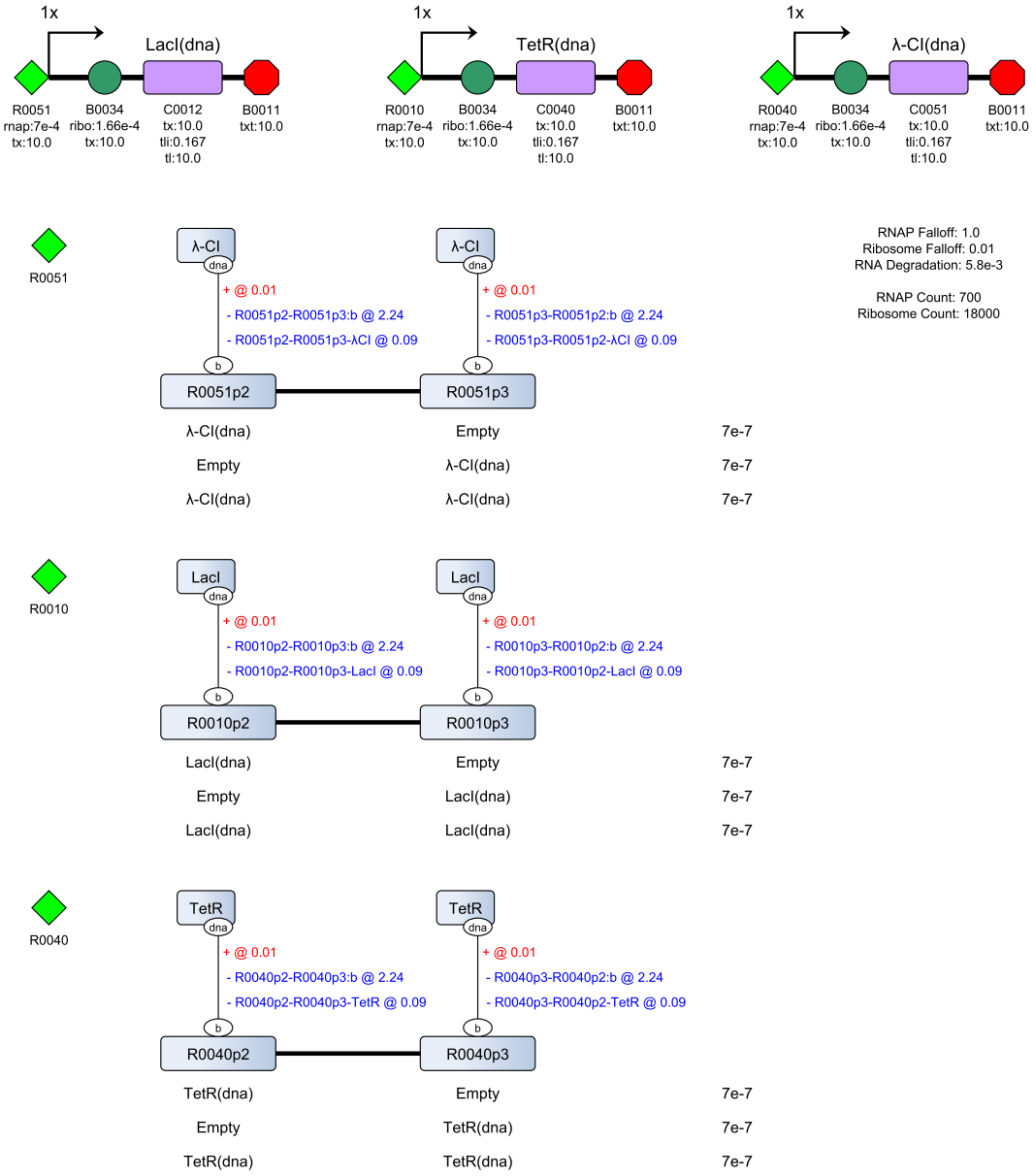


Figure 68: Visualising the genetic component of the repressilator (as defined in the KBBF). On top we define copy numbers and structure of the BioBrick devices involved in the model, as well as the various rate parameters they utilise. The bulk of the visualisation is devoted to describing operator structure and the effect of transcription factor binding upon RNAP attraction.

We begin by visualising the initial conditions for the genetic component. In particular, we define the immutable composition of individual parts in the initial conditions of the model, as well as how many copy numbers are involved. Different categories of parts are given different symbols in the visualisation, for example a green diamond for a promoter part. Each part is named, for example with its BioBrick part number, and labelled with the stochastic rates necessary according to its functionality in the KBBF. We give a list of part categories, visualisation symbols, and necessary rates (with the associated reference label in brackets) below.

- Promoter (green diamond): basal `RNAP` binding rate (rnap), transcription initiation rate (tx).
- Shim (black rectangle): transcription rate (tx).
- RBS (green circle): transcription rate (tx), ribosome binding rate (ribo).
- CDS (purple rounded rectangle): transcription rate (tx), translation initiation rate (tli), translation rate (tl).
- Protein Domain - Head (purple rounded square): transcription rate (tx), translation initiation rate (tli), translation rate (tl).
- Protein Domain - Internal (purple rounded square): transcription rate (tx), translation rate (tl).
- Protein Domain - Tail (purple rounded square): transcription rate (tx), translation rate (tl).
- Terminator (red octagon): transcription termination rate (txt), transcription readthrough (txr).

Not all of the above rates must strictly be specified, but by leaving them blank we provide a default value for their dynamics. By omitting the transcription readthrough of the terminator in Figure 68, for example, we set its rate to 0 and disable the functionality in the model.

Our visualisation requires two further components. First, in our device definition, we associate the protein coding sequence or internal protein domain with the agent created via its translation (in the example shown, `LacI`, `TetR`, and `λ-CI`).

Secondly, we need to specify the chain of operators per promoter, and their interactions with respective transcription factors. Per operator, we define the binding conditions of the different transcription factors and transcription factor complexes in a format reminiscent of Binding rule effects in the protein-protein visualisation. We then specify in a table below the visualisation the occupancy conditions upon the operators that cause a modification (also specified) to the RNAP binding rate.

Figure 69: Visualising the protein-protein interaction component of the repressilator (as defined in the KBBF). Given that the activity of the three proteins as a transcription factor is already covered in Figure 68, the only additional activity we need to describe is their degradation.

The visualisation displayed in Figure 68, combined with the KBBF, thus describes all of the repressilator model except for protein interactions not directly covered by the transcriptional activity. We show this as a standard MDS visualisation in Figure 69. Together, the two visualisations thus suffice to serve as an alternate representation of the KBBF repressilator model as a whole.

6.4 Use Case: Light-Based Communication in *E.coli*

Let us explore a further example of modelling in synthetic biology, building on the ideas described above. We revisit the light-based communication model introduced in Chapter 2 of this thesis, attempting to visualise the complex model in a more tractable manner. We also examine it in terms of an iterative modelling framework, identifying areas where a structured environment would have proved useful for collaborative development.

The International Genetically Engineered Machine (iGEM) competition is an annual undergraduate competition in synthetic biology, drawing together students from various backgrounds including biology, engineering, informatics, and social science. The goal of these interdisciplinary teams is to design and construct novel synthetic biological systems, utilising modular DNA building blocks known as BioBrick^{TM}parts, that extend the library of well-characterised modular parts for use in future projects (for further information, readers are directed to `http://ung.igem.org`).

The University of Edinburgh iGEM team was one of 118 such teams that participated in the 2010 competition, attaining a Gold Medal standard at the Jamboree held at the Massachusetts Institute of Technology in November. The focus of the team's project was the development of a light-based communication system, involving the establishment of three independent channels of communication in different spectral wavelengths to enable bacterial cells to react with each other as well as with electronic systems. Crucial to this was the modelling of the individual light sensors and emitters in action as well as analysis of the proposed system as a whole, and

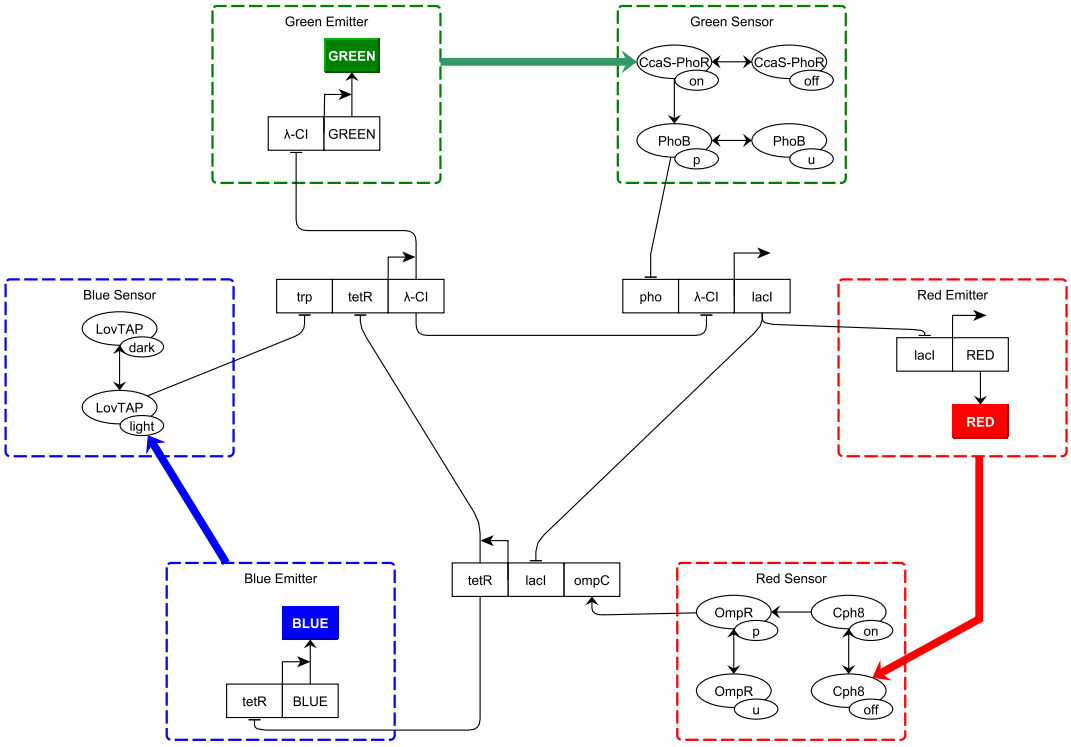


Figure 70: Modelling light emitting and light sensing pathways coupled to an Elowitz repressilator. At the centre the oscillating repressilator regulates the emission of light in the system. The light sensing pathways then provide input to this central regulator via a second promoter coupled to the same coding sequence, reinforcing or adjusting responses to synchronise the system. The central repressilator and the light emitting pathways are genetic components, built solely around the Kappa BioBrick Framework, but the light sensing pathways are protein interaction components that must need be manually specified by the modeller.

for their efforts the University of Edinburgh iGEM team won the Best Model special prize.

The modelled system used the standard Elowitz repressilator (as described above) as the basis of a composite device integrating three sets of light sensors and emitters (Figure 70). Previous efforts in the iGEM competition had focused on the development and characterisation of these sensors and emitters, but not on matching paired sensors and emitters of similar wavelengths, or on considering their combined use in a single system.

We have already outlined the details of the model and its simulation results in Chapter 2, but recap them briefly here.

6.4.1 The Model of Light-Based Communication

Each gene product in the repressilator loop is used to repress the production of light of a particular wavelength. In the meantime, the light sensing pathways provide input to the core re-

pressilator to reinforce the oscillatory response. The model of light communication is thus composed from seven components (the core repressilator as described above, along with emitters and sensors for the three wavelengths), each of which we may consider a functional model in its own right. In total, these modules contained 145 rules detailing the function of sixteen BioBrick parts (including the core repressilator), the function of three constitutively expressed light sensors, and the function of the proteins associated with the various pathways.

Green light emissions were based on the standard firefly (*Photinus pyralis*) luciferase enzyme, and then underwent site-directed mutagenesis to create a red light emitter. The blue light emission system was developed from a bacterial luciferase from *Xenorhabdus luminescens*. Within the model, these were all depicted in the simplest possible manner: a promoter repressed by a corresponding repressilator protein, controlling the direct production of the luciferase.

The red light sensor was based on a bacterial phytochrome, the blue light sensor on a plant phototropin, and a novel fusion protein was designed as a green light sensor. The red and hypothetical green sensors were modified from two-component regulatory system: red light turns off a sensor protein, which in turn stops activating a downstream activator attached to the corresponding repressilator part; green light instead turns *on* its sensor protein, which in turn activates a downstream repressor attached to the corresponding repressilator part. The blue light sensor was an allosteric hybrid protein with a less complicated mechanism: blue light activates the repressor directly, allowing it to repress the corresponding repressilator part. This introduces an inbuilt delay to the red and green sensors that is not present in the blue light sensor, which in turn means that the latter is noticeably more sensitive to perturbation experiments. The mechanism does not have a major effect in the overall model, however.

We verified that each module in isolation reacted to perturbatory inputs as expected (i.e. the downstream component of the blue light receptor activated upon reception of photons corresponding to the correct wavelength), and that the model as a whole did not display any aberrant behaviour from the sum of its parts. Typical simulation results for the complete system are displayed in Figure 71, showing an oscillating structure with each light output linked to the absence of its corresponding repressor (for example, high levels of red light are linked to an absence of the LacI repressor within the system). The light thus produced helps to reinforce the internal oscillations by inhibiting the production of a second repressor protein within the system (in the case of the red light, inhibition of the TetR repressor eventually allows blue light to be produced).

We hoped to synchronise cells running individual instances of the repressilator such that they expressed the same output wavelength. Thus we extended this intracellular model by simulating the behaviour of an idealised virtual colony of bacteria communicating with each other using the light produced within each cell. As explained in Chapter 2, we leveraged an early customised implementation of Spatial Kappa to define a two-dimensional hexagonal biofilm

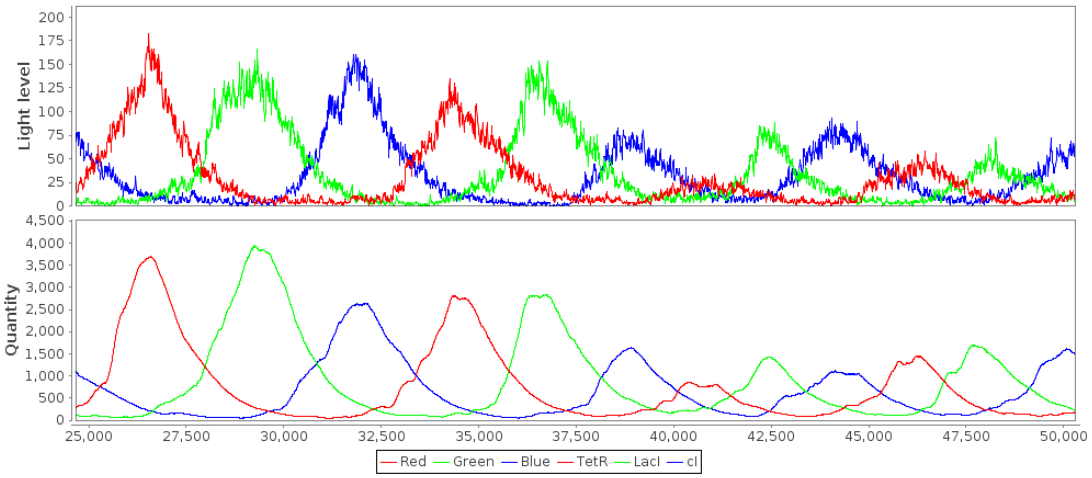


Figure 71: Simulation of the complete light network model, showing the interactions between light (top) and the genes in the core repressilator (bottom). Units (time and concentration) are arbitrary.

of cells, each with its independent synthetic light communication device, communicating with their neighbours in an attempt to synchronise. This communication took the form of diffusion of the 'light' agents between neighbouring cells according to a set (arbitrary but non-trivial) diffusion rate. Simulation results (Figure 72) showed that a communicating colony had less time-averaged deviation in light levels between cells, and therefore increased synchronisation, as desired.

6.4.2 Visualising the Light-Based Communication

We visualise the light-based communication circuit in the manner described in the previous section. Figure 73 and Figure 74 describe the genetic component of the model, whilst Figure 75 adds to protein-protein interactions necessary to complete the light feedback.

We note that the genetic component shown in Figure 73 and Figure 74 may be further distilled into a visualisation on the same agent-centric level as the protein-protein interactions in Figure 75, which would then correspond directly to the underlying model. However, the sheer number of rules on the `DNA` agent, as well as the complexity of their preconditions, makes such a visualisation a poor choice to describe the system. We find that the figures shown here fully specify the necessary details for defining the model in the KBBF in a manner that is intuitive to the modeller and easy to define.

6.4.3 Iterative Collaborative Development

The two-fold modular nature of modelling in the KBBF – on the genetic level and on the protein-protein interaction level – lends itself neatly to an iterative and incremental develop-

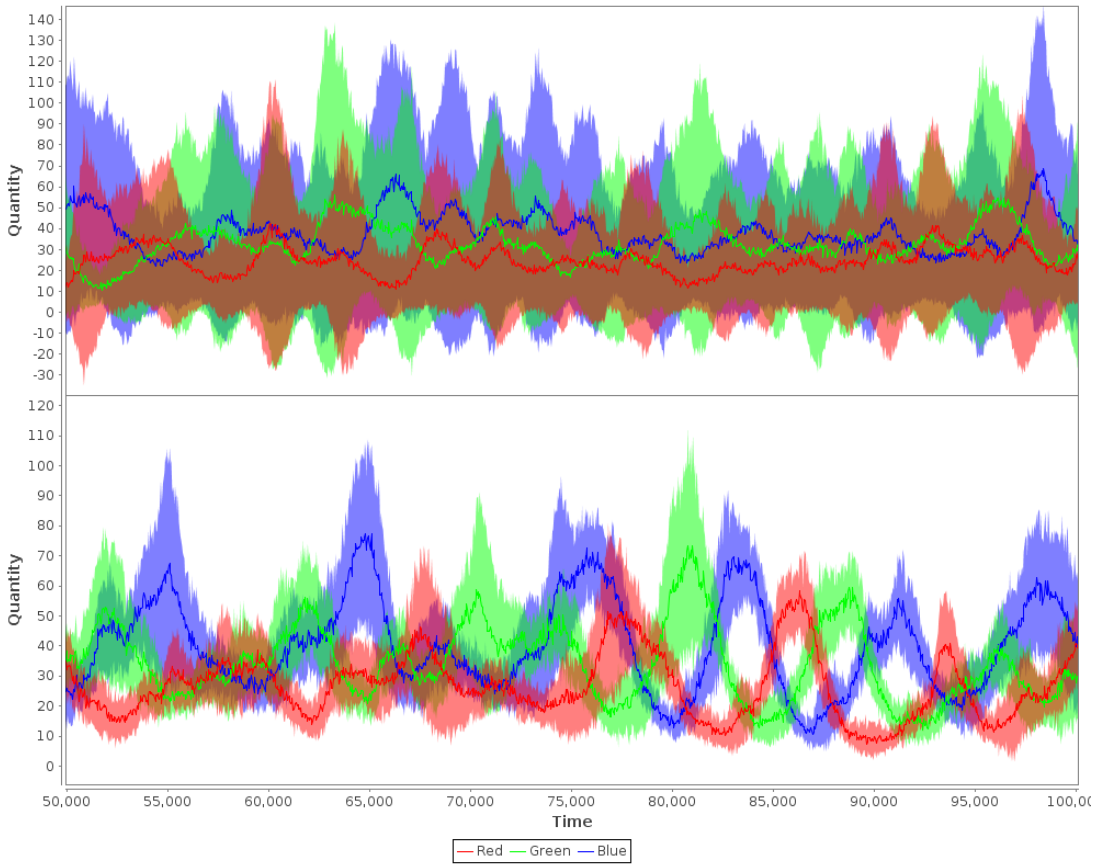


Figure 72: Comparison of mean cell light levels in 4x4 colonies of cells, both isolated (top) and communicating between cells in the colony (bottom). Shaded areas on the graphs show standard deviation of cell light levels from colony mean. Communicating colony shows lower standard deviations, and hence increased coherence in cell activity levels. Units (time and concentration) are arbitrary. Simulations conducted in Spatial Kappa tool version 1.0 (output modified manually to display mean and standard deviation).

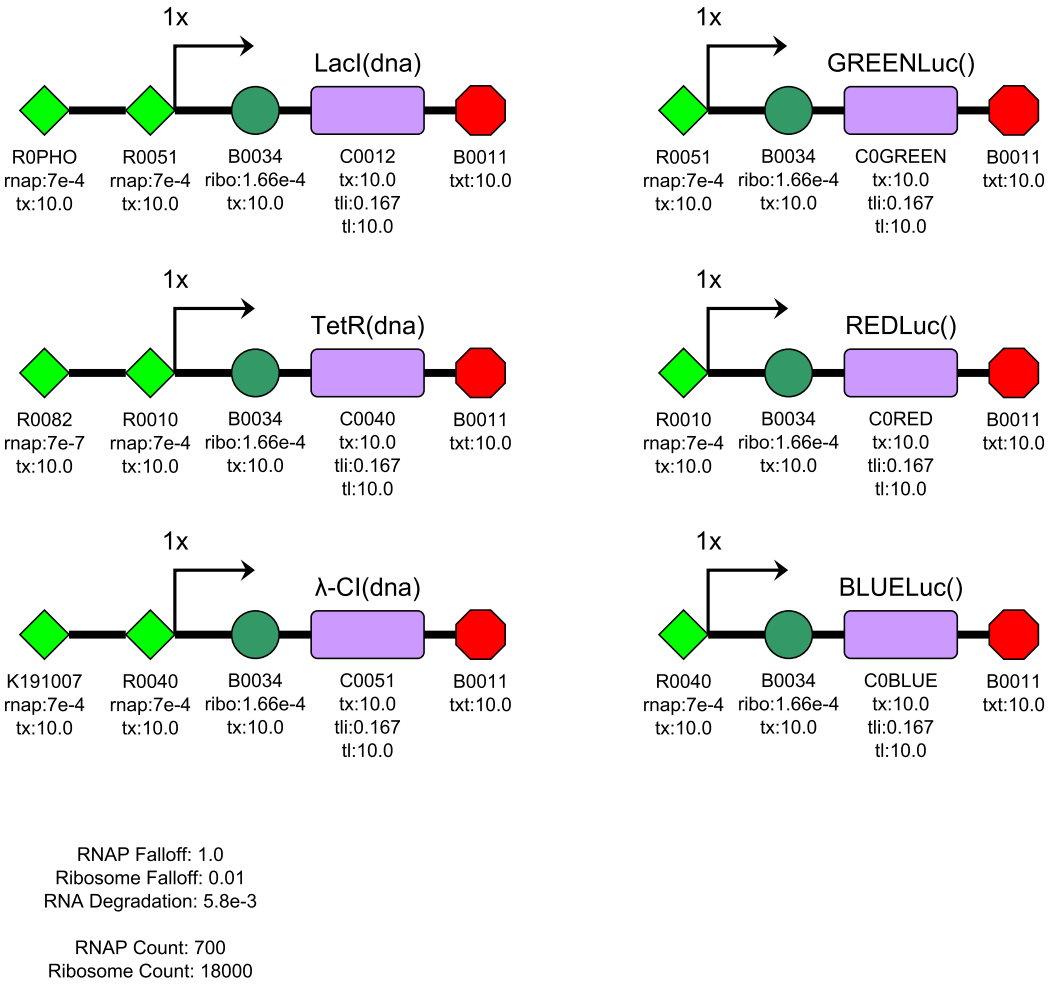


Figure 73: Visualising the copy numbers and structure of the BioBrick devices involved in the light-based communication circuit (defined in the KBBF).

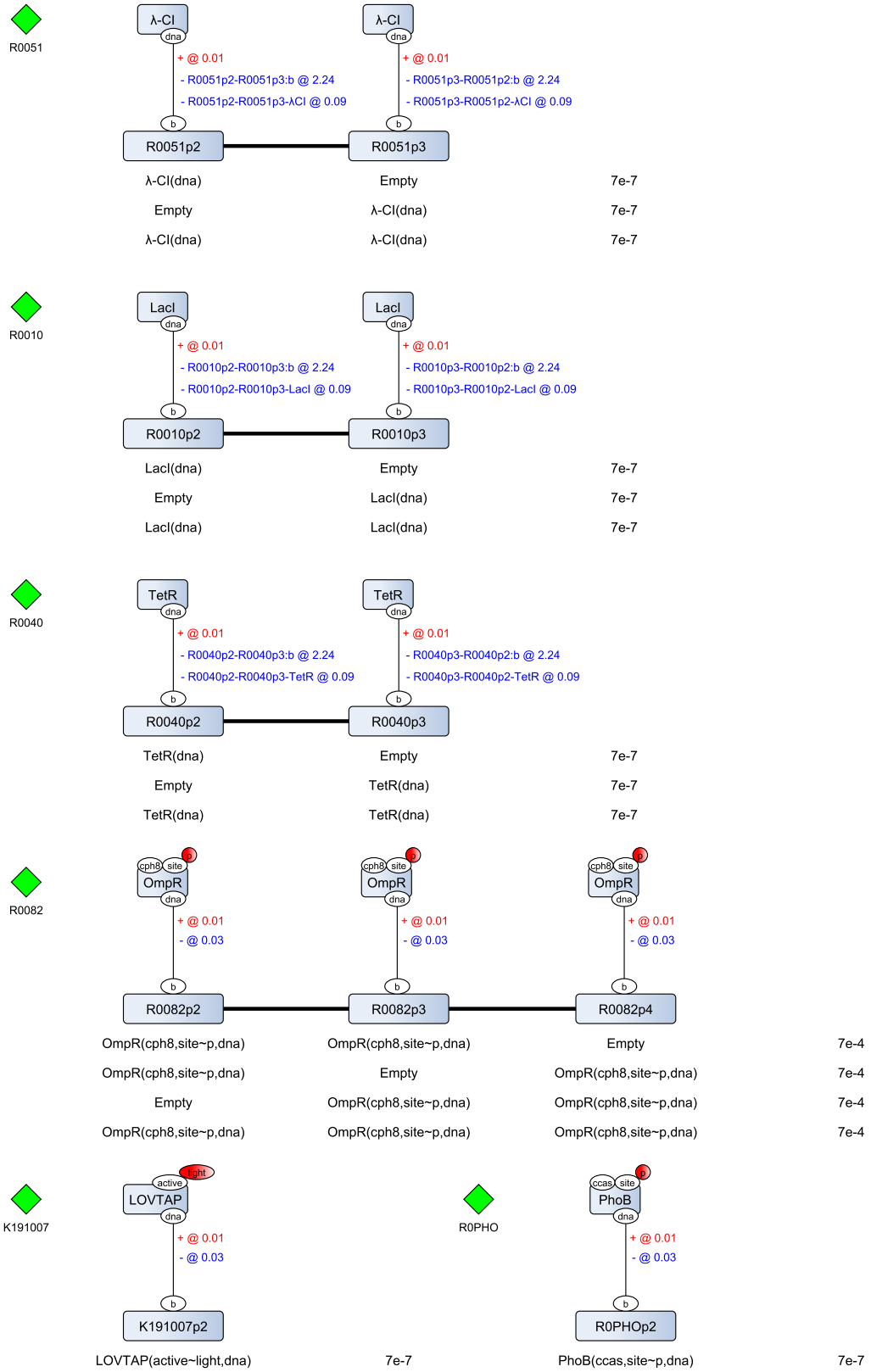


Figure 74: Visualising the operator structure and the effect of transcription factor binding upon RNAP attraction in the light-based communication circuit (defined in the KBBF).

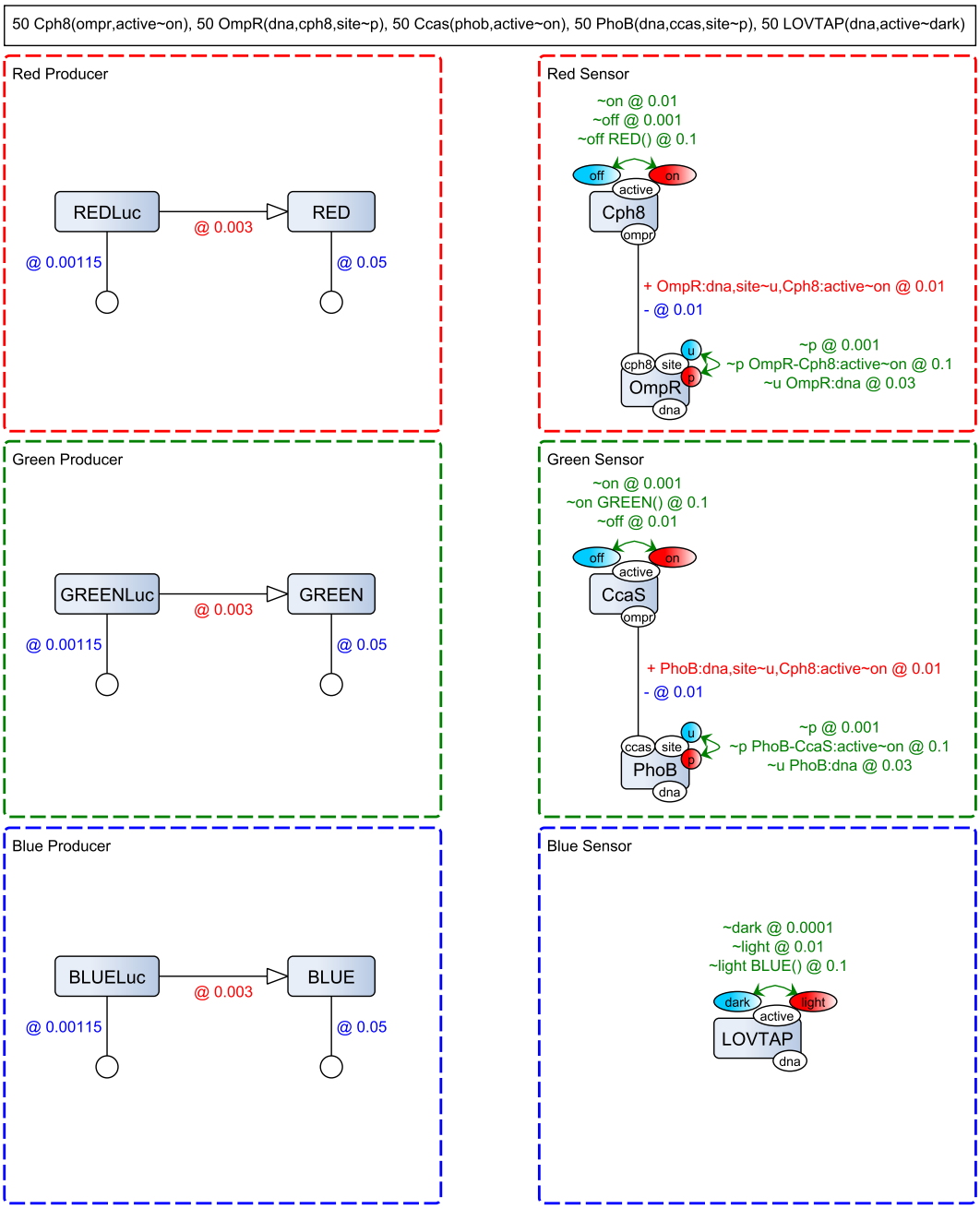
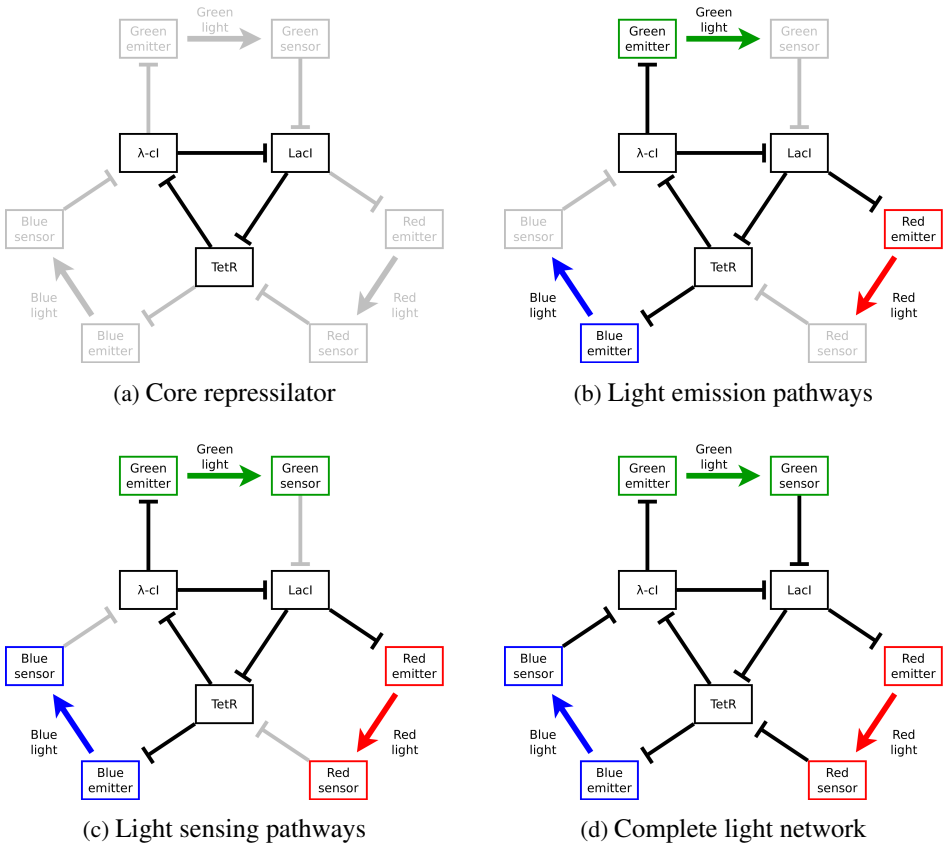


Figure 75: Visualising the protein-protein interaction component of the light-based communication circuit (defined in the KBBF). We show the light production and light sensing components for each wavelength.



(a) Core repressilator

(b) Light emission pathways

(c) Light sensing pathways

(d) Complete light network

Figure 76: Evolution of the light communication system, from core repressilator to complete network, showing the modular nature of the system and the iterative development approach.

mental approach to a project. The ability for individual modules to be developed and verified independently greatly simplifies the process of breaking down the overall effort into manageable units, amenable to repeated cycles of refinement and extension.

As an example, we composed the light-based communication model of seven components, each of which could be considered a functional model in its own right; these 'sub-models' could be validated individually before being combined into a whole.

The first iteration of development adapted the core Elowitz repressilator component (Figure 76a) from a similar model created by Ty Thomson as the proof-of-concept of the KBBF. We then added the light generation pathways (Figure 76b) to the model, thus producing oscillating light outputs linked to the oscillations of the core repressilator.

The next development iterations linked the light sensing pathways to the core components. Initially, we model these pathways without allowing the produced transcription factors to bind to the repressilator (Figure 76c); this gave us the ability to test the individual pathways in isolation, via perturbation analysis. Finally, we extend the interim models to allow transcription factors produced by the light sensing pathways to bind to the BioBrick parts in the core repres-

silator and affect their transcription directly (Figure 76d), thus resulting in the complete model of the overall target system.

6.5 Conclusions

The KBBF allows us to compose the genetic components simply by adding new rules and initial conditions to the model. Concatenating the protein-protein interaction components, however, still requires the user to merge them manually. For example, we wish to be able to revisit the light sensing pathways in isolation for further perturbation analysis, and to automatically reflect the results of this analysis to the underlying model: something that is not easily possible as currently specified. We have touched upon the need for better support for modular components and their testing in the previous chapter in the context of the high-osmolarity glycerol web in yeast, and reiterate it here.

A large proportion of the development involved repeated refinement of the model and its accompanying kinetic parameters to obtain the behaviour necessary for oscillation. Due to constraints on time and equipment in the biological component of the project, these rate parameters were derived from *in silico* trial-and-error analysis rather than *in vitro* experimentation. However, we lacked the tool support at the time to do this in a structured fashion. Only after such verification can we then look to store the derived value in a database of such parameters, explicitly associating the parent part with the rate so as to automate the processes of updating it over time and of incorporating it in newly developed models.

We feel that leveraging a modelling development framework in the presence of techniques such as parameter search will give us the capability to better explore individual or multiple variables (for example, rate parameters or a binding effect between two agents) in terms of their effect as modelling artefacts upon simulation observables. Current representations of the model code make it difficult to visualise such artefacts, and to automate their exploration by spawning multiple variants of the model. An object-oriented format model data structure, combined with the operational capabilities of an IDE, would neatly circumvent these difficulties.

Finally, we deliberately left unanswered the question of whether the model could approximate or predict the behaviour of such a synthetic circuit in its physical host. This was largely due to the fact that we were unable to complete the biological component of the IGEM project in parallel to the modelling, and thus had no means of testing the model (in relation to the parameter refinement above). But the advent of genomics and high-throughput measurement techniques, and the maturing of computational methodologies for explaining complex phenotypes and processing them in parallel, have paved the way for us to begin discussion on what we term a ‘virtual chassis’: a way to pair a synthetic circuit such as the light communication model with a ‘whole-cell’ model of its host organism, and to observe the effects that the former

has on the latter.

We explore these questions in further detail in Chapter 7: proposing a model development framework for iterative testing and refinement; updating model knowledge via underlying, collaboratively curated databases; and modular whole-cell models as they relate to the concept of a virtual chassis.

Chapter 7

Modular Development Environments and Visualisations

In this chapter we...

- ❖ Propose and create a draft implementation of a collaborative framework for developing, evolving, and iteratively refining a model over time.
- ❖ Introduce bidirectional model transformations to modelling in biology, allowing users to maintain consistency of edits over multiple user views upon the underlying source information.
- ❖ Evolve this model information through the process of creating and refining a biomodel knowledge base, including across different versions of the underlying modelling language.
- ❖ Discuss further visualisation of large and complex webs of data, including: automation of placement and layout of visualisations, hierarchically linking contact graphs with a chapter overview, and displaying dynamic stories on the static visualisations.
- ❖ Comment on how we might associate elements of a model in the KBBF with parametric or interaction data in an underlying bioinformatics or synthetic biology database, thus allowing us to automate updates over time.
- ❖ Introduce the Module Integration Simulator (*mois*), and the emphasis it places on verifying individual model components as part of a coherent whole.
- ❖ Explore whole cell models in *mois*, including the idea of coupling a synthetic circuit (such as one created in the KBBF) to a virtual host chassis.

The greatest strength of modelling in the rule-based Kappa language lies in how it alle-

viates the combinatorial complexity of models with multiple states (i.e. phosphorylation and binding). However, we have seen that this is not the only source of complexity in biological modelling; in particular, over the last two chapters, we have looked closer at complexity that arises from working with large-scale models over time. In Chapter 5, we introduced large, highly-interconnected models in which the source of complexity is the sheer size of the system, and we attempted to alleviate it by introducing visualisations to allow modellers to work on the model without the need to dive into dense and oft-impenetrable code. In Chapter 6, we looked at the complexity of promoters in models of transcription and translation, and proposed a structured framework for simplifying this via standardised genetic parts and automated rule generation.

In both cases, however, we have identified further questions to ask of our solutions. In the case of our visualisations, we ask how we might develop multi-scale visualisations ("overview first, zoom and filter, then details-on-demand") to present complex graphical information in a clear and concise manner, as well as how we might visualise model dynamics on a static model representation. In the case of the Kappa BioBrick Framework, we question how we might update the model as the underlying knowledge evolves in the form of collaboratively curated databases, and how we might test synthetic circuits thus created in a 'whole cell' virtual chassis. Finally, in both cases we ask how we might ease the collaborative development of a mechanistic biological model over time: by including visualisations and different versions of model code in a single workflow, by providing a means of collaborative editing of model components, and by supporting iterative testing and refinement.

In this chapter, we take a look at more recent and unfinished work that seeks to address these shortcomings.

7.1 A Framework to Aid the Iterative Collaborative Model Development Workflow

Bidirectional model transformations [16, 82] from software engineering and database management are a means to maintain consistency as we make updates upon different 'views' of an underlying knowledge base. Thus they provide an algorithmic basis for allowing users to easily manipulate model parameters and structure on the representation of their choice, and to write these updates back to the underlying knowledge base without fear of this choice affecting the final output. In the context of biological modelling these representations or views may include different versions of Kappa code (thus allowing us to algorithmically translate between them as necessary, and providing a structured means to address artefacts such as the bidirectional rules in the HOG web in Chapter 5), and visualisations thereof.

We have begun to implement such a framework, allowing us not only to work efficiently

with a rule-based model as a mutable repository of knowledge, but also to define a workflow for associating bioinformatics and literature knowledge with model elements (and thus evolving models alongside the underlying knowledge instead of abandoning them at a certain point in time). It would facilitate the communication of ideas between experimental and computational biologists, and act to aid collaboration in the creation and further development of biological models in the same way as integrated environments provided for domain experts and software engineers in model-driven software development.

Ultimately we are successful in applying a subset of functional bidirectional model transformations to a simple model, and in identifying avenues for further refinement. The content in this section is based on joint work with Assistant Professor Soichiro Hidaka, undertaken during a six-month internship to the National Institute of Informatics, Tokyo from September 2012 to March 2013. A preliminary version of the content was presented and published at the BX2013 workshop [95].

7.1.1 Bidirectional Model Transformations

Bidirectional model transformations are mechanisms for establishing – and re-establishing, in the presence of change – the relationships between models and code in particular when the related artifacts may be edited independently [17]. Such edits are necessary for different users to view information in specialised notations, possibly at different levels of abstraction, or if the related artifacts contain information independent of one another.

Bidirectional graph transformations, whether based on a Triple Graph Grammar (TGG) approach [72], a functional approach [43], or otherwise, may be considered as a specialised subarea of bidirectional model transformations based on synchronising and maintaining the consistency of two or more related graph data structures. Our aim is to apply these tools in an integrated development environment to the Kappa model code and visualisations.

The functional approach to bidirectional graph transformations, as epitomised by GRoundTram [44], allows us to use a graph query language (UnQL+ [45]) to perform forward transformations upon the biological knowledge base to translate it into the version of model code or visualisation of the modeller's choice, in the knowledge that the backward transformation (reflecting any changes made on this representation back to the source) is guaranteed to be correct.

7.1.2 A Bidirectional Collaboration Framework

Intuitively, we envision all operations upon a biological knowledge base as encoded by a model – for example the refinement of model parameters or the addition of components to the network – to correspond to well-known database query and manipulation operations available in UnQL and UnQL+: `SELECT`, (in-place) `UPDATE`, `INSERT`, and `DELETE`. We wish to make these

operations possible on a user-friendly representation of the knowledge base, whether this be model code or visualisation. Unfortunately, while such representations may help to display complex information in a natural manner, they are not amenable to the query-based approaches that make low-level graphs suitable as database representations and as subjects to bidirectional transformations. To resolve this, we propose to distil high-level data structures such as rules and agents to an equivalent edge-labelled digraph representation via bijective transformation, and to use graph querying languages such as UnQL+ (via GRoundTram) to manipulate and query this digraph data structure (DDS).

We might approach this scenario by:

1. Defining a model data structure for the knowledge base shared by domain experts including experimental biologists and computational modellers,

2. Defining bijective conversions between the above data structure and 'low-level' graph representations – the latter amenable to bidirectional transformations via the existing tool GRoundTram [44],

3. Defining bijective translations between the knowledge base and user-friendly domain specific representations,

4. Allowing users to query the knowledge base from these domain specific representations to produce a refined view suitable for manipulation, and

5. Transforming from the queries defined in (4) to UnQL+ (an extension to UnQL [6] with graph editing constructs) queries in GRoundTram so that the domain expert queries may be bidirectionalised via low-level graphs and UnQL+ queries upon them.

We establish bijective conversions and translations between the different data structures involved in the framework, thus enabling their bidirectional transformation via GRoundTram. Figure 77 displays a graphical overview of the framework, in which the region surrounded by the dotted line is revisited as Figure 78a and further explained later in this chapter.

We posit (1) to take a form similar to the visualisations described in Chapter 5, in the form of entities and rule effects. Based on this we sketch an overview of (4), assuming the presence of (2), (3), and (5). Both (1) and (4) are actively implemented in the form of OCaml modules attached to but independent of the GRoundTram distribution.

This approach may be compared with another data synchronisation strategy that uses bidirectional transformations, namely, the scenario outlined in [34]. As one possible usage of lenses, this scenario utilises a common abstract view created by multiple lenses over multiple concrete data storages. Each concrete data storage can maintain, in addition to data that is synchronised with other concrete data storages, its own data that does not participate in the synchronisation.

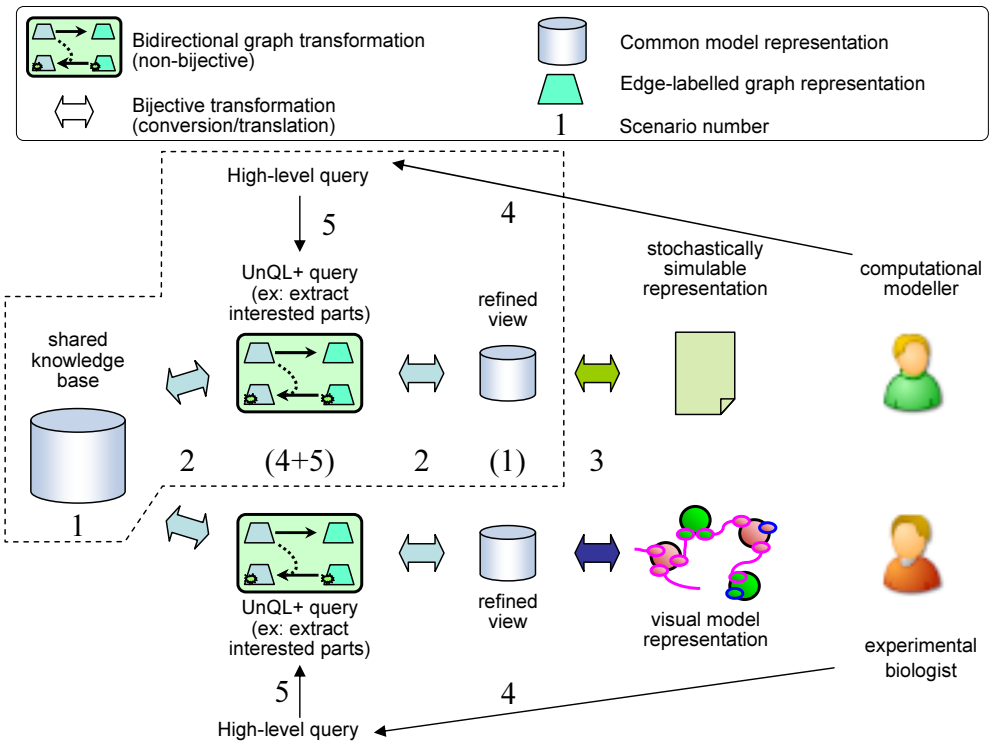


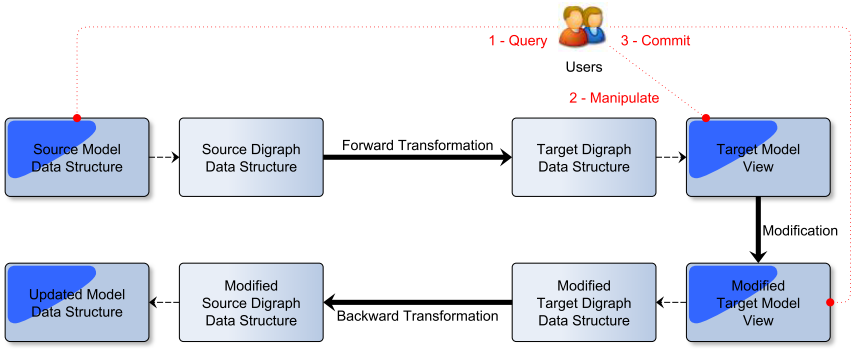
Figure 77: An overview of the proposed collaboration framework.

In contrast to this we adopt domain specific abstract views over a common concrete high-level data structure (Figure 78b), converting bijectively from this high-level data structure to a simpler format that enables bidirectional transformations. We also provide bijective mapping between domain specific representations and the common high-level data structure, thus allowing multiple user groups and representations to utilise the bidirectional transformations in a manner best suited to their needs. We believe that this approach is applicable to any scenario in which collaboration is paramount, rather than particular to biological modelling.

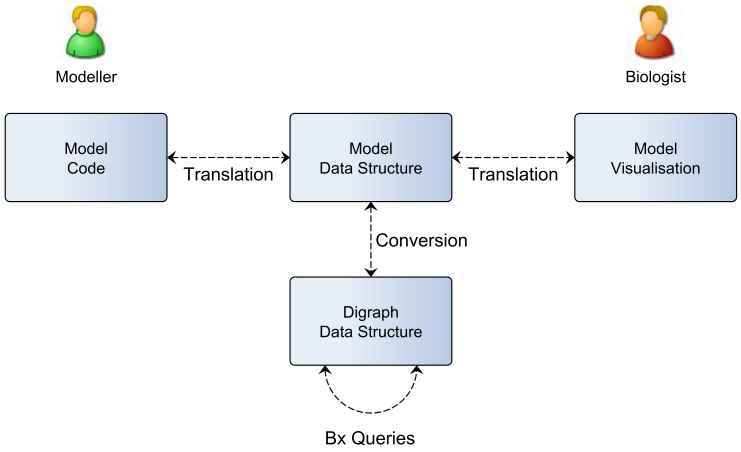
Let us proceed with a sample use case of the collaboration framework as applied to the extended Kinase-Phosphatase model introduced and visualised in Chapter 6. We provide implementation details regarding update operations on the framework in the section to follow.

7.1.3 Conversion

We begin by converting the extended Kinase-Phosphatase model to a digraph data structure (DDS) for the purposes of graph querying and bidirectional transformation (Figure 78b). We assume that the algorithms involved in translating (at the top level) and converting (between different levels) between these data structures are straightforwardly bijective and thus information preserving. We thus perform bidirectional transformations upon the low-level DDS, only utilising the aforementioned bijective algorithms to provide a user-friendly representation of



(a) A closeup of the bidirectional transformations involved in the framework.



(b) The data structures involved in the framework, emphasising the different representations that collaborators may utilise upon the same underlying knowledge.

Figure 78: Users interact on the top level with the appropriate representation for their task and expertise; these representations correspond to a model data structure which can be translated to its digraph equivalent. Queries and bidirectional transformations are performed on the digraph data structure and passed back to the top level.

this information.

To form this DDS, we parse the entities of the extended Kinase-Phosphatase to form a tree-like structure. Each branch in the tree represents a sequence of attribute information (stored on the edge rather than at the node) corresponding to the assumptions and information underlying the entity or the rule. Appropriate entities are then linked together according to the rule effect references, forming a vast interconnected web of information. A simplified example of such a DDS, as applied to the extended Kinase-Phosphatase model, is provided in Figure 79.

Of course, we are by no means limited to Kappa model code and visualisations as our representations upon the system; the opportunities for new representations are limitless, provided they can translate from a (theoretical) shared underlying data structure. Neither are we limited to Kappa as our top-end user interface; for example by applying the fragmentation algorithm [40] on a Kappa model we may extract a system of ODEs amenable to further analysis. We believe that this modularity is a major advantage in future applications for the framework.

Our next step is to define update operations possible via bidirectional graph transformation on the DDS.

7.1.4 Forward Transformation (Graph Querying)

Let us assume that our user is a modeller wishing to make an update on the underlying knowledge base, whether to add a bioinformatics or literature reference or to refine agents or interactions within the model. They would start with a complete representation of the source model data structure, for example a model visualisation. From this, they may query for a subset of the information encoded in the model to create a view, up to and including its entirety (i.e. an identity query). In future iterations of this framework, we expect this query to automatically translate to UnQL+ from a query upon the model data structure; for now, we abstract from this translation and focus on querying the DDS in UnQL+.

For example, let us assume that our user is an expert in the `Cytosol` (compartment id $c01$) of the extended Kinase-Phosphatase model. It is a simple matter to construct a graph query to select only the subset of the model that is of interest. We give the raw UnQL+ below, but eventually expect that we will be able to compose simple queries (i.e. those that isolate individual compartments and agents) for example by selecting the appropriate entity in a GUI.

```
select {Compartment:{c01:$g}} where {_*.Compartment.c01:$g} in $db
```

Such `SELECT` queries take advantage of the tree-like DDS structure to locate the target entity, isolating it and its children for the user's perusal. At the moment, however, their expressive power is constrained by limitations upon the ability of GRoundTram to provide bidirectional transformations upon their results. For example, although it is possible to define a more complex query to eliminate the transport rules entirely from a single compartment, we are currently

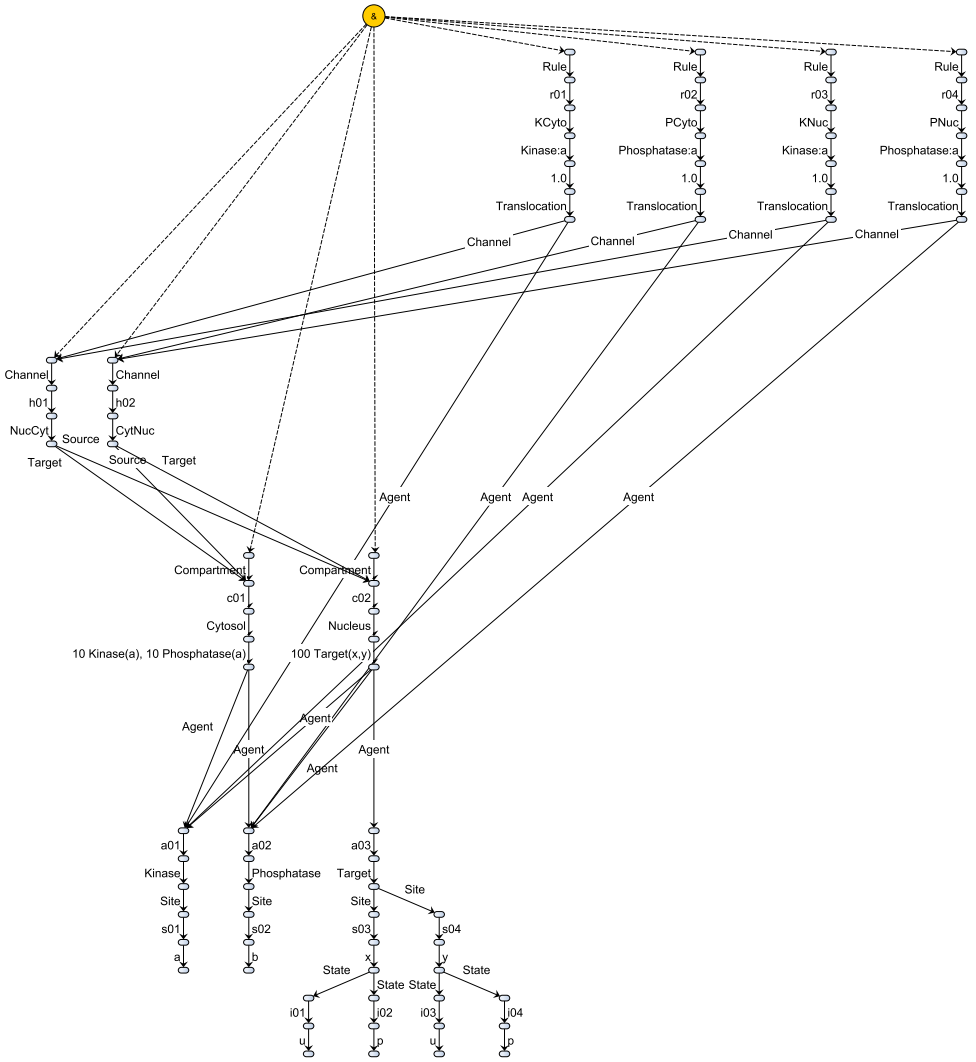
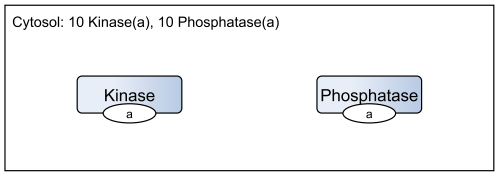
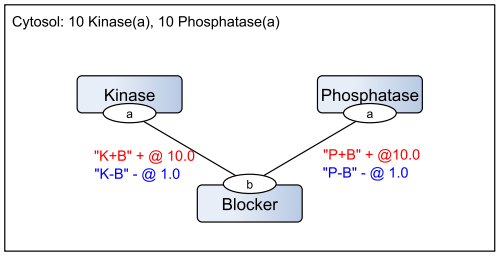


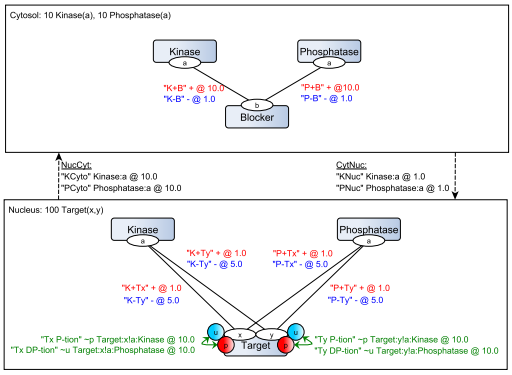
Figure 79: A simplified digraph data structure for the extended Kinase-Phosphatase model. For ease of representation, we omit unused attributes and all rules bar Translocation rules. The hierarchical branching tree-like structure of the entities in the model forms the large cluster at the bottom of the diagram. These are linked together by the rule effects (of which the full model has a far larger number).



(a) A target view on the extended Kinase-Phosphatase model after undergoing a simple query (forward transformation) eliminating all but the cytosolic compartment.



(b) The target view after modification. The user has added a new agent *Blocker* to the model, competitively binding with the existing *Kinase* and *Phosphatase* and thus preventing their nuclear transport.



(c) After committing the modifications, the source model data structure is updated such that the new source reflects the changes made.

Figure 80: A series of visualisations depicting how a model evolves through one update iteration of the framework. Note that the modification made is meaningless in the context of simulating the model – another separate update is required to add the *Blocker* to the initial conditions of the compartment!

unable to reflect insertions upon this target view (i.e. the addition of new agents to the compartment) back to the source in an efficient manner. In the future we wish to provide a wider, more complicated range of queries without compromising the functionality described in the following sections. For now, we believe that being able to restrict the view of a model to a single compartment or a single agent (whilst remaining assured that any modifications made will reflect correctly upon the underlying knowledge base) represents a major step forward in user functionality upon the visualisations and the model proper.

A forward transformation simply interprets the query to create a target DDS from the source. We then translate this bijectively to a user-friendly view as shown in Figure 80a.

7.1.5 User Update Post-Processing

The user is then able to modify the (selected subset of the) model as they desire. Again, we consider for now updates that correspond to well-known database query and manipulation operations as available in UnQL and UnQL+: (in-place) `UPDATE`, `INSERT`, and `DELETE`. Currently

we lack a dedicated `MERGE` or `JOIN` operation, although this would definitely of use when considering large, complex, and modular models such as the high-osmolarity glycerol web in yeast.

For example, let us suppose that our user wishes to add a third agent `Blocker` to the `Cytosol` compartment, which binds with the existing `Kinase` and `Phosphatase` agents and thus obstructs their translocation to the `Nucleus`. The corresponding changes to the model are shown in Figure 80b.

Simple in-place `UPDATE` refinement of edge labels – the renaming of entities within the model, the definition of new initial conditions, modifications made upon the rule preconditions, or rate parameter refinement – is possible via basic bidirectional transformations using the in-place update modules of GRoundTram. From a high-level user viewpoint, operations in this category correspond to model modifications that simply edit pre-existing information, regardless of the representation chosen.

`INSERT` operations via in-place modification of the target model are currently possible only if (1) edits are made directly on a subgraph of the source extracted by the query using the graph variable reference, and (2-1) the subgraph is not traversed by the query, or (2-2) the inserted subgraph does not include any subgraph that would be extracted by the query. Conditions (1) and (2-1) may be explained by [90] where the query is polymorphic with respect to the extracted subgraph, i.e., the query does not depend on the content of the subgraph and just sends it to the target without modification.

However, a subset of UnQL+ queries (for example, involving the ‘delete’ construct) recursively copy said source to obtain their desired result. Reflecting subgraph insertion from a target queried in this manner incurs high recursively enumerable complexity; any modifications made upon the target violate condition (1). By limiting queries to compartments and agents it is possible to reflect all necessary insertions as may be feasibly desired by users (compartments, agents to an existing compartment, sites to an existing agent, states to an existing site, all forms of transitions) upon a modified target back to an updated source.

With regards to the characteristics of the chosen DDS representation, the current structure ensures no cycles are formed upon entity or rule insertion, which acts to simplify the subgraph insertion process and keep it within the operating capabilities of the current GRoundTram implementation. Once again, the GRoundTram in-place update modules are used for such operations. From a high-level user viewpoint, operations in this category include the construction and integration of new structural or dynamic information, such as the addition of new agents and their behaviour into the model.

`DELETE` operations upon subgraphs also require careful balancing of the expressive power of UnQL+ against the current implementation of GRoundTram, for example ensuring that all associated rules and rule effects are eliminated along with the parent entity. The GRoundTram

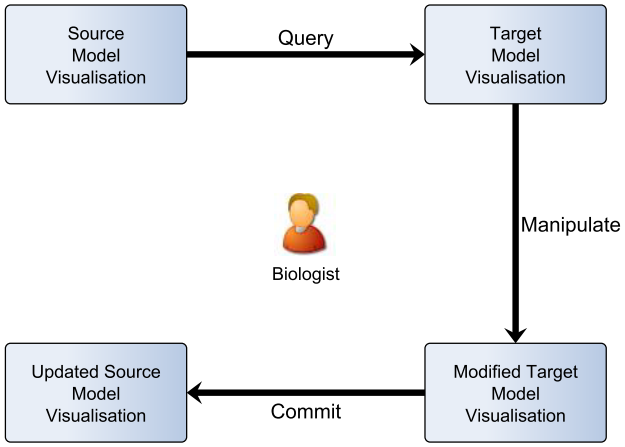


Figure 81: How a sample user may view the collaboration framework, completely abstracting away from the underlying implementation and the bidirectional graph transformation tools there.

module dedicated to deletion allows us to successfully reflect the deletion of both entities and transitions from a target view. From a high-level user viewpoint, operations in this category include the removal of existing structural or dynamic information from the model.

7.1.6 Backward Transformation

Once satisfied with his or her modifications, the user commits them to the backward transformation mechanism of GRoundTram. The modified target model is bijectively translated once more into its DDS and integrated back to the original source, creating an updated source model (Figure 80c) for future use.

7.1.7 Additional Considerations

To prove this concept we have implemented the above operations upon the DDS in a basic form, as a set of OCaml functions independent of the main GRoundTram distribution, and have used them to generate the results above. Throughout this process, it is our desire that the user remains completely unconcerned with the underlying DDS structures or the queries made upon them (Figure 81).

Another consideration arising due to the collaborative nature of biological modelling is the issue of conflicting modifications upon the knowledge base, requiring domain expert knowledge to identify and resolve. This will require the integration of a version control system into the framework, allowing such changes to be tracked and resolved without causing irrecoverable conflicts. For now we assume that the knowledge base is ‘locked’ during each edit.

The advantages of the framework as proposed are threefold: the ability to conduct UnQL+ graph queries on high-level graph data structures as long as the conversion (from MDS to

DDS) is well-defined; the guaranteed correctness of reflecting the target modification back to the source, no matter what view the modification is made on; and the opportunities for traceability and structured support (in terms of the manipulation options available to the user) for an iterative development methodology.

7.1.8 Future Directions

We have presented above an adaptation of the GRoundTram bidirectional graph transformation framework, aimed at facilitating the collaborative creation and integrated development of biological models. We have demonstrated some of the advantages of such an approach, including the querying of representations of high-level graph data structures using UnQL+ and the traceable reflection of user modifications back to their underlying source using bidirectional transformations.

In light of our current implementation there is room for further enhancement in the underlying GRoundTram implementation: streamlining input files and formats (currently UnCAL files are used interchangeably with both UnQL+ query inputs and DOT graph inputs); improving runtime error messages (currently difficult due to missing traceability between UnQL+ queries and their desugared internal UnCAL counterparts); further clarifying the conditions under which the in-place update module may be used for updates beyond simple edge label refinement; and generally improving user experience by providing a more accessible layer of functionality above the edge-labelled digraphs. Related to this, further work is also needed in broadening the range of queries available to the user, and ensuring their compatibility and correctness with the bidirectional graph transformations provided. It is hoped that the GRoundTram toolset grows in parallel with the framework, as new use cases and new issues are brought to light and examined.

Eventually we hope to generalise both the conversion algorithms and the querying language such that, simply by defining an input schema, any possible model data structure may be converted to an equivalent edge-labelled digraph representation and manipulated via bidirectional graph transformations. In particular, we hope to draw upon concurrent work regarding a schema-based formalism for high-level queries upon a low-level digraph data structure [86] in translating more complex queries from high-level views to low-level graphs.

Given underlying improvements in the functional bidirectional transformations, particularly where the complexity of `SELECT` operations is concerned, we hope to then use complex queries in UnQL as a basis for more efficient merge and partition algorithms upon a biological model. The aim of this functionality would be to aid modularisation of large complex models (for example the HOG web), in which collaborators may wish to isolate a component of the system and analyse it in isolation. For example, we might use a combination of `SELECT` operations to split a single complex model into multiple constituent parts, make tractable mod-

ifications on an easier-to-visualise component, and then bidirectionally reflect these changes back to an original source. Alternately, we might merge two smaller models into a larger whole by defining the overlap in the `SELECT` query. The advantage of using bidirectional transformations in this context would be in the guaranteed correctness of the modifications as reflected to the original source.

Such a collaborative framework might also eventually form the basis for the distributed development of a model data structure stored in a version-controlled repository. It can provide multiple user-oriented views on the underlying knowledge base via different versions of the model code and the visualisations defined in Chapter 5. It can also provide the means for associating specific parameters or rules with academic sources such as journal publications and bioinformatics databases, thus providing traceability when these sources are updated over time including the possibility of automatically reflecting these changes. We can see both of these points being a major advantage to a collaborative distributed development effort, with each individual user able to contribute his or her knowledge in the manner (view) that they feel most comfortable with.

7.2 Visualising Large and Complex Webs of Data

In Chapter 5 we introduced model visualisations to aid the presentation and interpretation of large complex webs of agents and interactions. In Chapter 6 we extended these to cater for models based on standardised synthetic genetic parts. In both cases we aimed to provide an alternate model representation for users who may be uncomfortable with impenetrable blocks of code, and we believe we have succeeded in creating an unambiguous visualisation of the model amenable to inspection and understanding.

Also in Chapter 5 we identified a number of shortcomings in our representations, lessons learnt from visualising the high-osmolarity glycerol web. In particular we wished to develop better methodologies for automating the generation of the visualisations, for visualising large models with multiple interconnected components, and for displaying dynamic information on the static visualisation. Such functionality would greatly enhance the utility of the visualisations in displaying model information in a concise, unambiguous manner.

In this section we propose further extensions for the visualisations to address these shortcomings, demonstrating their benefits on a variety of biological systems from Chapter 2.

7.2.1 Automating the Visualisation Process

We begin with perhaps one of the simplest models of interest that we can describe using this visualisation formalism: the cooperative assembly model with two agents `A(a1,a2,a3)` and `B(b1,b2)`, such that each site of `A` may bind to and unbind from each site of `B`. None of

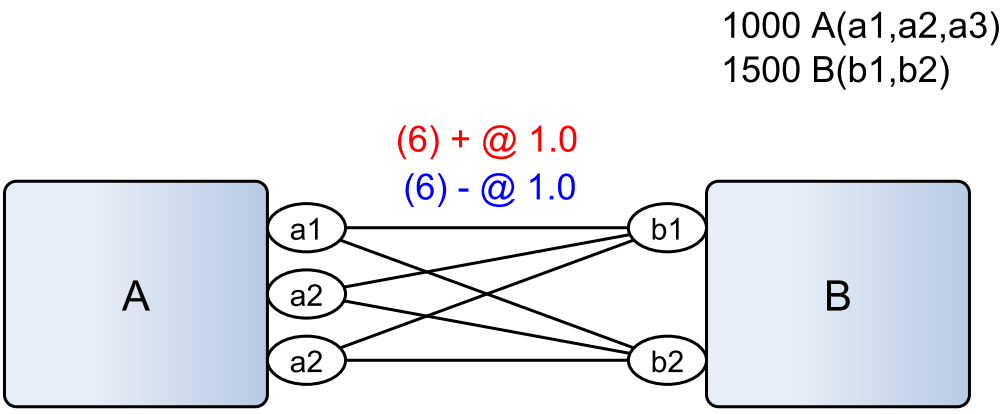


Figure 82: Visualising a simple cooperative assembly model with two agents `A` and `B`. `A` has three sites and `B` has two, and each site of `A` may independently bind to and unbind from each site of `B` at a nominal rate. Note the shorthand notation for labelling each of the six edges; this is possible since each rule is independent of every other, but with equivalent notation for precondition and effect.

these rules have preconditions, and given that we are interested in the structure of the finished product rather than any dynamics, the rate is defined at an equivalent value across all rules in the model. We visualise this model in Figure 82, in which we also adopt the convention of hiding explicit rule labels: since all rules are composed of only one basic effect, we have no need to amalgamate multiple effects into a single rule using such labels.

The first issue we encounter is the obvious difficulty of labelling overlapping and heavily interconnected edges. If we were to visualise the appropriate rules on each individual edge in Figure 82, we would easily double or triple the size of the visualisation. Here we take the solution of condensing the labels into a single line each for Binding and Unbinding, which we are able to do because every rule is a simple effect with no precondition beyond the obvious and the same rate. For a larger and more complex model, we can already see that this will be a major problem for current representations without the ability to provide “overview first, zoom and filter, then details-on-demand” [10].

Secondly, we address the issue of automating the generation of such visualisations. We have touched upon this previously, in that we currently lack the correct tool support (data formats and layout algorithms) to replicate human modelling expertise in laying out visualisations. In the case of a simple model such as the cooperative assembly model this is not as significant a problem, since the rules contain no inherent information (unlike for example the flow of a signalling cascade) and it makes sense to place the two agents side by side and give them equal prominence. In the case of models including MAPK cascades, including the HOG web

of Chapter 5, we might wish instead for a user to manually position the entities of the model (its Compartments and Agents), and for an automated algorithm to generate and label the edges between these entities. This methodology allows us to include expert information in the layout but also reduces the margin for human error by automating the most mistake-prone procedure (the transcription of rules into the visualisation).

7.2.2 Contact Graph Hierarchies

In Chapter 5, we introduce the high-osmolarity glycerol web in terms of eight interlinked chapters. Each chapter contains a set of rules representing a functional component of the system, and is implicitly connected to its neighbours via shared agents. These chapters and connections are represented in an abstract overview of the HOG web, which allows us to gain a picture of the overall system before we delve deeper into each chapter to explore the dynamics of the model. We believe that the high-level overview provided by these chapters is a useful abstraction by which to implement the first of the "overview first, zoom and filter, then details-on-demand" ideals posited by Cheng et al [10].

We thus propose an interface in which we hierarchically link the chapter overview to their constituent contact graphs. If the modeller has designated the division of a large body of rules into modular chapters (as in the example of the HOG web) this is a trivial task, as we can simply split the rules as specified. If not, then we must determine an appropriate methodology for automating this split: although we have yet to devise an approach that works across all models, we might for example choose to modularise across a major source of complexity in large models, the compartments.

For example, let us examine the synthetic switch introduced in Chapter 2, in which selectively inducible inputs specifically promote the reduction of `NPR1` to its monomeric form and create a unidirectional 'on' switch with the standard luciferase reporter gene as the circuit output. Again we ignore rule names for ease of representation, except for the oligomerisation and de-oligomerisation of `NPR1`, which combine four Binding and Unbinding effects respectively into a single rule. We visualise this model in Figure 83.

The rules of the nitrosylase induction model translate particularly well to visualisation, primarily due to the simplicity of the transcription factor activity and the lack of feedback from the circuit output to the input. Furthermore, we can easily modularise the visualisation according to compartment: the cytosolic interactions including induction response and luciferase feedback, and the nucleic interactions detailing transcription. Of course this is not the only possible modularisation, or even the optimal (as we probably would wish to separate the response from the feedback), but it is one that corresponds to an intuitive split of the system that makes sense from an overview representation as well.

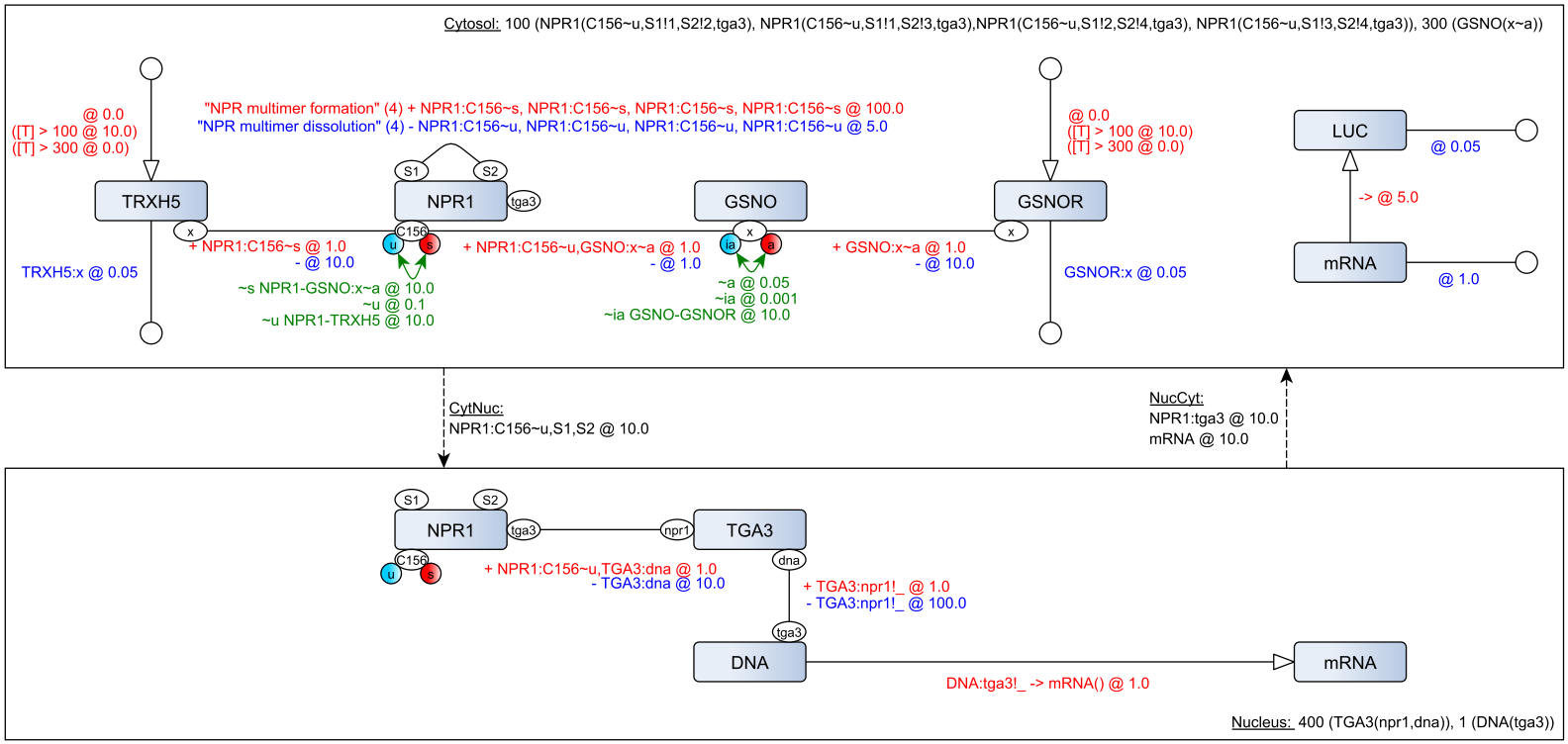


Figure 83: Visualising the synthetic switch. Note that we abbreviate the 'loc' site in this representation in favour of displaying the compartments explicitly, with transport between compartments indicated by a dotted arrow. Inducible inputs in the form of TRXh5 and the GSNO reductase GSNOR specifically promote the reduction of NPR1 to its monomeric form, thus producing luciferase as the circuit output. The system returns gradually to the pre-induction state as the inducible inputs filter out.

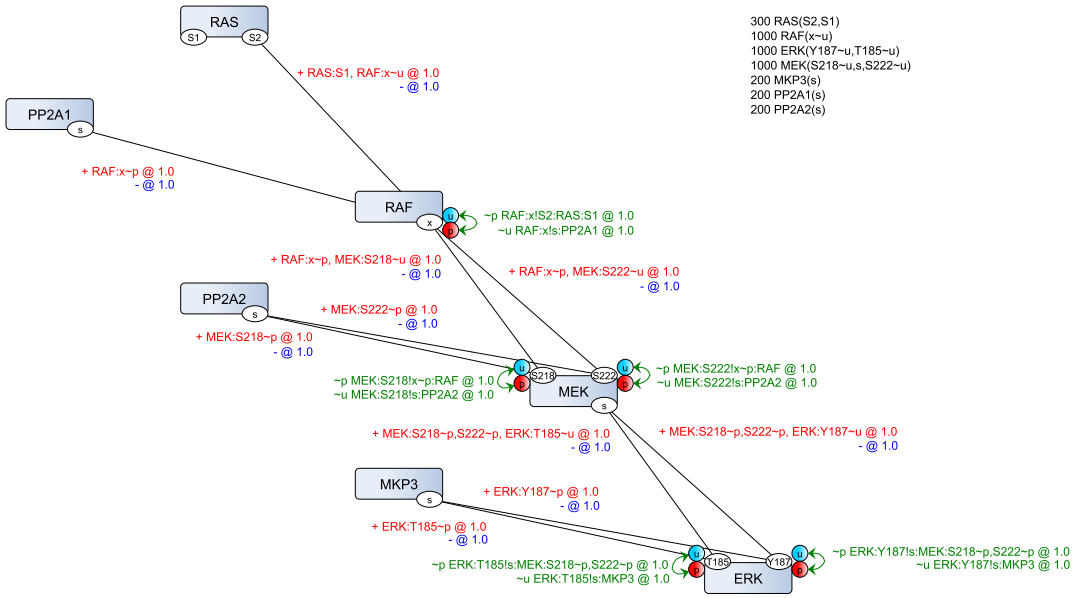


Figure 84: Visualising a simple mitogen-activated protein kinase cascade. The GTPase `RAS` activates the MAPKKK `RAF`, which in turn activates the MAPKK `MEK`, finally resulting in activation of the MAPK `ERK`; each intermediate also has a dedicated phosphatase (in order of appearance `PP2A1`, `PP2A2`, and `MKP3`). The cascade allows a small level of input signal from the top of the visualisation to quickly trigger a large response at the bottom.

7.2.3 Displaying Dynamic Stories on Static Visualisations

The final issue that we wish to address in our current iteration of model visualisations is the display of rule dynamics.

Our next visualised example is the simple MAPK cascade from [21] in Figure 84. Again we must use the explicit notation for depicting rule preconditions, since there are multiple ways by which `MEK-RAF`, `MEK-PP2A2`, `ERK-MEK`, and `ERK-MKP3` binding may occur. We must explicitly state this for each phosphorylation or de-phosphorylation rule, or we run the risk of ambiguity as to which binding is the precondition for the modification.

We subtly introduce the flow of information through the system by manually placing the MAPKKKs to the top of the visualisation and MAPKs at the bottom, which is an example of user input to the visualisation that would not be possible using automated layout algorithms without in-depth analysis of the modelled system.

We propose that we may represent this information flow explicitly as well, based on Figure 84. We do this by presenting the output of Kappa stories (a flowchart-like sequence of rules as applied to an initial state) on the visualisation.

As introduced in Chapter 1, Kappa stories perform rule-centric static analysis on the system, tracking a sequence of events that lead to a particular event of interest. For example, we

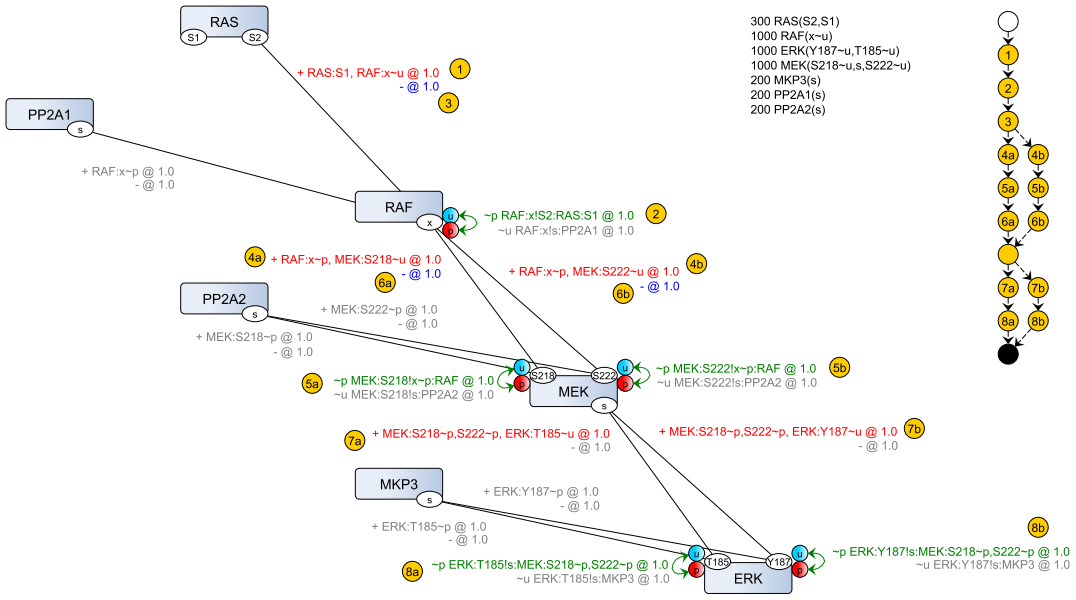


Figure 85: Visualising the flow of information through the simple mitogen-activated protein kinase cascade via a Kappa story. The story itself provides the sequence of rules that lead to a desired conclusion, in the form of a flowchart as seen newly added to the right of the visualisation. We might simply leave them annotated with the rule labels, or instead as shown label the effects in the visualisation.

might ask of the MAPK cascade model what rule sequence leads to the double phosphorylation of the MAPK `ERK`. We might then visualise the results as in Figure 85. An alternative output, suitable for presentation purposes, might then involve animating this flow of information one stage at a time.

Similarly, we might visualise stories on the model of the mammalian circadian clock. Here we introduce feedback to the model instead of a simple unidirectional flow. Our Kappa model of the mammalian circadian clock is based on an oscillating negative feedback loop, in which the `PER1` and `PER2` PERIOD proteins transport the `CRY1` and `CRY2` CRYPTOCHROME proteins into the nucleus of the cell, where they then inhibit their own production. A second negative feedback loop, centred on the `REV-ERB`α protein, reinforces this oscillation. We see these feedback loops, as well as the light perturbations inducing the transcription of `PER1` and `PER2` independently of the transcription factors upon the genes, in Figure 86.

We find that the genetic preconditions in this model are long and difficult to visualise, as can often be the case in synthetic biology as well. Partly this is caused by the fact that we cannot use the implicit notation for rule precondition labels, since there are multiple and ambiguous paths of binding from the `GENE` agent to its attached transcription factors that control its activity. We revisit this problem in the synthetic biology context in Chapter 8. We also find that the initial conditions, derived from a steady-state snapshot of the deterministic model

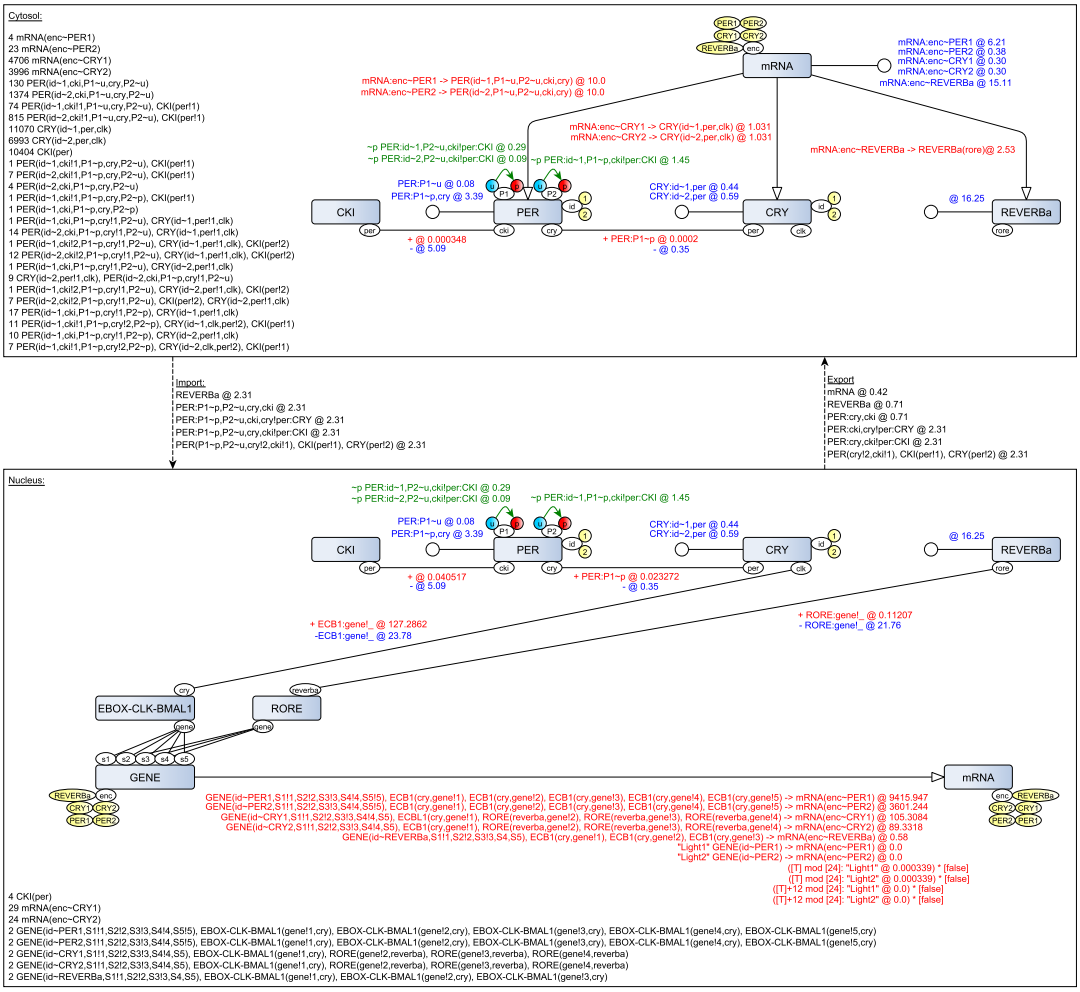


Figure 86: Visualising the Kappa model of the mammalian circadian clock. The PER and CRY proteins form one oscillating negative feedback loop, reinforced by a second centred on REV-ERBα as well as light perturbations that act directly upon the genes.

upon which this rule-based variant is based, are also extremely verbose due to the large genetic complexes involved. We believe that this emphasises the need for a layered interactive visualisation – "overview first, zoom and filter, then details-on-demand" – so that users may filter out unnecessary information and focus only on what they require.

On the other hand, we see that stories as shown on the MAPK model translate well even given the circular nature of the information flow in the mammalian circadian clock model. For example, let us seek the steps necessary for nuclear localisation of `PER1` unbound to `CRY`, assuming the initial state of the model consists of only genes and kinases. We show this particular story in Figure 87.

7.3 Bridging Bioinformatics with Modelling

We finish our discourse on how we might combat the growing temporal complexity in rule-based modelling by focusing on how we might improve the Kappa BioBrick Framework for synthetic gene parts. We discuss here how we could automate updates of the model as the underlying knowledge evolves from collaboratively curated databases. We also discuss how we might support modular components and their testing, such as we desired in the model of synthetic light communication, and how we might test synthetic circuits created in the KBBF in a 'whole cell' virtual chassis, allowing us to question whether the model can approximate or predict the behaviour of the circuit in its physical host. By improving traceability of the underlying data, and by providing support for asking interesting and verifiable questions of the model as it compares *in vivo* with *in silico*, we hope to ease the task of the modeller in tackling complex problems in synthetic biology.

7.3.1 Integrating the Kappa BioBrick Framework with a Virtual Parts Repository

As we have noted in Chapter 6, modelling in synthetic biology is one aspect of the current rule-based paradigm in which we are able to provide the basis of a full workflow from the underlying biological knowledge to a functional rule-based model, by utilising the Kappa BioBrick Framework. This can be achieved by noticing that the rates and interactions involved in the genetic elements of such a model are highly structured, and that their inclusion may be automated from a suitable source. What knowledge would we require, however, and where would we obtain it?

To answer the first question, we need only look at the inputs to the KBBF. Given a suitably detailed database, such as an ideally populated Registry of Standard Biological Parts, we might be interested in automatically defining promoter structure (repressor or activator, number of transcription factor binding sites and their cooperativity if any), rate information (transcription factor affinity, RBS and terminator efficiency, PoPS, and RiPS), and interactions of the protein

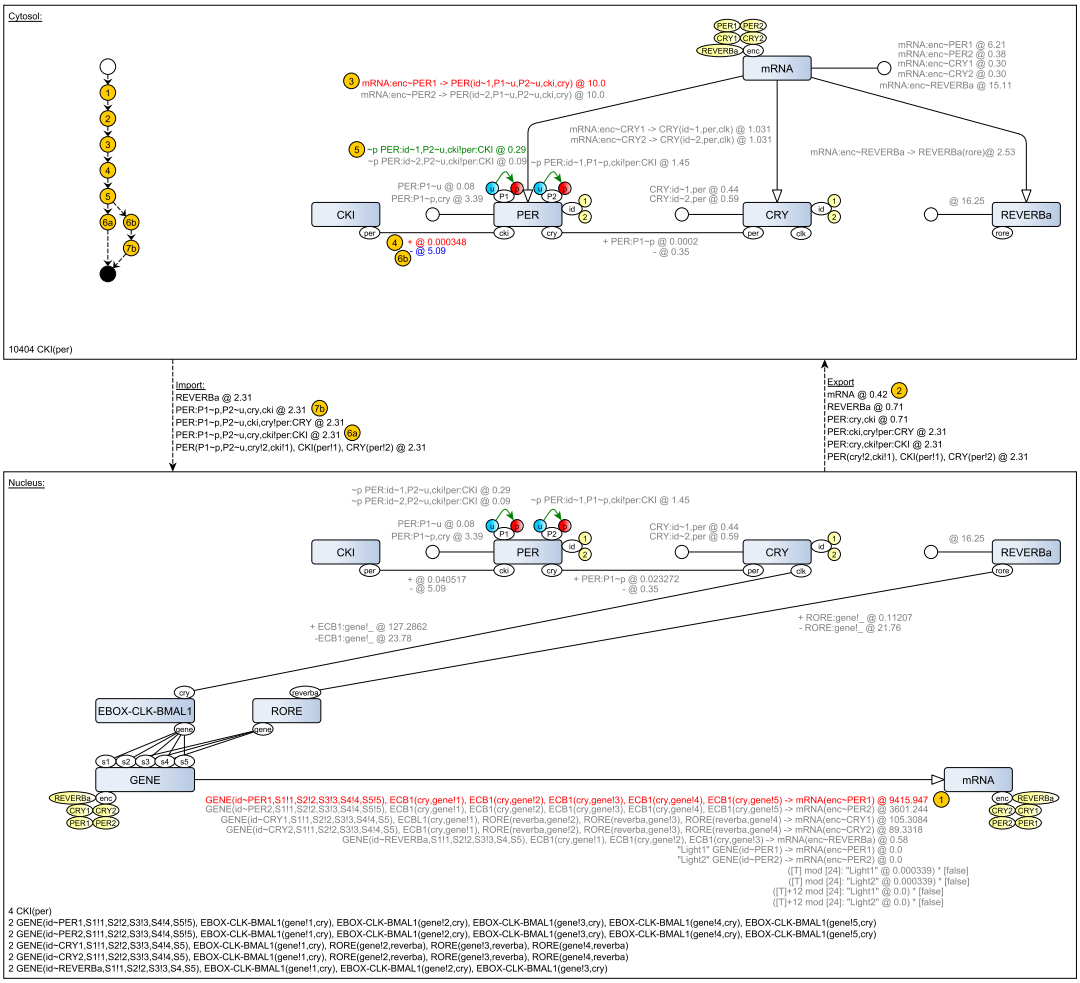


Figure 87: Visualising the flow of information through the mammalian circadian clock via a Kappa story; to do this, we remove the non-`CKI` proteins from the model and observe the rule applications as they are transcribed, translated, interact, and localise to the `Nucleus`. Even though this is a feedback system as opposed to the unidirectional information flow in the previous story in Figure 85, there are no discernible difficulties in displaying the sequence of events.

or domain associated with a particular coding sequence. This would reduce the burden on the modeller of finding and verifying individual rates related to the parts chosen for the model, and would also open up access to a traceable, community-curated source of model data. Of course, should the modeller so desire, they would still be able to design models or components from scratch, without being tied to such a resource.

To answer the second we turn to the Virtual Parts Repository (`http://www.virtualparts.org`), designed and maintained by Goksel Misirli, Anil Wipat, and Jennifer Hallinan of Newcastle University, which has the potential to grow into one such database capable of supplying such information to the KBBF. The Virtual Parts Repository is a catalogue of biological parts provided as computational models that can be composed to create genetic circuits. Standard virtual parts (SVPs) are provided for promoters, ribosome binding sites, terminators, shims, and coding sequences. Standardised inputs and outputs, based on widely-accepted biological signals such as PoPS and RiPS, allow larger systems to be constructed from basic biological parts analogous to how the KBBF chains together BioBrick parts to create a device. SVPs are associated both with reactions that are not dependent on other parts, such as degradation, as well as interactions with other parts that must be added to create simulable models. This provides a structured way of providing detailed protein interactions outside the scope of the KBBF.

We have thus collaborated with the Virtual Parts Repository to ensure that it provides a KBBF representation output of a synthetic genetic part (as described in Chatper 6) alongside other accepted formats such as SBML and SBOL. Providing for multiple representations of a SVP in this manner allows the Virtual Parts Repository to maintain various sets of information about a part that would be useful for different purposes; for example, some users may simply be interested in the genetic sequence of the part, its aliases, or a set of references from which to explore further information. Others, with greater interest in the detailed workings of the part and looking to build a mathematical model, may turn to the KBBF.

The current implementation, however, is very much in its infant stage. As yet, we have few verified parts in the repository and no examples of models having been created from them. This would be the next step of the repository – to encourage users (for example, future iGEM teams) to make use of the tool, and to work through the complications that will undoubtedly arise. Current and future efforts will continue to collaborate with Goksel et al. to populate the repository and to refine the modelling processes making use of it.

7.3.2 The Module Integration Simulator, *mois*

We now move on to discuss how we might provide further support for modular development and testing in synthetic biology, as well as how we might model a virtual chassis in the whole-cell paradigm.

The Module Integration Simulator, or *mois*, is software currently under development by William Waites of the School of Informatics, University of Edinburgh [91]. The objective of *mois* is to construct models of coupled dynamic systems, building on previous work in the field by Dominik Bucher. The premise is that models may be assembled from separate processes: encapsulated modules with defined function and input/output characteristics as part of a larger whole. Given these modules, which may be written in completely incompatible paradigms and languages, *mois* attempts to provide a standardised means for coupling and integrating them so as to calculate the trajectory of a coupled dynamic system. In other words, it marries modularisation techniques from software engineering with modelling for complex biological systems, thus providing a means for models developed by scientists proficient in particular languages or modules to be integrated into a coherent whole.

The immediate conceptual motivation for mois is the whole-cell model by Karr and Covert [49]. This describes the simulation of a single *Mycoplasma genitalium* cell life cycle in 28 independent processes, modelled as a mixture of ODEs, flux-balance optimisation, stochastic processes, and simple mathematical equations. The Karr model is composed of 80000 lines of Matlab code (+ 44000 lines of tests) that, while well documented and annotated via associated literature, is correspondingly complicated and difficult to reason about. In essence, we are faced with the need to understand the code-base (written in a language not designed for such things, without a view to being reusable for other problems) before we can understand the science underlying the model.

mois attempts to overcome the shortcomings of the Karr concept via:

- Modularisation: processes should be independently runnable and testable, such that it is possible to reason about them without extensive knowledge of the framework or of other processes.
- Statically Typing: scaling to large collaborative projects by removing a major source of error expressions and function calls that are ill-typed.
- Natural Syntax: making propositions within the modules as clear and as understandable (as close to the style of writing and notation commonly used within the literature) as possible.
- Variable Correspondence: integrating a mechanism for specifying a canonical identifier for any given quantity regardless of the name used in the process itself.
- Configurable Integration: supporting modular integrator development (parallel to modular process development), thus approaching the problem of how to solve the trade-off between the parallelism resulting from decoupling processes versus the accuracy gained by mutual interaction with tight coupling.

The software is published freely on GitHub at `https://edinburgh-rbm.github.io/`. For now it supports the modelling of modules composed of systems of ordinary differential equations and reaction networks (both stochastic and deterministic).

7.3.3 Designing and Testing Component Modules in *mois*

We aided in the development of the Module Integration Simulator via the creation, analysis, and verification of models in the framework, and by proposing functionality based on this modelling experience. In particular we focused on the modularisation of the Forger-Peskin mammalian circadian clock model underlying its Kappa counterpart introduced in Chapter 2, primarily for the reason that we already had experience working with it across multiple formats and thus had a concrete base from which to test the different modules in *mois*. Although this is not explicitly a model of synthetic biology, we extrapolate the lessons learnt to all modular models including those created in the Kappa BioBrick Framework.

As described previously, the Forger-Peskin model is based on an oscillating negative feedback loop, in which the PER1 and PER2 PERIOD proteins transport the CRY1 and CRY2 CRYPTOCHROME proteins into the nucleus of the simulated cell. A second negative feedback loop, centred on the REV-ERBa protein, reinforces this oscillation. The transcription factor complexes inhibit the production of their constituent proteins via standard transcription and translation processes. Finally, light perturbations induce the transcription of PER1 and PER2 independently of the transcription factors upon the genes, reinforcing the 'reset' of the clock. For now we utilise our meta-knowledge as a modeller to modularise the Forger-Peskin model, consisting of 73 equations on 75 species, as described above. We thus implement these four modules – the two feedback loops, the genetic response, and the light perturbations – in *mois* both as systems of ODEs and as derived deterministic reaction networks (thus allowing us to compare their implementations), and compare them against a basal monolithic model duplicating the Forger-Peskin implementation.

Our initial results are positive, in that we observe the exact same sustained 24-hour oscillations across both monolithic and modular implementations, thus proving that simulating modularised models need not have an adverse effect on the model dynamics. On the other hand, we observe aberrant behaviour when we vary the timestep at which the various modules communicate with one another, confirming that there exists an implicit dependence of a modular model on how often its constituent parts are allowed to communicate. Further studies hope to research the exact nature of this dependence, as well as whether we can determine an optimal timestep at which modules should communicate to minimise computational load but maximise simulation accuracy. We would also like to be able to test our Kappa implementation of the mammalian circadian clock against a stochastic reaction network, with the aim of further strengthening the modelling suite available to *mois*; we believe this will eventually be possible

by extending Ricardo Honorato-Zimmer's LMS-Kappa into the framework.

In addition to modular simulation as detailed above, a major advantage of *mois* lies in its ability to systematically test the individual modules and verify their activity. This follows a similar process to unit testing in software engineering, in which we write an accompanying test class to accompany the main module and may run these tests in *mois* independently of the simulator. Attributes of the model to be tested might include violation of invariants, reachability analysis (whether or not a system attains a given state), and comparison of time courses against a baseline. In the context of the mammalian circadian clock, we might wish to automate tests as to whether or not the individual oscillations ever stop, whether they are stable, and whether or not the modular variants of the model exactly match the monolithic baseline.

We are able to develop and run such tests for full simulations of the model, but experience issues when attempting to test each module in isolation. In particular we find it a non-trivial task to input to a test case simulation parameters that are ordinarily shared and generated over multiple modules (for example, a dynamic time series that depends on the activity of other components); we have yet to develop and understand good working practices to simulate to take best advantage of the capabilities offered by such a framework. We also encounter cases when we wish a module to adopt one set of behaviours in isolation but a second when cooperating with other modules, but this is not yet catered for in the *mois* implementation. It is our hope that further work in this area will elucidate solutions to these problems.

7.3.4 Whole Cell Chassis in *mois*

Whole-cell models as exemplified by Karr et al. [49] combine multiple modelling approaches, such as ordinary differential equations and flux balance analysis, to integrate all of a cell's molecular components and their interactions in a single computational object. As applied to synthetic biology, we might propose applying such whole-cell models as a virtual chassis, which would allow us to test *in silico* how proposed BioBrick devices (modelled for example using the Kappa BioBrick Framework) affect the physiology of the host. A grander vision would then be to have a selection of not only different chassis (for example, yeast (*Saccharomyces cerevisiae*), *Escherichia coli*, *Bacillus subtilis*, variants thereof, and so forth), but also different versions of chassis components (for example, signalling pathways) that would allow potential modellers to customise their model via plug and play.

Another application of the whole-cell modelling paradigm would be to test the Kappa BioBrick Framework's ability to account for resources (RNA polymerases and ribosomes) shared with other processes extant within the host. As stated throughout Chapter 6, Kappa allows for the refined modelling of transcriptional and translational activity, for instance facilitated search by transcription factors and multiple ribosomes upon mRNA. In a whole-cell model similar to that implemented by Karr et al., we might on the one hand use other modelling techniques such

as flux balance analysis for metabolism, and delegate the interactions of DNA and RNA to a high resolution Kappa model. This would allow us to apply the best modelling techniques to their most appropriate usage, and tie them all together in a coherent whole.

In contrast, current efforts spearheaded by Andrea Weisse [93] and Guillaume Terradot focus on the development of simpler, more focused models related to a single aspect of bacterial physiology (for example, cell life cycle or cell-wide RNAP usage). In particular, we have explored in *mois* a mechanistic model of an idealised bacterial cell, combining nutrient transport and metabolism with the biosynthetic processes of transcription and translation, developed by the former. Concretely, it consists of a system of ODEs with 14 variables, including nutrient, energy, four proteins (ribosome, transporter, metabolic, house-keeping) and their mRNA (in free and ribosome-bound forms). The growth rate of the cell is defined as a function of the number of translating ribosomes and energy, and given a steady source of nutrients eventually arrives at a balancing equilibrium.

The main assumptions made in the model are as follows:

- Finite energy, finite ribosomes, and finite proteome.
- No degradation of proteins (only dilution) and first-order degradation of mRNA.
- Mass action kinetics for binding and unbinding of mRNA with free ribosomes.
- Energy consumption via transcription is negligible, thus we are free to concentrate on consumption via translation only.

We translate the ODE model (defined in Matlab) into four *mois* modules: nutrient import and metabolism, translation, transcription, and growth / dilution. For nutrient import and metabolism, the model assumes Michaelis-Menten kinetics upon a constant external nutrient (constant environment). Translation assumes stepwise elongation of the peptide chain upon an mRNA-ribosome complex to derive the dependence of the translation rate on the energy levels of the cell; transcription assumes a similar mechanism with energy dependence, but unlike translation does not actually consume any energy from the system. Finally, the growth rate dilutes all intracellular species due to the redistribution of cellular content to mother and daughter cells. At steady state, growth is proportional to the rate of protein synthesis.

We then follow the lead of Weisse et al. [93] in comparing a standard repressilator, isolated from any cellular processes (as defined in Chapter 6) with a host-aware repressilator in which the translation of mRNA is tied to the ribosomes available from the whole-cell model. Although we have yet to study this system in detail, we may point to this modelling as a proof of concept as to using simple whole-cell models to track the effects of synthetic genetic circuits upon measurements of cell growth. In time we might advance to embedding circuits defined in the KBBF into such cell models, observing how they interact with the physiology of their host.

We may require extensions to future iterations of the KBBF and its visualisations to account for the host physiology under consideration, similar to the models for RNA polymerases and ribosomes already included. The advantage of such models would hopefully lie in allowing the modeller to adapt his or her synthetic circuit to the underlying organism *in silico* with only a modicum of further effort.

Chapter 8

Conclusions

In this chapter we...

- ❖ Review the contributions we have made in this thesis.
- ❖ Identify the deficiencies in our contributions.
- ❖ Consider future research across all fronts to address these deficiencies.

In this final chapter we summarise the accomplishments of this thesis with respect to the questions raised in Chapter 1, and point to areas that require further work.

8.1 Evaluation

In the introduction to this thesis, we motivate our work by identifying the need to focus further on complex interconnected and evolving 'webs' of biological information in addition to more traditional 'pathways', to better understand how biological interactions and associated experimental data may be reflected *in silico*. We ask the following questions based on existing modelling approaches in Kappa and Spatial Kappa:

- How can we more intuitively define a spatial projection (compartments and channels) upon a set of rules, agents, initial conditions, and perturbations, allowing us to write a model in space that can be verified against established results?
- Can we visualise the data structures and dynamics inherent to a rule-based model in either Kappa or Spatial Kappa, and utilise this visualisation in a graphical definition of models generated from a specification of synthetic parts?

We also bear in mind throughout this work how we might bridge bioinformatics and modelling, by providing traceable links and methodologies between the elements of a model and the underlying data or literature sources.

Our first question is further motivated by biological models of the mammalian circadian clock and the high osmolarity glycerol web in yeast, both of which rely on the spatial separation of interacting agents to convey a signal through the system. We investigate the current implementation of Spatial Kappa, by generating a test suite of models to analyse against existing theory in the form of the Stokes-Einstein relation and asymptotic solutions to the narrow escape problem. We find that although we ably approximate the former for one, two, and three-dimensional shapes, we are unable to reliably duplicate results for the latter. We raise questions both regarding the opacity of the simulator (i.e. the 'jumping' problem) and the representation of space as a Cartesian grid implemented as integral to the language. We then propose alternative definitions of space, each with advantages adapted to different modelling situations, in an effort to make complex modelling in space more intuitive to the user. We also look at further concepts such as limiting voxel occupancy, automatically deriving translocation rates, and the procedural generation of space via mathematical means, providing sample use cases for each.

Our second question is motivated by model development of the high osmolarity glycerol web and models in synthetic biology, in which we have experienced iterative development based on feedback from model results. We define a theoretical data structure for rule-based models in Kappa and Spatial Kappa, incorporating both entities and rules, and provide a methodology for visualising it that we test via the high osmolarity glycerol web model. We further optimise these visualisations for the definition of models in synthetic biology making use of reusable genetic parts, and provide a sample case study for this based on the 2010 University of Edinburgh IGEM project and its underlying repressilator system. Finally, we begin to address lessons learnt from these modelling efforts, including: the idea of a framework to aid model development and a workflow taking advantage of it; the automated visualisation of large and complex webs of data, dynamic stories on static contact graphs, and hierarchical linking of overviews to visualisations; the basis of a workflow in synthetic biology from the underlying biological knowledge to a functional rule-based model, utilising the Kappa BioBrick Framework; and the creation and testing of model components in the Module Integration Simulator.

However, the work presented in this thesis is deficient in many ways. Most glaringly, we lack a solid implementation for many of the features discussed, from our proposals on spatial re-implementations to our ideas for an integrated model development environment. We have yet to satisfactorily pinpoint the exact cause of the 'jumping' problem in space as a Cartesian grid, or to provide an exact reason why the asymptotic formulae for narrow escape do not apply. Furthermore, we would wish to run further tests on alternative definitions of rule-based space as well, to see whether we can solve the problems encountered in the lattice-like definition of space by changing the level of abstraction upon which we work. We also have more work to do regarding answering the open question of how we might automate the derivation of model space from experimental data, although we feel that defining space as a Kappa-like complex

and making better use of macro-observables including geometric and biophysical attributes may help alleviate this problem.

With regards to our proposed integrated model development environment, our visualisations lack conciseness at their lowest and most verbose level, as evidenced by the size of some of the graphics within this thesis. We also have yet to find a satisfactory solution to the use of Kappa-like syntax in the graphical representation (ideally it would be independent of the choice of language for the underlying rule-based model), or to alleviate the practical issues surrounding how we might present this in an editable form to the user (including hierarchical overlays, automation, and the display of dynamic stories). In addition, we have failed to outline a satisfactory machine-assisted workflow from bioinformatics to modelling – for example, we have left as an open question the exact methodology by which we might automate the derivation of the spatial structure of a model, or how we might optimally modularise it in preparation for implementation in the Module Integration Simulator. Ideally, following the clarification of these questions we would actually implement the development of a collaborative biological model in this framework as well, and iteratively develop it based not only on model results but also on experimental results as well.

We describe future work below that attempts to address these issues.

8.2 Future Efforts

The central goal of future efforts is to verify and refine the concepts introduced in this thesis whilst working towards a solid implementation. Our eventual aim is to establish a smooth workflow that will allow users to specify and modify, in an intuitive and iterative manner, a rule-based biological model embedded in a well-understood notion of space with traceable links to the underlying bioinformatics data. By enshrining this workflow in an integrated model development environment, we then hope to make it accessible to a wide range of users with varying computational and biological backgrounds.

8.2.1 Complex Models in Space

Space as a Cartesian Grid: The strength of the current implementation of space is how it discretises homogeneous space to allow for non-homogeneous behaviour. The voxel occupancy extension introduced in Chapter 4.1.1 is a good example of how to take further advantage of this, for models such as calcium diffusion in the synapse. However, we would need to plan such a model carefully to compensate for the diffusive inaccuracies discovered in Chapter 3. Our hope here would be to explore a methodology for working in such a definition of space so as to minimise the effect of the rigid assumptions made. Improving the efficiency of the simulator, whether by introducing parallelism or by refining the algorithms used, is also a priority. Finally,

we still need to isolate and rectify the exact cause of the 'jumping' problem, such that it does not undermine user confidence in Spatial Kappa. We would wish to prioritise future work in this direction, as the 'jumping' problem in particular requires prompt attention.

Space as a Kappa-like Complex: Our initial aim regarding space as a Kappa-like complex is to implement it based on the provisional operational semantics provided, and to verify it in the same manner as we have done for space as a Cartesian grid. One strength of such a representation lies in the abstract simplicity of the definition while still maintaining biological relevance via geometric and biophysical attributes and the automated inference of model parameters. Another strength is the fact that we can grow, shrink, and otherwise modify the spatial structure in the same way as we might a standard Kappa complex. We wish to develop use cases (based on models introduced in Chapter 2) to strengthen the case for this representation, for example a variant of the mammalian circadian clock (for which we can compare to previous versions) or growth of the cooperative assembly of a bacterial chemoreceptor. We would also like to further explore the automated derivation of spatial structures in this form, as well as their evolution throughout the course of a set of experimental observations. The advantage of pursuing this work lies in broadening the user definitions for Kappa models incorporating spatial definitions, but given that cBNGL provides similar functionality it is not a priority.

Space as a Cayley Graph: Similar to space as a Kappa-like complex, we wish to implement space as a Cayley graph based on our provisional operational semantics and to verify it against existing formulae. We then would work towards demonstrating the strength of the representation – the procedural generation of space via rigorous mathematics – via similar use cases. Again similar to future work in defining space as a Kappa-like complex, we believe that we should not prioritise this at the expense of verifying the current implementation of Spatial Kappa.

8.2.2 Complex Models over Time

Bridging Modelling and Bioinformatics: Consider an *ad hoc* process for mechanistic biological modelling, in which we construct an informal model representing the domain expert knowledge of the system of interest, then simplify this via a set of undocumented abstractions to an executable model (for example of ODEs). Because we track neither the domain knowledge nor the abstractions, and because we make no provision for the rapid evolution of the underlying biological knowledge base, revisiting and reusing models is problematic to the point where the literature is cluttered with single-use fit-to-purpose derivations. Languages such as Kappa approach this problem by intuitively encapsulate more of the expert model, and model repositories such as rulebase.org are a first step towards specifying the underlying knowledge and assumptions. But we have yet to design a development workflow that (partly) automates the derivation of knowledge from bioinformatics tools such as web-based databases into a rule-

based data structure, aside from a rudimentary and unverified ongoing effort in synthetic biology. We have also yet to fully understand the assumptions that underlie the collection of such data, that by necessity will bias the models under construction. Developing the methodologies to create such an interactive knowledge base would be invaluable in the rational construction, communication, and maintenance of a complex, realistic biological model. We believe that this is a goal worth pursuing due to the greater understanding of underlying biological mechanisms the community may thus obtain, though it may take many years of effort before research bears fruit.

Implementing an Integrated Model Development Environment: In this thesis we have laid out the backbone for the implementation for an integrated model development environment, including features such as multi-level visualisations ("overview first, zoom and filter, then details-on-demand") that correspond directly to the underlying model, updates based on model queries and bidirectional transformations, and an integrated version control system. We believe that such a framework would greatly lower the entry barrier to modelling in biology, and allow interesting new questions to be asked of the computational knowledge bases thus created. On the other hand we have yet to actually implement it, and would need to address further shortcomings we have identified. For example, we might attempt to reduce the verbosity of our visualisations by redefining our edge annotations. We would also need to come up with the best way to present this to the user in such a way that he or she is not overwhelmed by it all and can easily and intuitively make queries upon the information contained within. We have investigated such a workflow in this thesis via sample models, but wish to develop them into a full-fledged test suite to accompany the framework development. Again we believe that this is a goal worth pursuing immediately due to the benefits it provides to end users, as well as efforts already being made in this direction such as the Module Integration Simulator.

Extending to Synthetic Biology: We then might wish to extend this integrated model development environment with a specialised plug-in for modelling in synthetic biology based on the Kappa BioBrick Framework (KBBF). This would include linking to dedicated virtual parts repositories and integrating whole cell models to allow *in silico* testing of the synthetic device in a virtual chassis. Again, we have demonstrated the usefulness of the ideas in this thesis via sample models that we would like to develop further in the implementation. Given the close relation of this goal with the integrated model development environment described above, as well as implementations already in place and under development, we see this goal being achieved in the near future.

The Module Integration Simulator: The Module Integration Simulator *mois* is a useful tool for the construction of dynamically coupled systems. In particular, we believe that future applications include whole cell models and virtual chassis in synthetic biology, and the testing and verification of complex models divided into modular chapters. We wish to better

understand these applications, for example a workflow integrating modular testing into model development in such a way that multiple users can easily specify their expectations of the model assumptions. We also might wish to extend *mois*, both with rule-based modelling capability in Kappa, and with spatial functionality in simulations of modularly-defined cell models interacting across intercellular boundaries via quorum signalling or nutrient competition. Again, given implementations already in place and under development, we believe that the further development and refinement of *mois* will provide significant functionality to the community.

Developing Further Models: We believe that synthetic biology would be a good field to start identifying and developing further models for the integrated development environment, since it conforms intuitively to the iterative development paradigm. Systems constructed in this way are also easy to test and verify, as their behaviour can be controlled better than natural systems. Going beyond the reproduction of existing models, the development of novel models in collaboration with other researchers is likely also to reveal further practical problems, which we will need to address through future iterations of design and development. In particular, we have yet to iteratively develop models based on experimental feedback (as opposed to model feedback), and anticipate issues in clarifying this process. As we expect these to be exposed throughout development and publication of the many tools described previously, we feel that prioritising them as they occur remains the best approach.

8.3 In Closing

We began this thesis by identifying the need to focus further on complex interconnected and evolving 'webs' of biological information, to better understand how biological interactions and associated experimental data may be reflected *in silico*. The tangible artefacts of us approaching this need include our first attempts at identifying and solving problems raised by the development of the models presented in Chapter 2: verifying the current implementation of Spatial Kappa and proposing new definitions of space; providing methodologies for visualising and utilising Kappa models as knowledge bases and extending these ideas to synthetic biology via the Kappa BioBrick Framework; and thinking about the modularisation, iterative development, and testing of complex models in the Module Integration Simulator.

We now end by proposing further work in refining a development workflow from underlying experimental data to a computational model that can be shared between users as a biological knowledge base. We would like to prioritise further work on distilling bioinformatics knowledge into rule-based data structures, in particular making these data structures amenable to iterative refinement and visualisation, and on *mois* as a tool for the modular creation and testing of large models; we believe that, within a timescale of three to five years, these functionalities would have the greatest utility for end users of rule-based models.

The ideal eventual outcome of this body of work would be for this proposed integrated model development environment to pave the way towards asking questions that previously could not be asked, regarding complex rule-based models in space and over time. Much effort is required before the ideas presented here may even begin to help modellers understand complex biology. However, it is our humble opinion that we have made one small step forwards in this process.

Bibliography

[1] David Aldous. Hitting times for random walks on vertex-transitive graphs. In *Mathematical Proceedings of the Cambridge Philosophical Society*, volume 106, pages 179–191. Cambridge Univ Press, 1989.

[2] S.S. Andrews, N.J. Addy, R. Brent, and A.P. Arkin. Detailed simulations of cell biology with smoldyn 2.1. *PLoS Computational Biology*, 6(3):e1000705, 2010.

[3] J.A. Bachman and P. Sorger. New approaches to modeling complex biochemistry. *Nature Methods*, 8:130–131, 2011.

[4] J.W.S. Baron Rayleigh. The theory of sound, 1945.

[5] M.L. Blinov, J.R. Faeder, B. Goldstein, and W.S. Hlavacek. BioNetGen: software for rule-based modeling of signal transduction based on the interactions of molecular domains. *Bioinformatics*, 20(17):3289, 2004.

[6] P. Buneman, M. Fernandez, and D. Suciu. Unql: A query language and algebra for semistructured data based on structural recursion. *The VLDB JournalThe International Journal on Very Large Data Bases*, 9(1):76–110, 2000.

[7] L. Calzone, F. Fages, and S. Soliman. Biocham: an environment for modeling biological systems and formalizing experimental knowledge. *Bioinformatics*, 22(14):1805–1807, 2006.

[8] L. Cardelli. Brane calculi. In *Computational Methods in Systems Biology*, pages 257–278. Springer, 2005.

[9] D. Chandran, F.T. Bergmann, H.M. Sauro, et al. Tinkercell: Modular cad tool for synthetic biology. *Journal of Biological Engineering*, 3(1):19, 2009.

[10] H-C. Cheng, B.R. Angermann, F. Zhang, and M. Meier-Schellersheim. Networkviewer: visualizing biochemical reaction networks with embedded rendering of molecular interaction rules. *BMC Systems Biology*, 8(1):70, 2014.

[11] A.F. Cheviakov, M.J. Ward, and R. Straube. An asymptotic analysis of the mean first passage time for narrow escape problems: Part ii: The sphere. *Multiscale Modeling & Simulation*, 8(3):836–870, 2010.

[12] L.A. Chylek, B. Hu, M.L. Blinov, T. Emonet, J.R. Faeder, B. Goldstein, R.N. Gutenkunst, J.M. Haugh, T. Lipniacki, R.G. Posner, et al. Guidelines for visualizing and annotating rule-based models. *Molecular BioSystems*, 7(10):2779–2795, 2011.

[13] F. Ciocchetta and M.L. Guerriero. Modelling biological compartments in bio-pepa. *Electronic Notes in Theoretical Computer Science*, 227:77–95, 2009.

[14] F. Ciocchetta and J. Hillston. Bio-PEPA: an extension of the process algebra PEPA for biochemical networks. *Electronic Notes in Theoretical Computer Science*, 194(3):103–117, 2008.

[15] F. Ciocchetta and J. Hillston. Bio-PEPA: a framework for the modelling and analysis of biological systems. *Theoretical Computer Science*, 410(33-34):3065–3084, 2009.

[16] K. Czarnecki, J.N. Foster, Z. Hu, R. Lämmel, A. Schürr, and J.F. Terwilliger. Bidirectional transformations: A cross-discipline perspective. In *Theory and Practice of Model Transformations*, pages 260–283. Springer, 2009.

[17] K. Czarnecki and S. Helsen. Feature-based survey of model transformation approaches. *IBM Systems Journal*, 45(3):621–645, 2006.

[18] T.C. Damgaard, V. Danos, and J. Krivine. A language for the cell. Technical Report TR-2008-116, IT University of Copenhagen, 2008.

[19] T.C. Damgaard, E. HøJsgaard, and J. Krivine. Formal cellular machinery. *Electronic Notes in Theoretic Computer Science*, 284:55–74, 2012.

[20] T.C. Damgaard and J. Krivine. A generic language for biological systems based on bigraphs. Technical Report TR-2008-115, IT University of Copenhagen, 2008.

[21] V. Danos. Agile modelling of cellular signalling (invited paper). *Electronic Notes in Theoretical Computer Science*, 229(4):3–10, 2009.

[22] V. Danos, J. Feret, W. Fontana, and J. Krivine. Abstract interpretation of cellular signalling networks. In *Verification, Model Checking, and Abstract Interpretation*, pages 83–97. Springer, 2008.

[23] V. Danos, H. Koeppl, and J.R. Wilson-Kanamori. Cooperative assembly systems. *DNA Computing and Molecular Programming*, pages 1–20, 2011.

[24] V. Danos and C. Laneve. Core formal molecular biology. *Programming Languages and Systems*, pages 302–318, 2003.

[25] V. Danos and C. Laneve. Formal molecular biology. *Theoretical Computer Science*, 325(1):69–110, 2004.

[26] V. Danos and L.J. Schumacher. How liquid is biological signalling? *Theoretical Computer Science*, 410(11):1003–1012, 2009.

[27] J. Elf and M. Ehrenberg. Spontaneous separation of bi-stable biochemical systems into spatial domains of opposite phases. *Systems Biology*, 1(2):230–236, 2004.

[28] R.J. Ellis. Macromolecular crowding: Obvious but underappreciated. *Trends in Biochemical Sciences*, 26(10):597–604, 2001.

[29] M.B. Elowitz and S. Leibler. A synthetic oscillatory network of transcriptional regulators. *Nature*, 403(6767):335–338, 2000.

[30] J.M. Epstein. Why model? *Journal of Artificial Societies and Social Simulation*, 11(4):12, 2008.

[31] J.R. Faeder, M.L. Blinov, and W.S. Hlavacek. Rule-based modeling of biochemical systems with bionetgen. In *Systems Biology*, pages 113–167. Springer, 2009.

[32] D.B. Forger and C.S. Peskin. A detailed predictive model of the mammalian circadian clock. *Proceedings of the National Academy of Sciences*, 100(25):14806–14811, 2003.

[33] D.B. Forger and C.S. Peskin. Stochastic simulation of the mammalian circadian clock. *Proceedings of the National Academy of Sciences of the United States of America*, 102(2):321–324, 2005.

[34] J.N. Foster, M.B. Greenwald, J.T. Moore, B.C. Pierce, and A. Schmitt. Combinators for bidirectional tree transformations: A linguistic approach to the view-update problem. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 29(3):17, 2007.

[35] A. Funahashi, Y. Matsuoka, A. Jouraku, M. Morohashi, N. Kikuchi, and H. Kitano. Celldesigner 3.5: A versatile modeling tool for biochemical networks. *Proceedings of the IEEE*, 96(8):1254–1265, 2008.

[36] T.S. Gardner, C.R. Cantor, and J.J. Collins. Construction of a genetic toggle switch in escherichia coli. *Nature*, 403(6767):339–342, 2000.

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

[38] D.T. Gillespie. Stochastic simulation of chemical kinetics. *Annual Review of Physical Chemistry*, 58(1):35–55, 2007.

[39] D. Harel, Y. Setty, S. Efroni, N. Swerdlin, and I.R. Cohen. Concurrency in biological modeling: Behavior, execution and visualization. *Electronic Notes in Theoretical Computer Science*, 194(3):119–131, 2008.

[40] R. Harmer, V. Danos, J. Feret, J. Krivine, and W. Fontana. Intrinsic information carriers in combinatorial dynamical systems. *Chaos*, 20(3):037108, 2010.

[41] L.A. Harris, J.S. Hogg, and J.R. Faeder. Compartmental rule-based modeling of biochemical systems. In *Winter Simulation Conference*, pages 908–919. Winter Simulation Conference, 2009.

[42] J. Hattne, D. Fange, and J. Elf. Stochastic reaction-diffusion simulation with mesord. *Bioinformatics*, 21(12):2923–2924, 2005.

[43] S. Hidaka, Z. Hu, K. Inaba, H. Kato, K. Matsuda, and K. Nakano. Bidirectionalizing graph transformations. In *ACM Sigplan Notices*, volume 45, pages 205–216. ACM, 2010.

[44] S. Hidaka, Z. Hu, K. Inaba, H. Kato, and K. Nakano. Groundtram: An integrated framework for developing well-behaved bidirectional model transformations. In *Automated Software Engineering (ASE), 2011 26th IEEE/ACM International Conference on*, pages 480–483. IEEE, 2011.

[45] S. Hidaka, Z. Hu, H. Kato, and K. Nakano. A compositional approach to bidirectional model transformation. In *Software Engineering-Companion Volume, 2009. ICSE-Companion 2009. 31st International Conference on*, pages 235–238. IEEE, 2009.

[46] D. Holcman and Z. Schuss. Control of flux by narrow passages and hidden targets in cellular biology. *Reports on Progress in Physics*, 76(7):074601, 2013.

[47] C.Y. Huang and J.E. Ferrell. Ultrasensitivity in the mitogen-activated protein kinase cascade. *Proceedings of the National Academy of Sciences*, 93(19):10078–10083, 1996.

[48] S.A. Isaacson. The reaction-diffusion master equation as an asymptotic approximation of diffusion to a small target. *SIAM Journal on Applied Mathematics*, 70(1):77–111, 2009.

[49] J.R. Karr, J.C. Sanghvi, D.N. Macklin, M.V. Gutschow, J.M. Jacobs, B. Bolival Jr, N. Assad-Garcia, J.I. Glass, and M.W. Covert. A whole-cell computational model predicts phenotype from genotype. *Cell*, 150(2):389–401, 2012.

[50] M. Klann, L. Paulevé, T. Petrov, and H. Koeppl. Coarse-grained brownian dynamics simulation of rule-based models. In *Computational Methods in Systems Biology*, pages 64–77. Springer, 2013.

[51] E. Klipp, B. Nordlander, R. Kruger, P. Gennemark, and S. Hohmann. Integrative model of the response of yeast to osmotic shock. *Nature Biotechnology*, 23(8):975–982, 2005.

[52] K.W. Kohn, M.I. Aladjem, J.N. Weinstein, and Y. Pommier. Molecular interaction maps of bioregulatory networks: a general rubric for systems biology. *Molecular Biology of the Cell*, 17(1):1–13, 2006.

[53] C. Kuhn, K.V.S. Prasad, E. Klipp, and P. Gennemark. Formal representation of the high osmolarity glycerol pathway in yeast. *Genome Informatics*, 22:69–83, 2010.

[54] M. Kwiatkowski and I. Stark. The continuous π-calculus: A process algebra for biochemical modelling. In *Computational Methods in Systems Biology*, pages 103–122. Springer, 2008.

[55] C. Laneve and F. Tarissan. A simple calculus for proteins and cells. *Electronic Notes in Theoretical Computer Science*, 171(2):139–154, 2007.

[56] N. Le Novere, M. Hucka, H. Mi, S. Moodie, F. Schreiber, A. Sorokin, E. Demir, K. Wegner, M.I. Aladjem, S.M. Wimalaratne, et al. The systems biology graphical notation. *Nature Biotechnology*, 27(8):735–741, 2009.

[57] M. Lis, M.N. Artyomov, S. Devadas, and A.K. Chakraborty. Efficient stochastic simulation of reaction–diffusion processes via direct compilation. *Bioinformatics*, 25(17):2289–2291, 2009.

[58] C.F. Lopez, J.L. Muhlich, J.A. Bachman, and P.K. Sorger. Programming biological models in python using pysb. *Molecular Systems Biology*, 9(1), 2013.

[59] A. Mallavarapu, M. Thomson, B. Ullian, and J. Gunawardena. Programming with models: modularity and abstraction provide powerful capabilities for systems biology. *Journal of The Royal Society Interface*, 6(32):257, 2009.

[60] M.A. Marchisio, M. Colaiacovo, E. Whitehead, and J. Stelling. Modular, rule-based modeling for the design of eukaryotic synthetic gene circuits. *BMC Systems Biology*, 7(1):42, 2013.

[61] M.A. Marchisio and J. Stelling. Computational design of synthetic gene circuits with composable parts. *Bioinformatics*, 24(17):1903–1910, 2008.

[62] T.G. Mattos, C. Mejía-Monasterio, R. Metzler, and G. Oshanin. First passages in bounded domains: When is the mean first passage time meaningful? *Physical Review E*, 86(3):031143, 2012.

[63] C. Maus, S. Rybacki, and A.M. Uhrmacher. Rule-based multi-level modeling of cell biological systems. *BMC Systems Biology*, 5(1):166, 2011.

[64] S.R. McGuffee and A.H. Elcock. Diffusion, crowding & protein stability in a dynamic molecular model of the bacterial cytoplasm. *PLoS Computational Biology*, 6(3):e1000694, 2010.

[65] J.W. Moore. *Foundation technologies in synthetic biology: tools for use in understanding plant immunity.* PhD thesis, School of Engineering, University of Edinburgh, 2012.

[66] A. Palmisano, I. Mura, and C. Priami. From Odes to Language-based, executable models of Biological Systems. In *Pacific Symposium on Biocomputing 2009: Kohala Coast, Hawaii, USA, 5-9 January 2009*, page 239. World Scientific, 2009.

[67] M. Pedersen. A syntactic abstraction for rule-based languages with binding. *Electronic Notes in Theoretical Computer Science*, 277:53–64, 2011.

[68] S. Pelet, F. Rudolf, M. Nadal-Ribelles, E. de Nadal, F. Posas, and M. Peter. Transient activation of the hog mapk pathway regulates bimodal gene expression. *Science*, 332(6030):732–735, 2011.

[69] S. Pillay, M.J. Ward, A. Peirce, and T. Kolokolnikov. An asymptotic analysis of the mean first passage time for narrow escape problems: Part i: Two-dimensional domains. *Multiscale Modeling & Simulation*, 8(3):803–835, 2010.

[70] A. Regev, E.M. Panina, W. Silverman, L. Cardelli, and E. Shapiro. Bioambients: an abstraction for biological compartments. *Theoretical Computer Science*, 325(1):141–167, 2004.

[71] S. Schnell, R. Grima, and P.K. Maini. Multiscale modeling in biology. *American Scientist*, 95(2):134, 2007.

[72] A. Schürr and F. Klar. 15 years of triple graph grammars. In *Graph Transformations*, pages 411–425. Springer, 2008.

[73] Z. Schuss, A. Singer, and D. Holcman. The narrow escape problem for diffusion in cellular microdomains. *Proceedings of the National Academy of Sciences*, 104(41):16098–16103, 2007.

[74] R.P. Shetty, D. Endy, and T.F. Knight Jr. Engineering biobrick vectors from biobrick parts. *Journal of Biological Engineering*, 2(1):1–12, 2008.

[75] A. Singer, Z. Schuss, and D. Holcman. Narrow escape, part ii: The circular disk. *Journal of Statistical Physics*, 122(3):465–489, 2006.

[76] A. Singer, Z. Schuss, and D. Holcman. Narrow escape, part iii: Non-smooth domains and riemann surfaces. *Journal of Statistical Physics*, 122(3):491–509, 2006.

[77] A. Singer, Z. Schuss, and D. Holcman. Narrow escape and leakage of brownian particles. *Physical Review E*, 78(5):051111, 2008.

[78] A. Singer, Z. Schuss, D. Holcman, and R.S. Eisenberg. Narrow escape, part i. *Journal of Statistical Physics*, 122(3):437–463, 2006.

[79] A.M. Smith, W. Xu, Y. Sun, J.R. Faeder, and G.E. Marai. Rulebender: integrated modeling, simulation and visualization for rule-based intracellular biochemistry. *BMC Bioinformatics*, 13(Suppl 8):S3, 2012.

[80] O. Sorokina, A. Sorokin, and J.D. Armstrong. Towards a quantitative model of the post-synaptic proteome. *Molecular BioSystems*, 7(10):2813–2823, 2011.

[81] O. Sorokina, A. Sorokin, J.D. Armstrong, and V. Danos. A simulator for spatially extended kappa models. *Bioinformatics*, 29(23):3105, 2013.

[82] P. Stevens. A landscape of bidirectional model transformations. In *Generative and transformational techniques in software engineering II*, pages 408–424. Springer, 2008.

[83] D. Stewart. Spatial Biomodelling. Master's thesis, School of Informatics, University of Edinburgh, 2010.

[84] D. Stewart and J.R. Wilson-Kanamori. Modular modelling in synthetic biology: Light-based communication in ¡i¿e. coli¡/i¿. *Electronic Notes in Theoretical Computer Science*, 277:77–87, 2011.

[85] R. Suderman and E.J. Deeds. Machines vs. ensembles: Effective mapk signaling through heterogeneous sets of protein complexes. *PLoS Computational Biology*, 9(10):e1003278, 2013.

[86] Z. Tao. Biquery. http://www.prg.nii.ac.jp/members/stefanzan/biquery.html (accessed 29/7/14).

[87] C.D. Thompson-Walsh, J. Hayman, and G. Winskel. Containment in rule-based models. *Electronic Notes in Theoretical Computer Science*, 284:125–137, 2012.

[88] T. Thomson. Rule-Based Modeling of BioBrick Parts. Introduction to BioBrick modelling framework by Ty Thomson on Cellucidate, lost when site made inaccessible to public, 2009.

[89] T. Thomson, K.R. Benjamin, A. Bush, T. Love, D. Pincus, O. Resnekov, C.Y. Richard, A. Gordon, A. Colman-Lerner, D. Endy, et al. Scaffold number in yeast signaling system sets tradeoff between system output and dynamic range. *Proceedings of the National Academy of Sciences*, 108(50):20265–20270, 2011.

[90] J. Voigtländer. Bidirectionalization for free! (pearl). *ACM SIGPLAN Notices*, 44(1):165–176, 2009.

[91] W. Waites. The Design and Implementation of the Module Integration Simulator. Reference manual for mois tool and syntax, 2014.

[92] D. Wang, N. Amornsiripanitch, and X. Dong. A genomic approach to identify regulatory nodes in the transcriptional network of systemic acquired resistance in plants. *PLoS Pathogens*, 2(11):e123, 2006.

[93] A.Y. Weiße, D.A. Oyarzún, V. Danos, and P.S. Swain. Mechanistic links between cellular trade-offs, gene expression, and growth. *Proceedings of the National Academy of Sciences*, 112(9):E1038–E1047, 2015.

[94] J.R. Wilson-Kanamori. A Stochastic Rule-Based Model of the Mammalian Circadian Clock. Undergraduate honours thesis, School of Informatics, University of Edinburgh, 2010.

[95] J.R. Wilson-Kanamori and S Hidaka. A bidirectional collaboration framework for biomodel development. *Electronic Communications of the EASST*, 57, 2013.

[96] J.R. Wilson-Kanamori, R. Honorato-Zimmer, V. Danos, and T. Thomson. Kappa Rule-Based Modelling in Synthetic Biology. Scheduled to be published in Computational Methods in Synthetic Biology (2014), 2013.

[97] O. Wolkenhauer. Why model? *Frontiers in Physiology*, 5(21), 2014.

Appendix A

Kappa Model Code

A.1 Kinase-Phosphatase

```
# Vincent Danos and John Wilson-Kanamori
#
# Simple Kinase-Phosphatase - last update 25-06-2013
#
# A simple model of Goldbeter-Koshland kinase-phosphatase activity upon a target
     protein.
# The model achieves equilibrium depending on the rates between the two.


### Agents:
%agent: Kinase(a)
%agent: Phosphatase(a)
%agent: Target(x~u~p,y~u~p)


### Rules:

# Kinase action
'Kinase x binding' Kinase(a), Target(x) -> Kinase(a!1), Target(x!1) @ 1.0
'Kinase x phosphorylation' Kinase(a!1), Target(x~u!1) -> Kinase(a!1), Target(x~p
    !1) @ 100.0
'Kinase x unbinding' Kinase(a!1), Target(x!1) -> Kinase(a), Target(x) @ 5.0
'Kinase y binding' Kinase(a), Target(y) -> Kinase(a!1), Target(y!1) @ 10.0
'Kinase y phosphorylation' Kinase(a!1), Target(y~u!1) -> Kinase(a!1), Target(y~p
    !1) @ 100.0
'Kinase y unbinding' Kinase(a!1), Target(y!1) -> Kinase(a), Target(y) @ 5.0
```

```
# Phosphatase action
'Phosphatase x binding' Phosphatase(a), Target(x) -> Phosphatase(a!1), Target(x
    !1) @ 1.0
'Phosphatase x phosphorylation' Phosphatase(a!1), Target(x~p!1) -> Phosphatase(a
    !1), Target(x~u!1) @ 10.0
'Phosphatase x unbinding' Phosphatase(a!1), Target(x!1) -> Phosphatase(a),
    Target(x) @ 5.0
'Phosphatase y binding' Phosphatase(a), Target(y) -> Phosphatase(a!1), Target(y
    !1) @ 1.0
'Phosphatase y phosphorylation' Phosphatase(a!1), Target(y~p!1) -> Phosphatase(a
    !1), Target(y~u!1) @ 10.0
'Phosphatase y unbinding' Phosphatase(a!1), Target(y!1) -> Phosphatase(a),
    Target(y) @ 5.0


### Initial Conditions:
%init: 100 Target(x~u,y~u)
%init: 10 Kinase(a)
%init: 10 Phosphatase(a)

### Simulation:
%obs: 'TargetUP' Target(x~u,y~p)
%obs: 'TargetPU' Target(x~p,y~u)
%obs: 'TargetPP' Target(x~p,y~p)

### Stories:

### Perturbations:
```

A.2 Extended Kinase-Phosphatase

```
# John Wilson-Kanamori
#
# Extended Kinase-Phosphatase Model - last update 25-06-2013
#
# We extend the standard Kinase-Phosphatase Model with two compartments.
# In particular, the Kinase and Phosphatase start in the Cytosol and must travel
     to the Nucleus to act upon a Target.


### Spatial Structures:
%compartment: Cytosol
%compartment: Nucleus
%channel: CytNuc :Cytosol -> :Nucleus
%channel: NucCyt :Nucleus -> :Cytosol


### Agents:
%agent: Kinase(a)
%agent: Phosphatase(a)
%agent: Target(x~u~p,y~u~p)


### Rules:

# Kinase action
'K+Tx' Kinase(a), Target(x) -> Kinase(a!1), Target(x!1) @ 1.0
'Tx P-tion' Kinase(a!1), Target(x~u!1) -> Kinase(a!1), Target(x~p!1) @ 100.0
'K-Tx' Kinase(a!1), Target(x!1) -> Kinase(a), Target(x) @ 5.0
'K+Ty' Kinase(a), Target(y) -> Kinase(a!1), Target(y!1) @ 10.0
'Ty P-tion' Kinase(a!1), Target(y~u!1) -> Kinase(a!1), Target(y~p!1) @ 100.0
'K-Ty' Kinase(a!1), Target(y!1) -> Kinase(a), Target(y) @ 5.0

# Phosphatase action
'P+Tx' Phosphatase(a), Target(x) -> Phosphatase(a!1), Target(x!1) @ 1.0
'Tx DP-tion' Phosphatase(a!1), Target(x~p!1) -> Phosphatase(a!1), Target(x~u!1)
    @ 10.0
'P-Tx' Phosphatase(a!1), Target(x!1) -> Phosphatase(a), Target(x) @ 5.0
'P+Ty' Phosphatase(a), Target(y) -> Phosphatase(a!1), Target(y!1) @ 1.0
'Ty DP-tion' Phosphatase(a!1), Target(y~p!1) -> Phosphatase(a!1), Target(y~u!1)
    @ 10.0
```

```
'P-Ty' Phosphatase(a!1), Target(y!1) -> Phosphatase(a), Target(y) @ 5.0

# Translocation
'KCyto' Kinase(a) ->:NucCyt Kinase(a) @ 10.0
'KNuc' Kinase(a) ->:CytNuc Kinase(a) @ 10.0
'PCyto' Phosphatase(a) ->:NucCyt Phosphatase(a) @ 10.0
'PNuc' Phosphatase(a) ->:CytNuc Phosphatase(a) @ 10.0


### Initial Conditions:
%init: 100 :Nucleus Target(x~u,y~u)
%init: 10 :Cytosol Kinase(a)
%init: 10 :Cytosol Phosphatase(a)

### Simulation:
%obs: 'TargetUP' Target(x~u,y~p)
%obs: 'TargetPU' Target(x~p,y~u)
%obs: 'TargetPP' Target(x~p,y~p)

### Stories:

### Perturbations:
```

A.3 Cooperative Assembly

```
# Vincent Danos and John Wilson-Kanamori
#
# Cooperative Assembly System - last update 22-03-2011
#
# E.coli chemotactic receptor analysis.
#
# This is the expanded version of the cooperative assembly model, in which we
    fully elaborate the possible connections between the two agents involved.
# The simplified version of this model, as visualised in the thesis, simply sets
     k_off = k_off_v = k_off_t = 1.0 and condenses all equivalent rules.
# Concretely, each set of four rules below thus condenses into one basic rule (
    b1-a1, b1 a1, b1-a2, etc.).


### Agents:
%agent: A(a1,a2,a3)
%agent: B(b1,b2)


### Rules:
# The first set of 'k_off' rates describe a 'repulsive' model in which the more
    Bs bound to an A, the more likely they are to dissociate.
# The second set of 'k_off' rates describe a 'sticky' model in which the more Bs
     bound to an A, the less likely they are to dissociate.

%var: 'vol' 10
%var: 'k_on' 0.1/'vol'

#%var: 'k_off'  2
#%var: 'k_off_v' 5 * 'k_off'
#%var: 'k_off_t' 50 * 'k_off'

%var: 'coop_1' 1
%var: 'coop_2' 1

%var: 'k_off' 2
%var: 'k_off_v' 'k_off' / 'coop_1'
%var: 'k_off_t' 'k_off' / ('coop_2' * 'coop_1')
```

```
# Assumption - b1 and b2 can both bind to the same A, hence creating alkene
    bindings.
# To reverse this, need to explicitly describe A bonds (rather than using
    wildcards.
# KaSim does not know bidirectional rules, and Kappa cannot handle
    indistinguishable sites. Hence, things get a bit messy...
# For b1...

'b1-a1-00' B(b1 ), A(a1 ,a2 ,a3 ) -> B(b1!1), A(a1!1,a2 ,a3 ) @ 'k_on'
'b1-a1-01' B(b1 ), A(a1 ,a2 ,a3!_) -> B(b1!1), A(a1!1,a2 ,a3!_) @ 'k_on'
'b1-a1-10' B(b1 ), A(a1 ,a2!_,a3 ) -> B(b1!1), A(a1!1,a2!_,a3 ) @ 'k_on'
'b1-a1-11' B(b1 ), A(a1 ,a2!_,a3!_) -> B(b1!1), A(a1!1,a2!_,a3!_) @ 'k_on'

'b1 a1-00' B(b1!1), A(a1!1,a2 ,a3 ) -> B(b1 ), A(a1 ,a2 ,a3 ) @ 'k_off'
'b1 a1-01' B(b1!1), A(a1!1,a2 ,a3!_) -> B(b1 ), A(a1 ,a2 ,a3!_) @ 'k_off_v'
'b1 a1-10' B(b1!1), A(a1!1,a2!_,a3 ) -> B(b1 ), A(a1 ,a2!_,a3 ) @ 'k_off_v'
'b1 a1-11' B(b1!1), A(a1!1,a2!_,a3!_) -> B(b1 ), A(a1 ,a2!_,a3!_) @ 'k_off_t'

'b1-a2-00' B(b1 ), A(a1 ,a2 ,a3 ) -> B(b1!1), A(a1 ,a2!1,a3 ) @ 'k_on'
'b1-a2-01' B(b1 ), A(a1 ,a2 ,a3!_) -> B(b1!1), A(a1 ,a2!1,a3!_) @ 'k_on'
'b1-a2-10' B(b1 ), A(a1!_,a2 ,a3 ) -> B(b1!1), A(a1!_,a2!1,a3 ) @ 'k_on'
'b1-a2-11' B(b1 ), A(a1!_,a2 ,a3!_) -> B(b1!1), A(a1!_,a2!1,a3!_) @ 'k_on'

'b1 a2-00' B(b1!1), A(a1 ,a2!1,a3 ) -> B(b1 ), A(a1 ,a2 ,a3 ) @ 'k_off'
'b1 a2-01' B(b1!1), A(a1 ,a2!1,a3!_) -> B(b1 ), A(a1 ,a2 ,a3!_) @ 'k_off_v'
'b1 a2-10' B(b1!1), A(a1!_,a2!1,a3 ) -> B(b1 ), A(a1!_,a2 ,a3 ) @ 'k_off_v'
'b1 a2-11' B(b1!1), A(a1!_,a2!1,a3!_) -> B(b1 ), A(a1!_,a2 ,a3!_) @ 'k_off_t'

'b1-a3-00' B(b1 ), A(a1 ,a2 ,a3 ) -> B(b1!1), A(a1 ,a2 ,a3!1) @ 'k_on'
'b1-a3-01' B(b1 ), A(a1 ,a2!_,a3 ) -> B(b1!1), A(a1 ,a2!_,a3!1) @ 'k_on'
'b1-a3-10' B(b1 ), A(a1!_,a2 ,a3 ) -> B(b1!1), A(a1!_,a2 ,a3!1) @ 'k_on'
'b1-a3-11' B(b1 ), A(a1!_,a2!_,a3 ) -> B(b1!1), A(a1!_,a2!_,a3!1) @ 'k_on'

'b1 a3-00' B(b1!1), A(a1 ,a2 ,a3!1) -> B(b1 ), A(a1 ,a2 ,a3 ) @ 'k_off'
'b1 a3-01' B(b1!1), A(a1 ,a2!_,a3!1) -> B(b1 ), A(a1 ,a2!_,a3 ) @ 'k_off_v'
'b1 a3-10' B(b1!1), A(a1!_,a2 ,a3!1) -> B(b1 ), A(a1!_,a2 ,a3 ) @ 'k_off_v'
'b1 a3-11' B(b1!1), A(a1!_,a2!_,a3!1) -> B(b1 ), A(a1!_,a2!_,a3 ) @ 'k_off_t'


# Repeat for b2...

'b2-a1-00' B(b2 ), A(a1 ,a2 ,a3 ) -> B(b2!1), A(a1!1,a2 ,a3 ) @ 'k_on'
```

```
'b2-a1-01' B(b2 ), A(a1 ,a2 ,a3!_) -> B(b2!1), A(a1!1,a2 ,a3!_) @ 'k_on'
'b2-a1-10' B(b2 ), A(a1 ,a2!_,a3 ) -> B(b2!1), A(a1!1,a2!_,a3 ) @ 'k_on'
'b2-a1-11' B(b2 ), A(a1 ,a2!_,a3!_) -> B(b2!1), A(a1!1,a2!_,a3!_) @ 'k_on'

'b2 a1-00' B(b2!1), A(a1!1,a2 ,a3 ) -> B(b2 ), A(a1 ,a2 ,a3 ) @ 'k_off'
'b2 a1-01' B(b2!1), A(a1!1,a2 ,a3!_) -> B(b2 ), A(a1 ,a2 ,a3!_) @ 'k_off_v'
'b2 a1-10' B(b2!1), A(a1!1,a2!_,a3 ) -> B(b2 ), A(a1 ,a2!_,a3 ) @ 'k_off_v'
'b2 a1-11' B(b2!1), A(a1!1,a2!_,a3!_) -> B(b2 ), A(a1 ,a2!_,a3!_) @ 'k_off_t'

'b2-a2-00' B(b2 ), A(a1 ,a2 ,a3 ) -> B(b2!1), A(a1 ,a2!1,a3 ) @ 'k_on'
'b2-a2-01' B(b2 ), A(a1 ,a2 ,a3!_) -> B(b2!1), A(a1 ,a2!1,a3!_) @ 'k_on'
'b2-a2-10' B(b2 ), A(a1!_,a2 ,a3 ) -> B(b2!1), A(a1!_,a2!1,a3 ) @ 'k_on'
'b2-a2-11' B(b2 ), A(a1!_,a2 ,a3!_) -> B(b2!1), A(a1!_,a2!1,a3!_) @ 'k_on'

'b2 a2-00' B(b2!1), A(a1 ,a2!1,a3 ) -> B(b2 ), A(a1 ,a2 ,a3 ) @ 'k_off'
'b2 a2-01' B(b2!1), A(a1 ,a2!1,a3!_) -> B(b2 ), A(a1 ,a2 ,a3!_) @ 'k_off_v'
'b2 a2-10' B(b2!1), A(a1!_,a2!1,a3 ) -> B(b2 ), A(a1!_,a2 ,a3 ) @ 'k_off_v'
'b2 a2-11' B(b2!1), A(a1!_,a2!1,a3!_) -> B(b2 ), A(a1!_,a2 ,a3!_) @ 'k_off_t'

'b2-a3-00' B(b2 ), A(a1 ,a2 ,a3 ) -> B(b2!1), A(a1 ,a2 ,a3!1) @ 'k_on'
'b2-a3-01' B(b2 ), A(a1 ,a2!_,a3 ) -> B(b2!1), A(a1 ,a2!_,a3!1) @ 'k_on'
'b2-a3-10' B(b2 ), A(a1!_,a2 ,a3 ) -> B(b2!1), A(a1!_,a2 ,a3!1) @ 'k_on'
'b2-a3-11' B(b2 ), A(a1!_,a2!_,a3 ) -> B(b2!1), A(a1!_,a2!_,a3!1) @ 'k_on'

'b2 a3-00' B(b2!1), A(a1 ,a2 ,a3!1) -> B(b2 ), A(a1 ,a2 ,a3 ) @ 'k_off'
'b2 a3-01' B(b2!1), A(a1 ,a2!_,a3!1) -> B(b2 ), A(a1 ,a2!_,a3 ) @ 'k_off_v'
'b2 a3-10' B(b2!1), A(a1!_,a2 ,a3!1) -> B(b2 ), A(a1!_,a2 ,a3 ) @ 'k_off_v'
'b2 a3-11' B(b2!1), A(a1!_,a2!_,a3!1) -> B(b2 ), A(a1!_,a2!_,a3 ) @ 'k_off_t'


### Initial Conditions:
# Note the stoichiometry of the two sets of agents.
%var: 'n_A' (100 * 'vol')
%var: 'n_B' (150 * 'vol')
%init: 'n_A' (A(a1,a2,a3))
%init: 'n_B' (B(b1,b2))# E.coli chemotactic receptor analysis.


### Simulation:
# Observables on A.
%var: 'A0' A(a1 ,a2 ,a3 )
%var: 'a1' A(a1!_,a2 ,a3 )
```

```
%var: 'a2' A(a1 ,a2!_,a3 )
%var: 'a3' A(a1 ,a2 ,a3!_)
%var: 'A1' 'a1' + 'a2' + 'a3'
%var: 'a12' A(a1!_,a2!_,a3 )
%var: 'a13' A(a1!_,a2 ,a3!_)
%var: 'a23' A(a1 ,a2!_,a3!_)
%var: 'A2' 'a12' + 'a13' + 'a23'
%var: 'A3' A(a1!_,a2!_,a3!_)

%plot: 'A0'
%plot: 'A1'
%plot: 'A2'
%plot: 'A3'

# Observables on B.

%var: 'B0' B(b1 ,b2 )
%var: 'b1' B(b1!_,b2 )
%var: 'b2' B(b1 ,b2!_)
%var: 'B1' 'b1' + 'b2'
%var: 'B2' B(b1!_,b2!_)

%plot: 'B0'
%plot: 'B1'
%plot: 'B2'

# Structural observables.
# Edges simply count the number of single bindings (i.e. 3 edges in a t, 2 in an
     alkene).
# Similarly, there are three vs in a t, and the number of vs is equivalent to
    the number of A2s plus 3 * number of A3s minus the number of alkenes.
# Finally, A3 and ts are not quite equivalent... A3s can be an alkene, while ts
    cannot.

%var: 'b1a1' B(b1!1), A(a1!1)
%var: 'b1a2' B(b1!1), A(a2!1)
%var: 'b1a3' B(b1!1), A(a3!1)
%var: 'b2a1' B(b2!1), A(a1!1)
%var: 'b2a2' B(b2!1), A(a2!1)
%var: 'b2a3' B(b2!1), A(a3!1)
%var: 'edges' 'b1a1' + 'b1a2' + 'b1a3' + 'b2a1' + 'b2a2' + 'b2a3'
```

```
%var: 'b1b1a1a2' B(b1!1), B(b1!2), A(a1!1,a2!2)
%var: 'b1b1a1a3' B(b1!1), B(b1!2), A(a1!1,a3!2)
%var: 'b1b1a2a3' B(b1!1), B(b1!2), A(a2!1,a3!2)
%var: 'b1b2a1a2' B(b1!1), B(b2!2), A(a1!1,a2!2)
%var: 'b1b2a1a3' B(b1!1), B(b2!2), A(a1!1,a3!2)
%var: 'b1b2a2a3' B(b1!1), B(b2!2), A(a2!1,a3!2)
%var: 'b2b1a1a2' B(b2!1), B(b1!2), A(a1!1,a2!2)
%var: 'b2b1a1a3' B(b2!1), B(b1!2), A(a1!1,a3!2)
%var: 'b2b1a2a3' B(b2!1), B(b1!2), A(a2!1,a3!2)
%var: 'b2b2a1a2' B(b2!1), B(b2!2), A(a1!1,a2!2)
%var: 'b2b2a1a3' B(b2!1), B(b2!2), A(a1!1,a3!2)
%var: 'b2b2a2a3' B(b2!1), B(b2!2), A(a2!1,a3!2)
%var: 'vs' 'b1b1a1a2' + 'b1b1a1a3' + 'b1b1a2a3' + 'b1b2a1a2' + 'b1b2a1a3' + '
    b1b2a2a3' + 'b2b1a1a2' + 'b2b1a1a3' + 'b2b1a2a3' + 'b2b2a1a2' + 'b2b2a1a3' +
     'b2b2a2a3'

%var: 'b1b1b1a1a2a3' B(b1!1), B(b1!2), B(b1!3), A(a1!1,a2!2,a3!3)
%var: 'b1b1b2a1a2a3' B(b1!1), B(b1!2), B(b2!3), A(a1!1,a2!2,a3!3)
%var: 'b1b2b1a1a2a3' B(b1!1), B(b2!2), B(b1!3), A(a1!1,a2!2,a3!3)
%var: 'b1b2b2a1a2a3' B(b1!1), B(b2!2), B(b2!3), A(a1!1,a2!2,a3!3)
%var: 'b2b1b1a1a2a3' B(b2!1), B(b1!2), B(b1!3), A(a1!1,a2!2,a3!3)
%var: 'b2b1b2a1a2a3' B(b2!1), B(b1!2), B(b2!3), A(a1!1,a2!2,a3!3)
%var: 'b2b2b1a1a2a3' B(b2!1), B(b2!2), B(b1!3), A(a1!1,a2!2,a3!3)
%var: 'b2b2b2a1a2a3' B(b2!1), B(b2!2), B(b2!3), A(a1!1,a2!2,a3!3)
%var: 'ts' 'b1b1b1a1a2a3' + 'b1b1b2a1a2a3' + 'b1b2b1a1a2a3' + 'b1b2b2a1a2a3' + '
    b2b1b1a1a2a3' + 'b2b1b2a1a2a3' + 'b2b2b1a1a2a3' + 'b2b2b2a1a2a3'

%plot: 'edges'
%plot: 'vs'
%plot: 'ts'


# Double bonds ('alkenes')

%var: 'ba12' B(b1!1,b2!2), A(a1!1,a2!2)
%var: 'ba13' B(b1!1,b2!2), A(a1!1,a3!2)
%var: 'ba21' B(b1!1,b2!2), A(a2!1,a1!2)
%var: 'ba23' B(b1!1,b2!2), A(a2!1,a3!2)
%var: 'ba31' B(b1!1,b2!2), A(a3!1,a1!2)
%var: 'ba32' B(b1!1,b2!2), A(a3!1,a2!2)
%var: 'alkene' 'ba12' + 'ba13' + 'ba21' + 'ba23' + 'ba31' + 'ba32'
```

```
%plot: 'alkene'


### Stories:

### Perturbations:

# Take snapshots every 2 time units.
%mod: [T]>2 do $SNAPSHOT
%mod: [T]>4 do $SNAPSHOT
%mod: [T]>6 do $SNAPSHOT
%mod: [T]>8 do $SNAPSHOT
%mod: [T]>10 do $SNAPSHOT
```

A.4 A Simple MAPK Cascade

```
# John Wilson-Kanamori and Vincent Danos
#
# A Simple Huang-Ferrell MAPK Cascade - last update 22-09-2012


### Agents:
%agent: RAS(S1,S2)
%agent: RAF(x~u~p)
%agent: ERK(Y187~u~p,T185~u~p)
%agent: MEK(S218~u~p,s,S222~u~p)
%agent: MKP3(s)
%agent: PP2A2(s)
%agent: PP2A1(s)


### Rules:
'RAF-RAS binding' RAS(S2,S1),RAF(x~u) -> RAS(S2!1,S1),RAF(x~u!1) @ 1.0
'RAF-RAS phosphorylation' RAS(S2!1,S1),RAF(x~u!1) -> RAS(S2!1,S1),RAF(x~p!1) @
    1.0
'RAF-RAS unbinding' RAS(S2!1),RAF(x!1) -> RAS(S2),RAF(x) @ 1.0
'RAF-PP2A1 binding' PP2A1(s),RAF(x~p) -> PP2A1(s!1),RAF(x~p!1) @ 1.0
'RAF-PP2A1 dephosphorylation' PP2A1(s!1),RAF(x~p!1) -> PP2A1(s!1),RAF(x~u!1) @
    1.0
'RAF-PP2A1 unbinding' PP2A1(s!1),RAF(x!1) -> PP2A1(s),RAF(x) @ 1.0
'MEK-RAF binding S222' RAF(x~p),MEK(S222~u) -> RAF(x~p!1),MEK(S222~u!1) @ 1.0
'MEK-RAF binding S218' RAF(x~p),MEK(S218~u) -> RAF(x~p!1),MEK(S218~u!1) @ 1.0
'MEK-RAF phosphorylation S222' RAF(x~p!1),MEK(S222~u!1) -> RAF(x~p!1),MEK(S222~p
    !1) @ 1.0
'MEK-RAF phosphorylation S218' RAF(x~p!1),MEK(S218~u!1) -> RAF(x~p!1),MEK(S218~p
    !1) @ 1.0
'MEK-RAF unbinding S222' RAF(x!1),MEK(S222!1) -> RAF(x),MEK(S222) @ 1.0
'MEK-RAF unbinding S218' RAF(x!1),MEK(S218!1) -> RAF(x),MEK(S218) @ 1.0
'MEK-PP2A2 binding S222' PP2A2(s),MEK(S222~p) -> PP2A2(s!1),MEK(S222~p!1) @ 1.0
'MEK-PP2A2 binding S218' PP2A2(s),MEK(S218~p) -> PP2A2(s!1),MEK(S218~p!1) @ 1.0
'MEK-PP2A2 dephosphorylation S222' PP2A2(s!1),MEK(S222~p!1) -> PP2A2(s!1),MEK(
    S222~u!1) @ 1.0
'MEK-PP2A2 dephosphorylation S218' PP2A2(s!1),MEK(S218~p!1) -> PP2A2(s!1),MEK(
    S218~u!1) @ 1.0
'MEK-PP2A2 unbinding S222' MEK(S222!1),PP2A2(s!1) -> MEK(S222),PP2A2(s) @ 1.0
'MEK-PP2A2 unbinding S218' MEK(S218!1),PP2A2(s!1) -> MEK(S218),PP2A2(s) @ 1.0
```

```
'ERK-MEK binding T185' MEK(s,S218~p,S222~p),ERK(T185~u) -> MEK(s!1,S218~p,S222~p
    ),ERK(T185~u!1) @ 1.0
'ERK-MEK binding Y187' MEK(s,S218~p,S222~p),ERK(Y187~u) -> MEK(s!1,S218~p,S222~p
    ),ERK(Y187~u!1) @ 1.0
'ERK-MEK phosphorylation T185' MEK(s!1,S218~p,S222~p),ERK(T185~u!1) -> MEK(s!1,
    S218~p,S222~p),ERK(T185~p!1) @ 1.0
'ERK-MEK phosphorylation Y187' MEK(s!1,S218~p,S222~p),ERK(Y187~u!1) -> MEK(s!1,
    S218~p,S222~p),ERK(Y187~p!1) @ 1.0
'ERK-MEK unbinding T185' MEK(s!1),ERK(T185!1) -> MEK(s),ERK(T185) @ 1.0
'ERK-MEK unbinding Y187' MEK(s!1),ERK(Y187!1) -> MEK(s),ERK(Y187) @ 1.0
'ERK-MKP3 binding T185' MKP3(s),ERK(T185~p) -> MKP3(s!1),ERK(T185~p!1) @ 1.0
'ERK-MKP3 binding Y187' MKP3(s),ERK(Y187~p) -> MKP3(s!1),ERK(Y187~p!1) @ 1.0
'ERK-MKP3 dephosphorylation T185' MKP3(s!1),ERK(T185~p!1) -> MKP3(s!1),ERK(T185~
    u!1) @ 1.0
'ERK-MKP3 dephosphorylation Y187' MKP3(s!1),ERK(Y187~p!1) -> MKP3(s!1),ERK(Y187~
    u!1) @ 1.0
'ERK-MKP3 unbinding T185' MKP3(s!1),ERK(T185!1) -> MKP3(s),ERK(T185) @ 1.0
'ERK-MKP3 unbinding Y187' MKP3(s!1),ERK(Y187!1) -> MKP3(s),ERK(Y187) @ 1.0

### Initial Conditions:
%init: 300 RAS(S2,S1)
%init: 1000 RAF(x~u)
%init: 1000 ERK(Y187~u,T185~u)
%init: 1000 MEK(S218~u,s,S222~u)
%init: 200 MKP3(s)
%init: 200 PP2A1(s)
%init: 200 PP2A2(s)

### Simulation:
%obs: 'Fully Phosphorylated RAF' RAF(x~p?)
%obs: 'Fully Phosphorylated MEK' MEK(S222~p?,S218~p?)
%obs: 'Fully Phosphorylated ERK' ERK(T185~p?,Y187~p?)

### Stories:

### Perturbations:
```

A.5 The Mammalian Circadian Clock (Non-Spatial)

```
# John Wilson-Kanamori
#
# The Mammalian Circadian Clock (non-spatial) - last update 07-05-2014
#
# This model displays a circadian rhythm of approximately 24 hours.
# The oscillations lack stability, and are easily thrown out of synch with
# simulation time by stochastic events (i.e. a particularly large PER1 spike).


### Agents:
%agent: mRNA(enc~PER1~PER2~CRY1~CRY2~REVERBa,loc~nuc~cyt)
%agent: GENE(id~PER1~PER2~CRY1~CRY2~REVERBa,S1,S2,S3,S4,S5)
%agent: PER(id~1~2,P1~u~p,P2~u~p,cry,cki,loc~nuc~cyt)
%agent: CRY(id~1~2,per,clk,loc~nuc~cyt)
%agent: CKI(per,loc~nuc~cyt)
%agent: REVERBa(rore,loc~nuc~cyt)
%agent: EBOX-CLK-BMAL1(cry,gene)
%agent: RORE(reverba,gene)


### Rules:
# Translation
'tlpo' mRNA(enc~PER1,loc~cyt) -> mRNA(enc~PER1,loc~cyt), PER(id~1,P1~u,P2~u,cry,
    cki,loc~cyt) @ 10.0
'tlpt' mRNA(enc~PER2,loc~cyt) -> mRNA(enc~PER2,loc~cyt), PER(id~2,P1~u,P2~u,cry,
    cki,loc~cyt) @ 10.0
'tlro' mRNA(enc~CRY1,loc~cyt) -> mRNA(enc~CRY1,loc~cyt), CRY(id~1,per,clk,loc~
    cyt) @ 1.031
'tlrt' mRNA(enc~CRY2,loc~cyt) -> mRNA(enc~CRY2,loc~cyt), CRY(id~2,per,clk,loc~
    cyt) @ 1.031
'tlrv' mRNA(enc~REVERBa,loc~cyt) -> mRNA(enc~REVERBa,loc~cyt), REVERBa(rore,loc~
    cyt) @ 2.53

# Transcription
'trPo' GENE(id~PER1,S1!1,S2!2,S3!3,S4!4,S5!5), EBOX-CLK-BMAL1(cry,gene!1), EBOX-
    CLK-BMAL1(cry,gene!2), EBOX-CLK-BMAL1(cry,gene!3), EBOX-CLK-BMAL1(cry,gene
    !4), EBOX-CLK-BMAL1(cry,gene!5) -> GENE(id~PER1,S1!1,S2!2,S3!3,S4!4,S5!5),
    EBOX-CLK-BMAL1(cry,gene!1), EBOX-CLK-BMAL1(cry,gene!2), EBOX-CLK-BMAL1(cry,
    gene!3), EBOX-CLK-BMAL1(cry,gene!4), EBOX-CLK-BMAL1(cry,gene!5), mRNA(enc~
    PER1,loc~nuc) @ 9415.947
```

```
'trPt' GENE(id~PER2,S1!1,S2!2,S3!3,S4!4,S5!5), EBOX-CLK-BMAL1(cry,gene!1), EBOX-
    CLK-BMAL1(cry,gene!2), EBOX-CLK-BMAL1(cry,gene!3), EBOX-CLK-BMAL1(cry,gene
    !4), EBOX-CLK-BMAL1(cry,gene!5) -> GENE(id~PER2,S1!1,S2!2,S3!3,S4!4,S5!5),
    EBOX-CLK-BMAL1(cry,gene!1), EBOX-CLK-BMAL1(cry,gene!2), EBOX-CLK-BMAL1(cry,
    gene!3), EBOX-CLK-BMAL1(cry,gene!4), EBOX-CLK-BMAL1(cry,gene!5), mRNA(enc~
    PER2,loc~nuc) @ 3601.244
'trRo' GENE(id~CRY1,S1!1,S2!2,S3!3,S4!4,S5), EBOX-CLK-BMAL1(cry,gene!1), RORE(
    reverba,gene!2), RORE(reverba,gene!3), RORE(reverba,gene!4) -> GENE(id~CRY1,
    S1!1,S2!2,S3!3,S4!4,S5), EBOX-CLK-BMAL1(cry,gene!1), RORE(reverba,gene!2),
    RORE(reverba,gene!3), RORE(reverba,gene!4), mRNA(enc~CRY1,loc~nuc) @
    105.3084
'trRt' GENE(id~CRY2,S1!1,S2!2,S3!3,S4!4,S5), EBOX-CLK-BMAL1(cry,gene!1), RORE(
    reverba,gene!2), RORE(reverba,gene!3), RORE(reverba,gene!4) -> GENE(id~CRY2,
    S1!1,S2!2,S3!3,S4!4,S5), EBOX-CLK-BMAL1(cry,gene!1), RORE(reverba,gene!2),
    RORE(reverba,gene!3), RORE(reverba,gene!4), mRNA(enc~CRY2,loc~nuc) @ 89.3318
'trRv' GENE(id~REVERBa,S1!1,S2!2,S3!3,S4,S5), EBOX-CLK-BMAL1(cry,gene!1), EBOX-
    CLK-BMAL1(cry,gene!2), EBOX-CLK-BMAL1(cry,gene!3) -> GENE(id~REVERBa,S1!1,S2
    !2,S3!3,S4,S5), EBOX-CLK-BMAL1(cry,gene!1), EBOX-CLK-BMAL1(cry,gene!2), EBOX
    -CLK-BMAL1(cry,gene!3), mRNA(enc~REVERBa,loc~nuc) @ 0.58

# Degradation
'uRv' REVERBa() -> @ 16.25
'umPo' mRNA(enc~PER1,loc~cyt) -> @ 6.21
'umPt' mRNA(enc~PER2,loc~cyt) -> @ 0.38
'umRo' mRNA(enc~CRY1,loc~cyt) -> @ 0.30
'umRt' mRNA(enc~CRY2,loc~cyt) -> @ 0.30
'umRv' mRNA(enc~REVERBa,loc~cyt) -> @ 15.11
'up' PER(P1~p,cry) -> @ 3.39
'upu' PER(P1~u) -> @ 0.08
'uro' CRY(id~1,per) -> @ 0.44
'urt' CRY(id~2,per) -> @ 0.59

# Phosphorylation
'hoo' CKI(per!1), PER(id~1,P1~u,P2~u,cki!1) -> CKI(per!1), PER(id~1,P1~p,P2~u,
    cki!1) @ 0.29
'hot' PER(id~2,P1~u,P2~u,cki!1), CKI(per!1) -> PER(id~2,P1~p,P2~u,cki!1), CKI(
    per!1) @ 0.09
'hto' PER(id~1,P1~p,P2~u,cki!1), CKI(per!1) -> PER(id~1,P1~p,P2~p,cki!1), CKI(
    per!1) @ 1.45

# Nuclear Import / Export
'ne Ref1' PER(cry,cki,loc~nuc) -> PER(cry,cki,loc~cyt) @ 0.71
```

```
'ne Ref2' PER(cki,cry!1,loc~nuc), CRY(per!1,clk,loc~nuc) -> PER(cki,cry!1,loc~
    cyt), CRY(per!1,clk,loc~cyt) @ 0.71
'ne Ref3' PER(cry,cki!1,loc~nuc), CKI(per!1,loc~nuc) -> PER(cry,cki!1,loc~cyt),
    CKI(per!1,loc~cyt) @ 0.71
'ne Ref4' PER(cki!2,cry!1,loc~nuc), CKI(per!2,loc~nuc), CRY(per!1,clk,loc~nuc)
    -> PER(cki!2,cry!1,loc~cyt), CKI(per!2,loc~cyt), CRY(per!1,clk,loc~cyt) @
    0.71
'neRv' REVERBa(rore,loc~nuc) -> REVERBa(rore,loc~cyt) @ 0.71
'tmc' mRNA(loc~nuc) -> mRNA(loc~cyt) @ 0.42
'nl Ref1' PER(P1~p,P2~u,cry,cki,loc~cyt) -> PER(P1~p,P2~u,cry,cki,loc~nuc) @
    2.31
'nl Ref2' PER(P1~p,P2~u,cry!1,cki,loc~cyt), CRY(per!1,loc~cyt) -> PER(P1~p,P2~u,
    cry!1,cki,loc~nuc), CRY(per!1,loc~nuc) @ 2.31
'nl Ref3' PER(P1~p,P2~u,cki!1,loc~cyt), CKI(per!1,loc~cyt) -> PER(P1~p,P2~u,cki
    !1,loc~nuc), CKI(per!1,loc~nuc) @ 2.31
'nl Ref4' PER(P1~p,P2~u,cry!2,cki!1,loc~cyt), CKI(per!1,loc~cyt), CRY(per!2,loc~
    cyt) -> PER(P1~p,P2~u,cry!2,cki!1,loc~nuc), CKI(per!1,loc~nuc), CRY(per!2,
    loc~nuc) @ 2.31
'nlRv' REVERBa(loc~cyt) -> REVERBa(loc~nuc) @ 2.31

# Reactions
'ac Ref1' PER(cki,loc~cyt), CKI(per,loc~cyt) -> PER(cki!1,loc~cyt), CKI(per!1,
    loc~cyt) @ 0.000348
'ac Ref2' CKI(per,loc~nuc), PER(cki,loc~nuc) -> CKI(per!1,loc~nuc), PER(cki!1,
    loc~nuc) @ 0.040517
'ar Ref1' CRY(per,loc~cyt), PER(cry,P1~p,loc~cyt) -> CRY(per!1,loc~cyt), PER(cry
    !1,P1~p,loc~cyt) @ 0.0002
'ar Ref2' CRY(per,loc~nuc), PER(cry,P1~p,loc~nuc) -> CRY(per!1,loc~nuc), PER(cry
    !1,P1~p,loc~nuc) @ 0.023276
'bin' CRY(clk,loc~nuc), EBOX-CLK-BMAL1(cry,gene!_) -> CRY(clk!2,loc~nuc), EBOX-
    CLK-BMAL1(cry!2,gene!_) @ 127.2862
'binRv' REVERBa(rore,loc~nuc), RORE(gene!_,reverba) -> REVERBa(rore!1,loc~nuc),
    RORE(gene!_,reverba!1) @ 0.11207
'dc' CKI(per!1), PER(cki!1) -> CKI(per), PER(cki) @ 5.09
'dr' PER(cry!1), CRY(per!1) -> PER(cry), CRY(per) @ 0.35
'unbin' CRY(clk!2,loc~nuc), EBOX-CLK-BMAL1(cry!2,gene!_) -> CRY(clk,loc~nuc),
    EBOX-CLK-BMAL1(cry,gene!_) @ 23.78
'unbinRv' REVERBa(rore!1,loc~nuc), RORE(reverba!1,gene!_) -> REVERBa(rore,loc~
    nuc), RORE(reverba,gene!_) @ 21.76

# Light
%var: 'light' 0.0
```

```
'zg Ref1' GENE(id~PER1) -> GENE(id~PER1), mRNA(enc~PER1,loc~nuc) @ 'light'
'zg Ref2' GENE(id~PER2) -> GENE(id~PER2), mRNA(enc~PER2,loc~nuc) @ 'light'


### Initial Conditions:
%init: 2 GENE(id~PER1,S1!1,S2!2,S3!3,S4!4,S5!5), EBOX-CLK-BMAL1(gene!1,cry),
    EBOX-CLK-BMAL1(gene!2,cry), EBOX-CLK-BMAL1(gene!3,cry), EBOX-CLK-BMAL1(gene
    !4,cry), EBOX-CLK-BMAL1(gene!5,cry)
%init: 2 GENE(id~PER2,S1!1,S2!2,S3!3,S4!4,S5!5), EBOX-CLK-BMAL1(gene!1,cry),
    EBOX-CLK-BMAL1(gene!2,cry), EBOX-CLK-BMAL1(gene!3,cry), EBOX-CLK-BMAL1(gene
    !4,cry), EBOX-CLK-BMAL1(gene!5,cry)
%init: 2 GENE(id~CRY1,S1!1,S2!2,S3!3,S4!4,S5), EBOX-CLK-BMAL1(gene!1,cry), RORE(
    gene!2,reverba), RORE(gene!3,reverba), RORE(gene!4,reverba)
%init: 2 GENE(id~CRY2,S1!1,S2!2,S3!3,S4!4,S5), EBOX-CLK-BMAL1(gene!1,cry), RORE(
    gene!2,reverba), RORE(gene!3,reverba), RORE(gene!4,reverba)
%init: 2 GENE(id~REVERBa,S1!1,S2!2,S3!3,S4,S5), EBOX-CLK-BMAL1(gene!1,cry), EBOX
    -CLK-BMAL1(gene!2,cry), EBOX-CLK-BMAL1(gene!3,cry)
%init: 4 mRNA(enc~PER1,loc~cyt)
%init: 23 mRNA(enc~PER2,loc~cyt)
%init: 29 mRNA(enc~CRY1,loc~nuc)
%init: 4706 mRNA(enc~CRY1,loc~cyt)
%init: 24 mRNA(enc~CRY2,loc~nuc)
%init: 3996 mRNA(enc~CRY2,loc~cyt)
%init: 130 PER(id~1,cki,P1~u,cry,P2~u,loc~cyt)
%init: 1374 PER(id~2,cki,P1~u,cry,P2~u,loc~cyt)
%init: 74 PER(id~1,cki!1,P1~u,cry,P2~u,loc~cyt), CKI(per!1,loc~cyt)
%init: 815 PER(id~2,cki!1,P1~u,cry,P2~u,loc~cyt), CKI(per!1,loc~cyt)
%init: 11070 CRY(id~1,per,clk,loc~cyt)
%init: 6993 CRY(id~2,per,clk,loc~cyt)
%init: 10404 CKI(per,loc~cyt)
%init: 4 CKI(per,loc~nuc)
%init: 1 PER(id~1,cki!1,P1~p,cry,P2~u,loc~cyt), CKI(per!1,loc~cyt)
%init: 7 PER(id~2,cki!1,P1~p,cry,P2~u,loc~cyt), CKI(per!1,loc~cyt)
%init: 4 PER(id~2,cki,P1~p,cry,P2~u,loc~cyt)
%init: 1 PER(id~1,cki!1,P1~p,cry,P2~p,loc~cyt), CKI(per!1,loc~cyt)
%init: 1 PER(id~1,cki,P1~p,cry,P2~p,loc~cyt)
%init: 1 PER(id~1,cki,P1~p,cry!1,P2~u,loc~cyt), CRY(id~1,per!1,clk,loc~cyt)
%init: 14 PER(id~2,cki,P1~p,cry!1,P2~u,loc~cyt), CRY(id~1,per!1,clk,loc~cyt)
%init: 1 PER(id~1,cki!2,P1~p,cry!1,P2~u,loc~cyt), CRY(id~1,per!1,clk,loc~cyt),
    CKI(per!2,loc~cyt)
%init: 12 PER(id~2,cki!2,P1~p,cry!1,P2~u,loc~cyt), CRY(id~1,per!1,clk,loc~cyt),
    CKI(per!2,loc~cyt)
```

```
%init: 1 PER(id~1,cki,P1~p,cry!1,P2~u,loc~cyt), CRY(id~2,per!1,clk,loc~cyt)
%init: 9 CRY(id~2,per!1,clk,loc~cyt), PER(id~2,cki,P1~p,cry!1,P2~u,loc~cyt)
%init: 1 PER(id~1,cki!2,P1~p,cry!1,P2~u,loc~cyt), CRY(id~2,per!1,clk,loc~cyt),
    CKI(per!2,loc~cyt)
%init: 7 PER(id~2,cki!2,P1~p,cry!1,P2~u,loc~cyt), CKI(per!2,loc~cyt), CRY(id~2,
    per!1,clk,loc~cyt)
%init: 17 PER(id~1,cki,P1~p,cry!1,P2~p,loc~cyt), CRY(id~1,per!1,clk,loc~cyt)
%init: 11 PER(id~1,cki!1,P1~p,cry!2,P2~p,loc~cyt), CRY(id~1,clk,per!2,loc~cyt),
    CKI(per!1,loc~cyt)
%init: 10 PER(id~1,cki,P1~p,cry!1,P2~p,loc~cyt), CRY(id~2,per!1,clk,loc~cyt)
%init: 7 PER(id~1,cki!1,P1~p,cry!2,P2~p,loc~cyt), CRY(id~2,clk,per!2,loc~cyt),
    CKI(per!1,loc~cyt)


### Simulation:
#%obs: 'PER1 mRNA' mRNA(enc~PER1)
#%obs: 'PER2 mRNA' mRNA(enc~PER2)
#%obs: 'CRY1 mRNA' mRNA(enc~CRY1)
#%obs: 'CRY2 mRNA' mRNA(enc~CRY2)
%obs: 'PER1' PER(id~1)
%obs: 'PER2' PER(id~2)
%obs: 'CRY1' CRY(id~1)
%obs: 'CRY2' CRY(id~2)

### Perturbations:
%mod: [T]>12 do $UPDATE 'light' 0.00
%mod: [T]>24 do $UPDATE 'light' 0.000339
%mod: [T]>36 do $UPDATE 'light' 0.00
%mod: [T]>48 do $UPDATE 'light' 0.000339
%mod: [T]>60 do $UPDATE 'light' 0.00
%mod: [T]>72 do $UPDATE 'light' 0.000339
%mod: [T]>84 do $UPDATE 'light' 0.00
%mod: [T]>96 do $UPDATE 'light' 0.000339
%mod: [T]>108 do $UPDATE 'light' 0.00
%mod: [T]>120 do $UPDATE 'light' 0.000339
%mod: [T]>132 do $UPDATE 'light' 0.00
%mod: [T]>144 do $UPDATE 'light' 0.000339
%mod: [T]>156 do $UPDATE 'light' 0.00
%mod: [T]>168 do $UPDATE 'light' 0.000339
%mod: [T]>180 do $UPDATE 'light' 0.00
%mod: [T]>192 do $UPDATE 'light' 0.000339
%mod: [T]>204 do $UPDATE 'light' 0.00
```

```
%mod: [T]>216 do $UPDATE 'light' 0.000339
%mod: [T]>228 do $UPDATE 'light' 0.00
%mod: [T]>240 do $UPDATE 'light' 0.000339
%mod: [T]>252 do $UPDATE 'light' 0.00
```

A.6 The Mammalian Circadian Clock (Spatial)

```
# John Wilson-Kanamori
#
# The Mammalian Circadian Clock - last update 01-07-2014
#
# This model displays a circadian rhythm of approximately 24 hours.
# The oscillations lack stability, and are easily thrown out of synch with
# simulation time by stochastic events (i.e. a particularly large PER1 spike).


### Spatial Structures:
%compartment: Nucleus
%compartment: Cytosol
%channel: Export :Nucleus -> :Cytosol
%channel: Import :Cytosol -> :Nucleus


### Agents:
%agent: mRNA(enc~PER1~PER2~CRY1~CRY2~REVERBa)
%agent: GENE(id~PER1~PER2~CRY1~CRY2~REVERBa,S1,S2,S3,S4,S5)
%agent: PER(id~1~2,P1~u~p,P2~u~p,cry,cki)
%agent: CRY(id~1~2,per,clk)
%agent: CKI(per)
%agent: REVERBa(rore)
%agent: EBOX-CLK-BMAL1(cry,gene)
%agent: RORE(reverba,gene)


### Rules:
# Translation
'tlpo' :Cytosol mRNA(enc~PER1) -> :Cytosol mRNA(enc~PER1), PER(id~1,P1~u,P2~u,
    cry,cki) @ 10.0
'tlpt' :Cytosol mRNA(enc~PER2) -> :Cytosol mRNA(enc~PER2), PER(id~2,P1~u,P2~u,
    cry,cki) @ 10.0
'tlro' :Cytosol mRNA(enc~CRY1) -> :Cytosol mRNA(enc~CRY1), CRY(id~1,per,clk) @
    1.031
'tlrt' :Cytosol mRNA(enc~CRY2) -> :Cytosol mRNA(enc~CRY2), CRY(id~2,per,clk) @
    1.031
'tlrv' :Cytosol mRNA(enc~REVERBa) -> :Cytosol mRNA(enc~REVERBa), REVERBa(rore) @
     2.53
```

```
# Transcription
'trPo' :Nucleus GENE(id~PER1,S1!1,S2!2,S3!3,S4!4,S5!5), EBOX-CLK-BMAL1(cry,gene
    !1), EBOX-CLK-BMAL1(cry,gene!2), EBOX-CLK-BMAL1(cry,gene!3), EBOX-CLK-BMAL1(
    cry,gene!4), EBOX-CLK-BMAL1(cry,gene!5) -> :Nucleus GENE(id~PER1,S1!1,S2!2,
    S3!3,S4!4,S5!5), EBOX-CLK-BMAL1(cry,gene!1), EBOX-CLK-BMAL1(cry,gene!2),
    EBOX-CLK-BMAL1(cry,gene!3), EBOX-CLK-BMAL1(cry,gene!4), EBOX-CLK-BMAL1(cry,
    gene!5), mRNA(enc~PER1) @ 9415.947
'trPt' :Nucleus GENE(id~PER2,S1!1,S2!2,S3!3,S4!4,S5!5), EBOX-CLK-BMAL1(cry,gene
    !1), EBOX-CLK-BMAL1(cry,gene!2), EBOX-CLK-BMAL1(cry,gene!3), EBOX-CLK-BMAL1(
    cry,gene!4), EBOX-CLK-BMAL1(cry,gene!5) -> :Nucleus GENE(id~PER2,S1!1,S2!2,
    S3!3,S4!4,S5!5), EBOX-CLK-BMAL1(cry,gene!1), EBOX-CLK-BMAL1(cry,gene!2),
    EBOX-CLK-BMAL1(cry,gene!3), EBOX-CLK-BMAL1(cry,gene!4), EBOX-CLK-BMAL1(cry,
    gene!5), mRNA(enc~PER2) @ 3601.244
'trRo' :Nucleus GENE(id~CRY1,S1!1,S2!2,S3!3,S4!4,S5), EBOX-CLK-BMAL1(cry,gene!1)
    , RORE(reverba,gene!2), RORE(reverba,gene!3), RORE(reverba,gene!4) -> :
    Nucleus GENE(id~CRY1,S1!1,S2!2,S3!3,S4!4,S5), EBOX-CLK-BMAL1(cry,gene!1),
    RORE(reverba,gene!2), RORE(reverba,gene!3), RORE(reverba,gene!4), mRNA(enc~
    CRY1) @ 105.3084
'trRt' :Nucleus GENE(id~CRY2,S1!1,S2!2,S3!3,S4!4,S5), EBOX-CLK-BMAL1(cry,gene!1)
    , RORE(reverba,gene!2), RORE(reverba,gene!3), RORE(reverba,gene!4) -> :
    Nucleus GENE(id~CRY2,S1!1,S2!2,S3!3,S4!4,S5), EBOX-CLK-BMAL1(cry,gene!1),
    RORE(reverba,gene!2), RORE(reverba,gene!3), RORE(reverba,gene!4), mRNA(enc~
    CRY2) @ 89.3318
'trRv' :Nucleus GENE(id~REVERBa,S1!1,S2!2,S3!3,S4,S5), EBOX-CLK-BMAL1(cry,gene
    !1), EBOX-CLK-BMAL1(cry,gene!2), EBOX-CLK-BMAL1(cry,gene!3) -> :Nucleus GENE
    (id~REVERBa,S1!1,S2!2,S3!3,S4,S5), EBOX-CLK-BMAL1(cry,gene!1), EBOX-CLK-
    BMAL1(cry,gene!2), EBOX-CLK-BMAL1(cry,gene!3), mRNA(enc~REVERBa) @ 0.58

# Degradation
'uRv' REVERBa() -> @ 16.25
'umPo' :Cytosol mRNA(enc~PER1) -> @ 6.21
'umPt' :Cytosol mRNA(enc~PER2) -> @ 0.38
'umRo' :Cytosol mRNA(enc~CRY1) -> @ 0.30
'umRt' :Cytosol mRNA(enc~CRY2) -> @ 0.30
'umRv' :Cytosol mRNA(enc~REVERBa) -> @ 15.11
'up' PER(P1~p,cry) -> @ 3.39
'upu' PER(P1~u) -> @ 0.08
'uro' CRY(id~1,per) -> @ 0.44
'urt' CRY(id~2,per) -> @ 0.59

# Phosphorylation
'hoo' CKI(per!1), PER(id~1,P1~u,P2~u,cki!1) -> CKI(per!1), PER(id~1,P1~p,P2~u,
```

```
    cki!1) @ 0.29
'hot' PER(id~2,P1~u,P2~u,cki!1), CKI(per!1) -> PER(id~2,P1~p,P2~u,cki!1), CKI(
    per!1) @ 0.09
'hto' PER(id~1,P1~p,P2~u,cki!1), CKI(per!1) -> PER(id~1,P1~p,P2~p,cki!1), CKI(
    per!1) @ 1.45

# Nuclear Import / Export
'ne Ref1' PER(cry,cki) ->:Export PER(cry,cki) @ 0.71
'ne Ref2' PER(cki,cry!1), CRY(per!1,clk) ->:Export PER(cki,cry!1), CRY(per!1,clk
    ) @ 0.71
'ne Ref3' PER(cry,cki!1), CKI(per!1) ->:Export PER(cry,cki!1), CKI(per!1) @ 0.71
'ne Ref4' PER(cki!2,cry!1), CKI(per!2), CRY(per!1,clk) ->:Export PER(cki!2,cry
    !1), CKI(per!2), CRY(per!1,clk) @ 0.71
'neRv' REVERBa(rore) ->:Export REVERBa(rore) @ 0.71
'tmc' mRNA() ->:Export mRNA() @ 0.42
'nl Ref1' PER(P1~p,P2~u,cry,cki) ->:Import PER(P1~p,P2~u,cry,cki) @ 2.31
'nl Ref2' PER(P1~p,P2~u,cry!1,cki), CRY(per!1) ->:Import PER(P1~p,P2~u,cry!1,cki
    ), CRY(per!1) @ 2.31
'nl Ref3' PER(P1~p,P2~u,cki!1,cry), CKI(per!1) ->:Import PER(P1~p,P2~u,cki!1,cry
    ), CKI(per!1) @ 2.31
'nl Ref4' PER(P1~p,P2~u,cry!2,cki!1), CKI(per!1), CRY(per!2) ->:Import PER(P1~p,
    P2~u,cry!2,cki!1), CKI(per!1), CRY(per!2) @ 2.31
'nlRv' REVERBa() ->:Import REVERBa() @ 2.31

# Reactions
'ac Ref1' :Cytosol PER(cki), CKI(per) -> :Cytosol PER(cki!1), CKI(per!1) @
    0.000348
'ac Ref2' :Nucleus CKI(per), PER(cki) -> :Nucleus CKI(per!1), PER(cki!1) @
    0.040517
'ar Ref1' :Cytosol CRY(per), PER(cry,P1~p) -> :Cytosol CRY(per!1), PER(cry!1,P1~
    p) @ 0.0002
'ar Ref2' :Nucleus CRY(per), PER(cry,P1~p) -> :Nucleus CRY(per!1), PER(cry!1,P1~
    p) @ 0.023276
'bin' :Nucleus CRY(clk), EBOX-CLK-BMAL1(cry,gene!_) -> :Nucleus CRY(clk!2), EBOX
    -CLK-BMAL1(cry!2,gene!_) @ 127.2862
'binRv' :Nucleus REVERBa(rore), RORE(gene!_,reverba) -> :Nucleus REVERBa(rore!1)
    , RORE(gene!_,reverba!1) @ 0.11207
'dc' CKI(per!1), PER(cki!1) -> CKI(per), PER(cki) @ 5.09
'dr' PER(cry!1), CRY(per!1) -> PER(cry), CRY(per) @ 0.35
'unbin' :Nucleus CRY(clk!2), EBOX-CLK-BMAL1(cry!2,gene!_) -> :Nucleus CRY(clk),
    EBOX-CLK-BMAL1(cry,gene!_) @ 23.78
'unbinRv' :Nucleus REVERBa(rore!1), RORE(reverba!1,gene!_) -> :Nucleus REVERBa(
```

```
    rore), RORE(reverba,gene!_) @ 21.76

# Light
%var: 'light' 0.0
'zg Ref1' :Nucleus GENE(id~PER1) -> :Nucleus GENE(id~PER1), mRNA(enc~PER1) @ '
    light'
'zg Ref2' :Nucleus GENE(id~PER2) -> :Nucleus GENE(id~PER2), mRNA(enc~PER2) @ '
    light'


### Initial Conditions:
%init: 2 :Nucleus GENE(id~PER1,S1!1,S2!2,S3!3,S4!4,S5!5), EBOX-CLK-BMAL1(gene!1,
    cry), EBOX-CLK-BMAL1(gene!2,cry), EBOX-CLK-BMAL1(gene!3,cry), EBOX-CLK-BMAL1
    (gene!4,cry), EBOX-CLK-BMAL1(gene!5,cry)
%init: 2 :Nucleus GENE(id~PER2,S1!1,S2!2,S3!3,S4!4,S5!5), EBOX-CLK-BMAL1(gene!1,
    cry), EBOX-CLK-BMAL1(gene!2,cry), EBOX-CLK-BMAL1(gene!3,cry), EBOX-CLK-BMAL1
    (gene!4,cry), EBOX-CLK-BMAL1(gene!5,cry)
%init: 2 :Nucleus GENE(id~CRY1,S1!1,S2!2,S3!3,S4!4,S5), EBOX-CLK-BMAL1(gene!1,
    cry), RORE(gene!2,reverba), RORE(gene!3,reverba), RORE(gene!4,reverba)
%init: 2 :Nucleus GENE(id~CRY2,S1!1,S2!2,S3!3,S4!4,S5), EBOX-CLK-BMAL1(gene!1,
    cry), RORE(gene!2,reverba), RORE(gene!3,reverba), RORE(gene!4,reverba)
%init: 2 :Nucleus GENE(id~REVERBa,S1!1,S2!2,S3!3,S4,S5), EBOX-CLK-BMAL1(gene!1,
    cry), EBOX-CLK-BMAL1(gene!2,cry), EBOX-CLK-BMAL1(gene!3,cry)
%init: 4 :Cytosol mRNA(enc~PER1)
%init: 23 :Cytosol mRNA(enc~PER2)
%init: 29 :Nucleus mRNA(enc~CRY1)
%init: 4706 :Cytosol mRNA(enc~CRY1)
%init: 24 :Nucleus mRNA(enc~CRY2)
%init: 3996 :Cytosol mRNA(enc~CRY2)
%init: 130 :Cytosol PER(id~1,cki,P1~u,cry,P2~u)
%init: 1374 :Cytosol PER(id~2,cki,P1~u,cry,P2~u)
%init: 74 :Cytosol PER(id~1,cki!1,P1~u,cry,P2~u), CKI(per!1)
%init: 815 :Cytosol PER(id~2,cki!1,P1~u,cry,P2~u), CKI(per!1)
%init: 11070 :Cytosol CRY(id~1,per,clk)
%init: 6993 :Cytosol CRY(id~2,per,clk)
%init: 10404 :Cytosol CKI(per)
%init: 4 :Nucleus CKI(per)
%init: 1 :Cytosol PER(id~1,cki!1,P1~p,cry,P2~u), CKI(per!1)
%init: 7 :Cytosol PER(id~2,cki!1,P1~p,cry,P2~u), CKI(per!1)
%init: 4 :Cytosol PER(id~2,cki,P1~p,cry,P2~u)
%init: 1 :Cytosol PER(id~1,cki!1,P1~p,cry,P2~p), CKI(per!1)
%init: 1 :Cytosol PER(id~1,cki,P1~p,cry,P2~p)
```

```
%init: 1 :Cytosol PER(id~1,cki,P1~p,cry!1,P2~u), CRY(id~1,per!1,clk)
%init: 14 :Cytosol PER(id~2,cki,P1~p,cry!1,P2~u), CRY(id~1,per!1,clk)
%init: 1 :Cytosol PER(id~1,cki!2,P1~p,cry!1,P2~u), CRY(id~1,per!1,clk), CKI(per
    !2)
%init: 12 :Cytosol PER(id~2,cki!2,P1~p,cry!1,P2~u), CRY(id~1,per!1,clk), CKI(per
    !2)
%init: 1 :Cytosol PER(id~1,cki,P1~p,cry!1,P2~u), CRY(id~2,per!1,clk)
%init: 9 :Cytosol CRY(id~2,per!1,clk), PER(id~2,cki,P1~p,cry!1,P2~u)
%init: 1 :Cytosol PER(id~1,cki!2,P1~p,cry!1,P2~u), CRY(id~2,per!1,clk), CKI(per
    !2)
%init: 7 :Cytosol PER(id~2,cki!2,P1~p,cry!1,P2~u), CKI(per!2), CRY(id~2,per!1,
    clk)
%init: 17 :Cytosol PER(id~1,cki,P1~p,cry!1,P2~p), CRY(id~1,per!1,clk)
%init: 11 :Cytosol PER(id~1,cki!1,P1~p,cry!2,P2~p), CRY(id~1,clk,per!2), CKI(per
    !1)
%init: 10 :Cytosol PER(id~1,cki,P1~p,cry!1,P2~p), CRY(id~2,per!1,clk)
%init: 7 :Cytosol PER(id~1,cki!1,P1~p,cry!2,P2~p), CRY(id~2,clk,per!2), CKI(per
    !1)

### Simulation:
#%obs: 'PER1 mRNA' mRNA(enc~PER1)
#%obs: 'PER2 mRNA' mRNA(enc~PER2)
#%obs: 'CRY1 mRNA' mRNA(enc~CRY1)
#%obs: 'CRY2 mRNA' mRNA(enc~CRY2)
%obs: 'PER1' PER(id~1)
%obs: 'PER2' PER(id~2)
%obs: 'CRY1' CRY(id~1)
%obs: 'CRY2' CRY(id~2)

### Perturbations:
%mod: [T]>12 do $UPDATE 'light' 0.00
%mod: [T]>24 do $UPDATE 'light' 0.000339
%mod: [T]>36 do $UPDATE 'light' 0.00
%mod: [T]>48 do $UPDATE 'light' 0.000339
%mod: [T]>60 do $UPDATE 'light' 0.00
%mod: [T]>72 do $UPDATE 'light' 0.000339
%mod: [T]>84 do $UPDATE 'light' 0.00
%mod: [T]>96 do $UPDATE 'light' 0.000339
%mod: [T]>108 do $UPDATE 'light' 0.00
%mod: [T]>120 do $UPDATE 'light' 0.000339
%mod: [T]>132 do $UPDATE 'light' 0.00
%mod: [T]>144 do $UPDATE 'light' 0.000339
```

```
%mod: [T]>156 do $UPDATE 'light' 0.00
%mod: [T]>168 do $UPDATE 'light' 0.000339
%mod: [T]>180 do $UPDATE 'light' 0.00
%mod: [T]>192 do $UPDATE 'light' 0.000339
%mod: [T]>204 do $UPDATE 'light' 0.00
%mod: [T]>216 do $UPDATE 'light' 0.000339
%mod: [T]>228 do $UPDATE 'light' 0.00
%mod: [T]>240 do $UPDATE 'light' 0.000339
%mod: [T]>252 do $UPDATE 'light' 0.00
```

A.7 The High-Osmolarity Glycerol Web in Yeast (Non-Spatial)

```
# John Wilson-Kanamori and Peter Krenn
#
# Model of the HOG pathway in Yeast (version 11-2012)
#
# Based on work by Peter Krenn (peter.krenn@edu.uni-graz.at).
# Assembly and experimental validation of a rule based model for the high
    osmolarity glycerol (HOG) web in yeast.


# Rules:
#######
# Chapter 'MAPK System'
'Ssk1Dimer_AutPhosph_Ssk2' Ssk1(Localiz~cyto,Dimer!1,Asp554~u),Ssk2(294_413!2,
    Thr1460~u,Localiz~cyto),Ssk1(Dimer!1,Ssk2_22!2,Asp554~u,Localiz~cyto) ->
    Ssk1(Localiz~cyto,Dimer!1,Asp554~u),Ssk2(294_413!2,Thr1460~p,Localiz~cyto),
    Ssk1(Dimer!1,Ssk2_22!2,Asp554~u,Localiz~cyto) @ 10
# 'Ssk1Dimer_Phosph_Ssk2 Ref2' Ssk1(Localiz~cyto,Dimer!1,Asp554~u),Ssk1(Dimer!1,
    Ssk2_22!2,Asp554~u,Localiz~cyto),Ssk2(294_413!2,Thr1460~u,Localiz~cyto,pbs2
    !3),Pbs2(ssk44_57!3) -> Ssk1(Localiz~cyto,Dimer!1,Asp554~u),Ssk1(Dimer!1,
    Ssk2_22!2,Asp554~u,Localiz~cyto),Ssk2(294_413!2,Thr1460~p,Localiz~cyto,pbs2
    !3),Pbs2(ssk44_57!3) @ 1.0 # Inactive rule
# 'Ssk2_Dephosph' Ssk2(Thr1460~p) -> Ssk2(Thr1460~u) @ 0.001 # Inactive rule
'Ptc1aNbp2_Dephos_Ssk2' Ptc1(Localiz~cyto,P~a,Nbp2!3),Nbp2(Localiz~cyto,Ptc1!3,
    Pbs2!2),Pbs2(Nbp2!2,ssk44_57!1,Localiz~cyto),Ssk2(pbs2!1,Localiz~cyto,
    Thr1460~p) -> Ptc1(Localiz~cyto,P~a,Nbp2!2),Nbp2(Localiz~cyto,Ptc1!2,Pbs2!3)
    ,Pbs2(Nbp2!3,ssk44_57!1,Localiz~cyto),Ssk2(pbs2!1,Localiz~cyto,Thr1460~u) @
    5
'Ssk2cyto_Binds_Pbs2cyto' Ssk2(pbs2,Localiz~cyto),Pbs2(ssk44_57,Localiz~cyto)
    <-> Ssk2(pbs2!1,Localiz~cyto),Pbs2(ssk44_57!1,Localiz~cyto) @ 0.00005,0.1
# 'Ssk2cyto_Binds_Pbs2cyto Ref1' Pbs2(ssk44_57,Localiz~cyto),Ssk2(pbs2,Localiz~
    cyto,Thr1460~u) <-> Pbs2(ssk44_57!1,Localiz~cyto),Ssk2(pbs2!1,Localiz~cyto,
    Thr1460~u) @ 0.000005,0.01 # Inactive rule
# 'Ssk2membr_Binds_Pbs2membr' Pbs2(ssk44_57,Localiz~membr),Ssk2(Localiz~membr,
    pbs2) <-> Pbs2(ssk44_57!1,Localiz~membr),Ssk2(Localiz~membr,pbs2!1) @
    0.00005,0.01 # Inactive rule
'Ssk2_Phosph_Pbs2Thr' Ssk2(pbs2!1,Thr1460~p),Pbs2(ssk44_57!1,Thr518~u) -> Ssk2(
    pbs2!1,Thr1460~p),Pbs2(ssk44_57!1,Thr518~p) @ 10
'Ssk2_Phosph_Pbs2Ser' Ssk2(pbs2!1,Thr1460~p),Pbs2(ssk44_57!1,Ser514~u) -> Ssk2(
    pbs2!1,Thr1460~p),Pbs2(ssk44_57!1,Ser514~p) @ 10
# 'Ssk2_Phosph_Pbs2SerThr' Ssk2(pbs2!1,Thr1460~p),Pbs2(ssk44_57!1,Thr518~u,
```

```
    Ser514~u) -> Ssk2(pbs2!1,Thr1460~p),Pbs2(ssk44_57!1,Thr518~p,Ser514~p) @ 10
    # Inactive rule
'Ssk1Dimer_AutPhosph_Ssk22cyt' Ssk1(Asp554~u,Dimer!1,Localiz~cyto),Ssk1(Asp554~u
    ,Dimer!1,Ssk2_22!2,Localiz~cyto),Ssk22(98_179!2,Thr1460~u,Localiz~cyto) ->
    Ssk1(Asp554~u,Dimer!1,Localiz~cyto),Ssk1(Asp554~u,Dimer!1,Ssk2_22!2,Localiz~
    cyto),Ssk22(98_179!2,Thr1460~p,Localiz~cyto) @ 10
# 'Ssk1Dimercyt_Phosph_Ssk22cyt Ref2' Ssk1(Asp554~u,Dimer!1,Localiz~cyto),Ssk22
    (98_179!2,Thr1460~u,Localiz~cyto,pbs2!3),Ssk1(Asp554~u,Dimer!1,Ssk2_22!2,
    Localiz~cyto),Pbs2(ssk44_57!3) -> Ssk1(Asp554~u,Dimer!1,Localiz~cyto),Ssk22
    (98_179!2,Thr1460~p,Localiz~cyto,pbs2!3),Ssk1(Asp554~u,Dimer!1,Ssk2_22!2,
    Localiz~cyto),Pbs2(ssk44_57!3) @ 0.1 # Inactive rule
'Ssk22cyto_Binds_Pbs2cyto' Ssk22(pbs2,Localiz~cyto),Pbs2(ssk44_57,Localiz~cyto)
    <-> Ssk22(pbs2!1,Localiz~cyto),Pbs2(ssk44_57!1,Localiz~cyto) @ 0.00005,0.1
# 'Ssk22cyto_Binds_Pbs2cyto Ref1' Pbs2(ssk44_57,Localiz~cyto),Ssk22(pbs2,Localiz
    ~cyto,Thr1460~u) <-> Pbs2(ssk44_57!1,Localiz~cyto),Ssk22(pbs2!1,Localiz~cyto
    ,Thr1460~u) @ 0.000005,0.01 # Inactive rule
# 'Ssk22membr_Binds_Pbs2membr' Ssk22(pbs2,Localiz~membr),Pbs2(ssk44_57,Localiz~
    membr) <-> Ssk22(pbs2!1,Localiz~membr),Pbs2(ssk44_57!1,Localiz~membr) @
    0.00005,0.01 # Inactive rule
'Ssk22_Phosph_Pbs2Thr' Ssk22(pbs2!1,Thr1460~p),Pbs2(ssk44_57!1,Thr518~u) ->
    Ssk22(pbs2!1,Thr1460~p),Pbs2(ssk44_57!1,Thr518~p) @ 10
'Ssk22_Phosph_Pbs2Ser' Ssk22(pbs2!1,Thr1460~p),Pbs2(ssk44_57!1,Ser514~u) ->
    Ssk22(pbs2!1,Thr1460~p),Pbs2(ssk44_57!1,Ser514~p) @ 10
# 'Ssk22_Phosph_Pbs2SerThr' Ssk22(pbs2!1,Thr1460~p),Pbs2(ssk44_57!1,Thr518~u,
    Ser514~u) -> Ssk22(pbs2!1,Thr1460~p),Pbs2(ssk44_57!1,Thr518~p,Ser514~p) @ 10
     # Inactive rule
# 'Ssk22_Dephosph' Ssk22(Thr1460~p) -> Ssk22(Thr1460~u) @ 0.001 # Inactive rule
'Ptc1aNbp2_Dephos_Ssk22' Nbp2(Localiz~cyto,Ptc1!3,Pbs2!2),Ptc1(Localiz~cyto,P~a,
    Nbp2!3),Pbs2(Nbp2!2,ssk44_57!1,Localiz~cyto),Ssk22(pbs2!1,Thr1460~p,Localiz~
    cyto) -> Nbp2(Localiz~cyto,Ptc1!2,Pbs2!3),Ptc1(Localiz~cyto,P~a,Nbp2!2),Pbs2
    (Nbp2!3,ssk44_57!1,Localiz~cyto),Ssk22(pbs2!1,Thr1460~u,Localiz~cyto) @ 5
# 'T_Ste11_Phosph_Pbs2Thr' Ste11(Sho1!1,Ser281Ser285Thr286~ppp,Localiz~membr),
    Pbs2(sho91_102!2,Thr518~u,Localiz~membr),Sho1(SH3_342_346!1,SH3_338!2,
    Localiz~membr,X) -> Ste11(Sho1!1,Ser281Ser285Thr286~ppp,Localiz~membr),Pbs2(
    sho91_102!2,Thr518~p,Localiz~membr),Sho1(SH3_342_346!1,SH3_338!2,Localiz~
    membr,X) @ 10 # Inactive rule
# 'T_Ste11_Phosph_Pbs2Thr Ref2' Ste11(Sho1!1,Ser281Ser285Thr286~ppp,Localiz~
    membr,SAM!_),Pbs2(sho91_102!2,Thr518~u,Localiz~membr),Sho1(SH3_342_346!1,
    SH3_338!2,Localiz~membr,X~a) -> Ste11(Sho1!1,Ser281Ser285Thr286~ppp,Localiz~
    membr,SAM!_),Pbs2(sho91_102!2,Thr518~p,Localiz~membr),Sho1(SH3_342_346!1,
    SH3_338!2,Localiz~membr,X~a) @ 10 # Inactive rule
# 'T_Ste11_Phosph_Pbs2Thr Ref3' Ste11(Sho1!1,Ser281Ser285Thr286~ppp,Localiz~
```

```
    membr),Pbs2(sho91_102!2,Thr518~u,Localiz~membr),Sho1(SH3_342_346!1,SH3_338
    !2,Localiz~membr,X~a) -> Ste11(Sho1!1,Ser281Ser285Thr286~ppp,Localiz~membr),
    Pbs2(sho91_102!2,Thr518~p,Localiz~membr),Sho1(SH3_342_346!1,SH3_338!2,
    Localiz~membr,X~a) @ 10 # Inactive rule
# 'T_Ste11_Phosph_Pbs2Ser' Ste11(Sho1!1,Ser281Ser285Thr286~ppp,Localiz~membr),
    Pbs2(Ser514~u,sho91_102!2,Localiz~membr),Sho1(SH3_342_346!1,SH3_338!2,
    Localiz~membr,X) -> Ste11(Sho1!1,Ser281Ser285Thr286~ppp,Localiz~membr),Pbs2(
    Ser514~p,sho91_102!2,Localiz~membr),Sho1(SH3_342_346!1,SH3_338!2,Localiz~
    membr,X) @ 10 # Inactive rule
# 'T_Ste11_Phosph_Pbs2Ser Ref2' Sho1(SH3_342_346!1,SH3_338!2,Localiz~membr,X~a),
    Pbs2(Ser514~u,sho91_102!2,Localiz~membr),Ste11(Sho1!1,Ser281Ser285Thr286~ppp
    ,Localiz~membr,SAM!_) -> Sho1(SH3_342_346!1,SH3_338!2,Localiz~membr,X~a),
    Pbs2(Ser514~p,sho91_102!2,Localiz~membr),Ste11(Sho1!1,Ser281Ser285Thr286~ppp
    ,Localiz~membr,SAM!_) @ 10 # Inactive rule
# 'T_Ste11_Phosph_Pbs2Ser Ref3' Ste11(Sho1!1,Ser281Ser285Thr286~ppp,Localiz~
    membr),Sho1(SH3_342_346!1,SH3_338!2,Localiz~membr,X~a),Pbs2(Ser514~u,
    sho91_102!2,Localiz~membr) -> Ste11(Sho1!1,Ser281Ser285Thr286~ppp,Localiz~
    membr),Sho1(SH3_342_346!1,SH3_338!2,Localiz~membr,X~a),Pbs2(Ser514~p,
    sho91_102!2,Localiz~membr) @ 10 # Inactive rule
'Pbs2pp_Unbind_Sho1' Sho1(SH3_338!1,X),Pbs2(sho91_102!1,Thr518~p,Ser514~p) ->
    Sho1(SH3_338,X),Pbs2(sho91_102,Thr518~p,Ser514~p) @ 10
'Ste11membr_Binds_Pbs2membr' Pbs2(Localiz~membr,Ste11),Ste11(Localiz~membr,Pbs2)
     <-> Pbs2(Localiz~membr,Ste11!1),Ste11(Localiz~membr,Pbs2!1) @ 0.0,0.1
'Ste11membr_Binds_Pbs2Umembr Ref1' Ste11(Localiz~membr,Ser281Ser285Thr286~ppp,
    Pbs2),Pbs2(Localiz~membr,Ste11,sho91_102,Thr518~u,Ser514~u) <-> Ste11(
    Localiz~membr,Ser281Ser285Thr286~ppp,Pbs2!1),Pbs2(Localiz~membr,Ste11!1,
    sho91_102,Thr518~u,Ser514~u) @ 0.00000005,0.1
'Ste11membr_Binds_Pbs2Umembr Ref2' Pbs2(Localiz~membr,Ste11,Ser514~p,Thr518~u,
    sho91_102),Ste11(Localiz~membr,Pbs2,Ser281Ser285Thr286~ppp) <-> Pbs2(Localiz
    ~membr,Ste11!1,Ser514~p,Thr518~u,sho91_102),Ste11(Localiz~membr,Pbs2!1,
    Ser281Ser285Thr286~ppp) @ 0.000005,0.1
'Ste11membr_Binds_Pbs2Umembr Ref3' Pbs2(Localiz~membr,Ste11,Thr518~p,Ser514~u,
    sho91_102),Ste11(Localiz~membr,Pbs2,Ser281Ser285Thr286~ppp) <-> Pbs2(Localiz
    ~membr,Ste11!1,Thr518~p,Ser514~u,sho91_102),Ste11(Localiz~membr,Pbs2!1,
    Ser281Ser285Thr286~ppp) @ 0.000005,0.1
# 'Ste11membr_Phosph_Pbs2UmembrThrSer' Ste11(Localiz~membr,Ser281Ser285Thr286~
    ppp,Pbs2!1),Pbs2(Localiz~membr,Ste11!1,AA389~m,sho91_102,Thr518~u,Ser514~u)
    -> Ste11(Localiz~membr,Ser281Ser285Thr286~ppp,Pbs2!1),Pbs2(Localiz~membr,
    Ste11!1,AA389~m,sho91_102,Thr518~p,Ser514~p) @ 1 # Inactive rule
'Ste11membr_Phosph_Pbs2membrThr' Pbs2(Localiz~membr,Ste11!1,Thr518~u),Ste11(
    Localiz~membr,Ser281Ser285Thr286~ppp,Pbs2!1) -> Pbs2(Localiz~membr,Ste11!1,
    Thr518~p),Ste11(Localiz~membr,Ser281Ser285Thr286~ppp,Pbs2!1) @ 10
```

```
'Ste11membr_Phosph_Pbs2membrSer' Pbs2(Localiz~membr,Ste11!1,Ser514~u),Ste11(
    Localiz~membr,Ser281Ser285Thr286~ppp,Pbs2!1) -> Pbs2(Localiz~membr,Ste11!1,
    Ser514~p),Ste11(Localiz~membr,Ser281Ser285Thr286~ppp,Pbs2!1) @ 10
'Ste11pppcyto_Binds_Pbs2cyto' Pbs2(Localiz~cyto,Ste11),Ste11(Localiz~cyto,Pbs2)
    <-> Pbs2(Localiz~cyto,Ste11!1),Ste11(Localiz~cyto,Pbs2!1) @ 0.0,0.1
'Ste11pppcyto_Binds_Pbs2cyto Ref1' Pbs2(Localiz~cyto,Ste11,sho91_102),Ste11(
    Localiz~cyto,Ser281Ser285Thr286~ppp,Pbs2) <-> Pbs2(Localiz~cyto,Ste11!1,
    sho91_102),Ste11(Localiz~cyto,Ser281Ser285Thr286~ppp,Pbs2!1) @
    0.00000005,0.1
'Ste11pppcyto_Binds_Pbs2cyto Ref2' Ste11(Localiz~cyto,Pbs2,Ser281Ser285Thr286~
    ppp),Pbs2(Localiz~cyto,Ste11,Thr518~u,Ser514~p,sho91_102) <-> Ste11(Localiz~
    cyto,Pbs2!1,Ser281Ser285Thr286~ppp),Pbs2(Localiz~cyto,Ste11!1,Thr518~u,
    Ser514~p,sho91_102) @ 0.000005,0.1
'Ste11pppcyto_Binds_Pbs2cyto Ref3' Ste11(Localiz~cyto,Pbs2,Ser281Ser285Thr286~
    ppp),Pbs2(Localiz~cyto,Ste11,Thr518~p,Ser514~u,sho91_102) <-> Ste11(Localiz~
    cyto,Pbs2!1,Ser281Ser285Thr286~ppp),Pbs2(Localiz~cyto,Ste11!1,Thr518~p,
    Ser514~u,sho91_102) @ 0.000005,0.1
'Ste11cyto_Phosph_Pbs2cytoSer' Pbs2(Localiz~cyto,Ste11!1,Ser514~u),Ste11(Localiz
    ~cyto,Ser281Ser285Thr286~ppp,Pbs2!1) -> Pbs2(Localiz~cyto,Ste11!1,Ser514~p),
    Ste11(Localiz~cyto,Ser281Ser285Thr286~ppp,Pbs2!1) @ 10
'Ste11cyto_Phosph_Pbs2cytoThr' Pbs2(Localiz~cyto,Ste11!1,Thr518~u),Ste11(Localiz
    ~cyto,Pbs2!1,Ser281Ser285Thr286~ppp) -> Pbs2(Localiz~cyto,Ste11!1,Thr518~p),
    Ste11(Localiz~cyto,Pbs2!1,Ser281Ser285Thr286~ppp) @ 10
# 'Pbs2_Deph_Thr' Pbs2(Thr518~p) -> Pbs2(Thr518~u) @ 0.0000001 # Inactive rule
# 'Pbs2_Deph_Ser' Pbs2(Ser514~p) -> Pbs2(Ser514~u) @ 0.0000001 # Inactive rule
'Hog1_Binds_Pbs2_1' Pbs2(Localiz~cyto,HBD-I,Thr518,Ser514),Hog1(Localiz~cyto,
    Thr174,Tyr176,CD~pbd1) <-> Pbs2(Localiz~cyto,HBD-I!1,Thr518,Ser514),Hog1(
    Localiz~cyto,Thr174,Tyr176,CD~pbd1!1) @ 0.0,0.5
'Hog1_Binds_Pbs2_1 Ref2' Hog1(Localiz~cyto,Thr174~u,Tyr176~u,CD~pbd1),Pbs2(
    Localiz~cyto,Ser514~p,Thr518~p,HBD-I) <-> Hog1(Localiz~cyto,Thr174~u,Tyr176~
    u,CD~pbd1!1),Pbs2(Localiz~cyto,Ser514~p,Thr518~p,HBD-I!1) @ 0.000005,0.5
'Hog1_Binds_Pbs2_1 Ref3' Hog1(Localiz~cyto,Thr174~p,Tyr176~u,CD~pbd1),Pbs2(
    Localiz~cyto,HBD-I,Ser514~p,Thr518~p) <-> Hog1(Localiz~cyto,Thr174~p,Tyr176~
    u,CD~pbd1!1),Pbs2(Localiz~cyto,HBD-I!1,Ser514~p,Thr518~p) @ 0.00005,0.5
'Hog1_Binds_Pbs2_1 Ref4' Hog1(Localiz~cyto,Thr174~u,Tyr176~p,CD~pbd1),Pbs2(
    Localiz~cyto,HBD-I,Ser514~p,Thr518~p) <-> Hog1(Localiz~cyto,Thr174~u,Tyr176~
    p,CD~pbd1!1),Pbs2(Localiz~cyto,HBD-I!1,Ser514~p,Thr518~p) @ 0.00005,0.5
'Hog1_Binds_Pbs2_2' Pbs2(Localiz~cyto,HBD-I!1,Thr518,Ser514),Hog1(Localiz~cyto,
    Thr174,Tyr176,CD~pbd1!1) -> Pbs2(Localiz~cyto,HBD-I!1,Thr518,Ser514),Hog1(
    Localiz~cyto,Thr174,Tyr176,CD~pbd2!1) @ 20
'Hog1_Unbind_Pbs2_pbd2' Pbs2(Localiz~cyto,HBD-I!1,Thr518,Ser514),Hog1(Localiz~
    cyto,Thr174,Tyr176,CD~pbd2!1) -> Pbs2(Localiz~cyto,HBD-I,Thr518,Ser514),Hog1
```

```
    (Localiz~cyto,Thr174,Tyr176,CD~pbd1) @ 0.1
'Hog1cyto_Binds_Pbs2_Direct' Pbs2(Localiz~cyto,HBD-I),Hog1(Localiz~cyto,CD~pbd1)
     <-> Pbs2(Localiz~cyto,HBD-I!1),Hog1(Localiz~cyto,CD~pbd2!1) @ 0.0,0.1
'Hog1cyto_Binds_Pbs2_Direct Ref2' Pbs2(Localiz~cyto,HBD-I,Thr518~p,Ser514~p),
    Hog1(Localiz~cyto,Tyr176~u,Thr174~u,CD~pbd1) <-> Pbs2(Localiz~cyto,HBD-I!1,
    Thr518~p,Ser514~p),Hog1(Localiz~cyto,Tyr176~u,Thr174~u,CD~pbd2!1) @
    0.0000005,0.1
'Hog1cyto_Binds_Pbs2_Direct Ref3' Hog1(Localiz~cyto,Thr174~u,Tyr176~p,CD~pbd1),
    Pbs2(Localiz~cyto,HBD-I,Thr518~p,Ser514~p) <-> Hog1(Localiz~cyto,Thr174~u,
    Tyr176~p,CD~pbd2!1),Pbs2(Localiz~cyto,HBD-I!1,Thr518~p,Ser514~p) @
    0.00005,0.1
'Hog1cyto_Binds_Pbs2_Direct Ref4' Hog1(Localiz~cyto,Thr174~p,Tyr176~u,CD~pbd1),
    Pbs2(Localiz~cyto,HBD-I,Thr518~p,Ser514~p) <-> Hog1(Localiz~cyto,Thr174~p,
    Tyr176~u,CD~pbd2!1),Pbs2(Localiz~cyto,HBD-I!1,Thr518~p,Ser514~p) @
    0.00005,0.1
'Pbs2_Phosph_Hog1Thr' Pbs2(HBD-I!1,Ser514~p,Thr518~p,Localiz~cyto),Hog1(Thr174~u
    ,Localiz~cyto,CD~pbd2!1) -> Pbs2(HBD-I!1,Ser514~p,Thr518~p,Localiz~cyto),
    Hog1(Thr174~p,Localiz~cyto,CD~pbd2!1) @ 5
'Pbs2_Phosph_Hog1Tyr' Pbs2(HBD-I!1,Ser514~p,Thr518~p,Localiz~cyto),Hog1(Localiz~
    cyto,Tyr176~u,CD~pbd2!1) -> Pbs2(HBD-I!1,Ser514~p,Thr518~p,Localiz~cyto),
    Hog1(Localiz~cyto,Tyr176~p,CD~pbd2!1) @ 10
# 'Pbs2_Phosph_Hog1Proc' Pbs2(HBD-I!1,Ser514~p,Thr518~p,Localiz~cyto),Hog1(
    Thr174~u,Localiz~cyto,Tyr176~u,CD~pbd2!1) -> Pbs2(HBD-I!1,Ser514~p,Thr518~p,
    Localiz~cyto),Hog1(Thr174~p,Localiz~cyto,Tyr176~p,CD~pbd2!1) @ 10 # Inactive
     rule
'Pbs2_Phosph_Hog1Thr_Pbd1' Pbs2(HBD-I!1,Ser514~p,Thr518~p,Localiz~cyto),Hog1(
    Thr174~u,Localiz~cyto,CD~pbd1!1) -> Pbs2(HBD-I!1,Ser514~p,Thr518~p,Localiz~
    cyto),Hog1(Thr174~p,Localiz~cyto,CD~pbd1!1) @ 0.05
'Pbs2_Phosph_Hog1Tyr_Pbd1' Pbs2(HBD-I!1,Ser514~p,Thr518~p,Localiz~cyto),Hog1(
    Localiz~cyto,Tyr176~u,CD~pbd1!1) -> Pbs2(HBD-I!1,Ser514~p,Thr518~p,Localiz~
    cyto),Hog1(Localiz~cyto,Tyr176~p,CD~pbd1!1) @ 0.1
'Ptc1_Binds_Npb2' Ptc1(Nbp2,P),Nbp2(Ptc1) <-> Ptc1(Nbp2!1,P),Nbp2(Ptc1!1) @
    0.0001,0.00001
# 'Ptc1_Binds_Npb2 Ref1' Nbp2(Ptc1),Ptc1(Nbp2,P~a) <-> Nbp2(Ptc1!1),Ptc1(Nbp2!1,
    P~a) @ 1.0,1.0 # Inactive rule
'Nbp2_Binds_Pbs2' Pbs2(Nbp2,Localiz~cyto),Nbp2(Pbs2,Localiz~cyto) <-> Pbs2(Nbp2
    !1,Localiz~cyto),Nbp2(Pbs2!1,Localiz~cyto) @ 0.0,1.0
'Nbp2_Binds_Pbs2pp Ref1' Nbp2(Pbs2,Localiz~cyto),Pbs2(Nbp2,Localiz~cyto,Ser514~p
    ,Thr518~p) <-> Nbp2(Pbs2!1,Localiz~cyto),Pbs2(Nbp2!1,Localiz~cyto,Ser514~p,
    Thr518~p) @ 0.0001,1.0
'Ptc1p_Dephosph_Pbs2Ser' Ptc1(Localiz~cyto,P~a,Nbp2!1),Nbp2(Ptc1!1,Localiz~cyto,
    Pbs2!2),Pbs2(Nbp2!2,Ser514~p,Localiz~cyto) -> Ptc1(Localiz~cyto,P~a,Nbp2!1),
```

```
    Nbp2(Ptc1!1,Localiz~cyto,Pbs2!2),Pbs2(Nbp2!2,Ser514~u,Localiz~cyto) @ 5
'Ptc1p_Dephosph_Pbs2Thr' Ptc1(Nbp2!1,Localiz~cyto,P~a),Nbp2(Pbs2!2,Ptc1!1,
    Localiz~cyto),Pbs2(Thr518~p,Nbp2!2,Localiz~cyto) -> Ptc1(Nbp2!1,Localiz~cyto
    ,P~a),Nbp2(Pbs2!2,Ptc1!1,Localiz~cyto),Pbs2(Thr518~u,Nbp2!2,Localiz~cyto) @
    5
# 'Ptc1p_Dephosph_Pbs2SerThr' Ptc1(Nbp2!1,Localiz~cyto,P~a),Nbp2(Pbs2!2,Ptc1!1,
    Localiz~cyto),Pbs2(Thr518~p,Nbp2!2,Localiz~cyto,Ser514~p) -> Ptc1(Nbp2!1,
    Localiz~cyto,P~a),Nbp2(Pbs2!2,Ptc1!1,Localiz~cyto),Pbs2(Thr518~u,Nbp2!2,
    Localiz~cyto,Ser514~u) @ 10 # Inactive rule
# 'Ptc1_Dephos_Hog1Thr_cyto' Nbp2(Ptc1!1,Pbs2!2,Localiz~cyto),Pbs2(Localiz~cyto,
    Nbp2!2,HBD-I!3),Hog1(Localiz~cyto,Thr174~p,CD!3),Ptc1(Nbp2!1,Localiz~cyto,P)
     -> Nbp2(Ptc1!1,Pbs2!2,Localiz~cyto),Pbs2(Localiz~cyto,Nbp2!2,HBD-I!3),Hog1(
    Localiz~cyto,Thr174~u,CD!3),Ptc1(Nbp2!1,Localiz~cyto,P) @ 10 # Inactive rule
'Ptc1_Dephos_Hog1Thr_cyto Ref1' Nbp2(Ptc1!1,Pbs2!2,Localiz~cyto),Pbs2(Localiz~
    cyto,Nbp2!2,HBD-I!3),Hog1(Localiz~cyto,Thr174~p,CD!3),Ptc1(Nbp2!1,Localiz~
    cyto,P~a) -> Nbp2(Ptc1!1,Pbs2!2,Localiz~cyto),Pbs2(Localiz~cyto,Nbp2!2,HBD-I
    !3),Hog1(Localiz~cyto,Thr174~u,CD!3),Ptc1(Nbp2!1,Localiz~cyto,P~a) @ 10
'Ptc1a_Binds_Hog1_cyto' Ptc1(Localiz~cyto,Hog,P~a),Hog1(Thr174,Localiz~cyto,
    Ptc23) <-> Ptc1(Localiz~cyto,Hog!1,P~a),Hog1(Thr174,Localiz~cyto,Ptc23!1) @
    0.0,0.1
'Ptc1a_Binds_Hog1 Ref2' Hog1(Localiz~cyto,Ptc23,Thr174~p),Ptc1(Localiz~cyto,Hog,
    P~a,Nbp2) <-> Hog1(Localiz~cyto,Ptc23!1,Thr174~p),Ptc1(Localiz~cyto,Hog!1,P~
    a,Nbp2) @ 0.000005,0.1
'Ptc1a_Binds_Hog1_nuc' Ptc1(Localiz~nuc,Hog,P),Hog1(Thr174,Localiz~nuc,Ptc23)
    <-> Ptc1(Localiz~nuc,Hog!1,P),Hog1(Thr174,Localiz~nuc,Ptc23!1) @ 0.0,0.1
'Ptc1a_Binds_Hog1_nuc Ref1' Ptc1(Localiz~nuc,Hog,P~a,Nbp2),Hog1(Localiz~nuc,
    Ptc23,Thr174~p,TF) <-> Ptc1(Localiz~nuc,Hog!1,P~a,Nbp2),Hog1(Localiz~nuc,
    Ptc23!1,Thr174~p,TF) @ 0.000005,0.1
'Ptc1a_Binds_Hog1_nuc Ref2' Ptc1(Localiz~nuc,Hog,Nbp2,P~a),Hog1(Localiz~nuc,
    Ptc23,Thr174~p,TF!_) <-> Ptc1(Localiz~nuc,Hog!1,Nbp2,P~a),Hog1(Localiz~nuc,
    Ptc23!1,Thr174~p,TF!_) @ 0.0000001,0.1
'Ptc1_DephosDirect_Hog1' Ptc1(Localiz~cyto,Hog!1,P~a),Hog1(Thr174~p,Localiz~cyto
    ,Ptc23!1) -> Ptc1(Localiz~cyto,Hog!1,P~a),Hog1(Thr174~u,Localiz~cyto,Ptc23
    !1) @ 1
'Ptc1_DephosDirect_Hog1_nuc' Ptc1(Localiz~nuc,Hog!1,P~a),Hog1(Thr174~p,Localiz~
    nuc,Ptc23!1) -> Ptc1(Localiz~nuc,Hog!1,P~a),Hog1(Thr174~u,Localiz~nuc,Ptc23
    !1) @ 1
'Hog1p_Activat_Ptc1' Hog1(Localiz,Thr174~p,Tyr176~p),Ptc1(Localiz,P~ia) -> Hog1(
    Localiz,Thr174~p,Tyr176~p),Ptc1(Localiz,P~a) @ 0.000005
# 'Ptc1_Deact' Ptc1(Localiz,P~a) -> Ptc1(Localiz,P~ia) @ 0.000005 # Inactive
    rule
# 'Hog1pp_Activat_Ptc23' Hog1(Localiz~cyto,Thr174~p,Tyr176~p),Ptc23(Localiz~cyto
```

```
    ,P~ia) -> Hog1(Localiz~cyto,Thr174~p,Tyr176~p),Ptc23(Localiz~cyto,P~a) @
    0.0000005 # Inactive rule
'Hog1pp_Binds_FeedbDummy' Hog1(Thr174,Tyr176,Localiz~cyto,Phosphat),
    FeedbackDummy(Hog1pp) <-> Hog1(Thr174,Tyr176,Localiz~cyto,Phosphat!1),
    FeedbackDummy(Hog1pp!1) @ 0.0,0.1
'Hog1pp_Binds_FeedbDummy Ref1' Hog1(Thr174~p,Tyr176~p,Localiz~cyto,Phosphat),
    FeedbackDummy(Hog1pp,X~ia) <-> Hog1(Thr174~p,Tyr176~p,Localiz~cyto,Phosphat
    !1),FeedbackDummy(Hog1pp!1,X~ia) @ 0.00000005,0.5
'Hog1pp_Activat_FeddbDummy' Hog1(Localiz~cyto,Thr174~p,Tyr176~p,Phosphat!1),
    FeedbackDummy(Hog1pp!1,X~ia) -> Hog1(Localiz~cyto,Thr174~p,Tyr176~p,Phosphat
    !1),FeedbackDummy(Hog1pp!1,X~a) @ 1
'FeedbDummy_Binds_Ptc23' FeedbackDummy(Ptc23,X),Ptc23(Localiz,P,Hog1) <->
    FeedbackDummy(Ptc23!1,X),Ptc23(Localiz,P,Hog1!1) @ 0.0,0.1
'FeedbDummy_Binds_Ptc23 Ref1' FeedbackDummy(Ptc23,X~a),Ptc23(Localiz~cyto,P~ia,
    Hog1) <-> FeedbackDummy(Ptc23!1,X~a),Ptc23(Localiz~cyto,P~ia,Hog1!1) @
    0.00005,0.1
# 'FeedbDummy_Binds_Ptc23 Ref2' FeedbackDummy(Ptc23,X~a),Ptc23(Localiz,Hog1,P~ia
    ) <-> FeedbackDummy(Ptc23!1,X~a),Ptc23(Localiz,Hog1!1,P~ia) @ 0.00005,0.1 #
    Inactive rule
'FeedbDummy_Act_Ptc23' FeedbackDummy(Ptc23!1,X~a),Ptc23(Localiz~cyto,P~ia,Hog1
    !1) -> FeedbackDummy(Ptc23!1,X~a),Ptc23(Localiz~cyto,P~a,Hog1!1) @ 10
# 'FeedbDummy_Act_Ptc23 Ref1' FeedbackDummy(Ptc23!1,X~a),Ptc23(Localiz~nuc,P~ia,
    Hog1!1) -> FeedbackDummy(Ptc23!1,X~a),Ptc23(Localiz~nuc,P~a,Hog1!1) @ 10 #
    Inactive rule
'FeedbackDummy_Inactivat' FeedbackDummy(X~a) -> FeedbackDummy(X~ia) @ 0.00000001
'Ptc23a_Binds_Hog1Cyto' Ptc23(P,Hog1,Localiz~cyto),Hog1(Ptc23,Localiz~cyto) <->
    Ptc23(P,Hog1!1,Localiz~cyto),Hog1(Ptc23!1,Localiz~cyto) @ 0.0,0.1
'Ptc23a_Binds_Hog1 Ref1' Ptc23(Localiz~cyto,P~a,Hog1),Hog1(Ptc23,Thr174,Localiz~
    cyto) <-> Ptc23(Localiz~cyto,P~a,Hog1!1),Hog1(Ptc23!1,Thr174,Localiz~cyto) @
     0.0000005,0.1
'Ptc23a_Binds_Hog1Nuc' Hog1(Ptc23,Localiz~nuc),Ptc23(P,Hog1,Localiz~nuc) <->
    Hog1(Ptc23!1,Localiz~nuc),Ptc23(P,Hog1!1,Localiz~nuc) @ 0.0,0.1
'Ptc23a_Binds_Hog1Nuc Ref1' Ptc23(Localiz~nuc,P~a,Hog1),Hog1(Ptc23,Localiz~nuc,
    TF,Thr174~p) <-> Ptc23(Localiz~nuc,P~a,Hog1!1),Hog1(Ptc23!1,Localiz~nuc,TF,
    Thr174~p) @ 0.00005,0.1
'Ptc23a_Binds_Hog1Nuc Ref2' Ptc23(Hog1,Localiz~nuc,P~a),Hog1(Ptc23,Localiz~nuc,
    TF!_,Thr174~p) <-> Ptc23(Hog1!1,Localiz~nuc,P~a),Hog1(Ptc23!1,Localiz~nuc,TF
    !_,Thr174~p) @ 0.000001,0.1
'Ptc23a_Deph_Hog1ThrCyto' Ptc23(Localiz~cyto,P~a,Hog1!1),Hog1(Ptc23!1,Thr174~p,
    Localiz~cyto) -> Ptc23(Localiz~cyto,P~a,Hog1!1),Hog1(Ptc23!1,Thr174~u,
    Localiz~cyto) @ 10
'Ptc23a_Desph_Hog1ThrNuc' Ptc23(Localiz~nuc,P~a,Hog1!1),Hog1(Ptc23!1,Thr174~p,
```

```
    Localiz~nuc) -> Ptc23(Localiz~nuc,P~a,Hog1!1),Hog1(Ptc23!1,Thr174~u,Localiz~
    nuc) @ 10
'Ptcs23_Binds_Pbs2p' Ptc23(Hog1,P,Localiz~cyto),Pbs2(Ptc23,Localiz~cyto) <->
    Ptc23(Hog1!1,P,Localiz~cyto),Pbs2(Ptc23!1,Localiz~cyto) @ 0.0,1
'Ptcs23_Binds_Pbs2p Ref1' Ptc23(Hog1,P~a,Localiz~cyto),Pbs2(Ptc23,Localiz~cyto,
    Ser514~p,Thr518~p) <-> Ptc23(Hog1!1,P~a,Localiz~cyto),Pbs2(Ptc23!1,Localiz~
    cyto,Ser514~p,Thr518~p) @ 0.0000005,1
'Ptc23_Dephosph_Pbs2' Ptc23(P~a,Hog1!1,Localiz~cyto),Pbs2(Ptc23!1,Thr518~p,
    Ser514~p,Localiz~cyto) -> Ptc23(P~a,Hog1!1,Localiz~cyto),Pbs2(Ptc23!1,Thr518
    ~u,Ser514~u,Localiz~cyto) @ 1
'Ptc23_Binds_Ssk2' Ptc23(Localiz~cyto,P,Hog1),Ssk2(Localiz~cyto,phosphatas) <->
    Ptc23(Localiz~cyto,P,Hog1!1),Ssk2(Localiz~cyto,phosphatas!1) @ 0.0,1
'Ptc23_Binds_Ssk2 Ref1' Ptc23(Localiz~cyto,P~a,Hog1),Ssk2(Localiz~cyto,
    phosphatas,Thr1460~p) <-> Ptc23(Localiz~cyto,P~a,Hog1!1),Ssk2(Localiz~cyto,
    phosphatas!1,Thr1460~p) @ 0.0000005,1
'Ptc23_Dephosph_Ssk2' Ptc23(Localiz~cyto,P~a,Hog1!1),Ssk2(phosphatas!1,Thr1460~p
    ,Localiz~cyto) -> Ptc23(Localiz~cyto,P~a,Hog1!1),Ssk2(phosphatas!1,Thr1460~u
    ,Localiz~cyto) @ 1
'Ptc23_Binds_Ssk22' Ptc23(Localiz~cyto,P,Hog1),Ssk22(Localiz~cyto,phosphatas)
    <-> Ptc23(Localiz~cyto,P,Hog1!1),Ssk22(Localiz~cyto,phosphatas!1) @ 0.0,1
'Ptc23_Binds_Ssk22 Ref1' Ptc23(Localiz~cyto,P~a,Hog1),Ssk22(Localiz~cyto,
    phosphatas,Thr1460~p) <-> Ptc23(Localiz~cyto,P~a,Hog1!1),Ssk22(Localiz~cyto,
    phosphatas!1,Thr1460~p) @ 0.0000005,1
'Ptc23_Dephosph_Ssk22' Ptc23(Localiz~cyto,P~a,Hog1!1),Ssk22(Thr1460~p,Localiz~
    cyto,phosphatas!1) -> Ptc23(Localiz~cyto,P~a,Hog1!1),Ssk22(Thr1460~u,Localiz
    ~cyto,phosphatas!1) @ 1
'Ptc23_Deact' Ptc23(Localiz,P~a) -> Ptc23(Localiz,P~ia) @ 0.0005
'Hog1ppcyto_Act_Ptpcyto' Ptp(Localiz~cyto,P~ia,mapk),Hog1(Localiz~cyto,Thr174~p,
    Tyr176~p) -> Ptp(Localiz~cyto,P~a,mapk),Hog1(Localiz~cyto,Thr174~p,Tyr176~p)
     @ 0.000005
'Hog1ppNuc_Act_PtpNuc' Ptp(Localiz~nuc,P~ia,mapk),Hog1(Localiz~nuc,Thr174~p,
    Tyr176~p) -> Ptp(Localiz~nuc,P~a,mapk),Hog1(Localiz~nuc,Thr174~p,Tyr176~p) @
     0.00005
# 'Ptpcyto_Deact' Ptp(Localiz~cyto,P~a,mapk) -> Ptp(Localiz~cyto,P~ia,mapk) @
    0.001 # Inactive rule
# 'PtpNuc_Deact' Ptp(Localiz~nuc,P~a,mapk) -> Ptp(Localiz~nuc,P~ia,mapk) @ 0.01
    # Inactive rule
'Ptp_Binds_Hog1_cyto' Ptp(Localiz~cyto,mapk,P),Hog1(Localiz~cyto,CD~pbd1) <->
    Ptp(Localiz~cyto,mapk!1,P),Hog1(Localiz~cyto,CD~pbd1!1) @ 0.0,0.1
'Ptp_Binds_Hog1_cyto Ref3' Ptp(Localiz~cyto,mapk,P~a),Hog1(Localiz~cyto,Tyr176~p
    ,CD~pbd1) <-> Ptp(Localiz~cyto,mapk!1,P~a),Hog1(Localiz~cyto,Tyr176~p,CD~
    pbd1!1) @ 0.000005,0.1
```

```
'Ptp_Binds_Hog1_cyto_pbd2' Ptp(Localiz~cyto,mapk!1,P~a),Hog1(Localiz~cyto,CD~
    pbd1!1) <-> Ptp(Localiz~cyto,mapk!1,P~a),Hog1(Localiz~cyto,CD~pbd2!1) @
    1,0.1
'Ptp_Dephosph_Hog1Tyr_cyto' Ptp(Localiz~cyto,mapk!1,P~a),Hog1(Tyr176~p,Localiz~
    cyto,CD~pbd2!1) -> Ptp(Localiz~cyto,mapk,P~a),Hog1(Tyr176~u,Localiz~cyto,CD~
    pbd1) @ 10
'Ptp_Binds_Hog1_nuc' Ptp(Localiz~nuc,mapk),Hog1(Localiz~nuc,Tyr176,CD~pbd1) <->
    Ptp(Localiz~nuc,mapk!1),Hog1(Localiz~nuc,Tyr176,CD~pbd1!1) @ 0.0,0.1
'Ptp_Binds_Hog1_nuc Ref1' Ptp(Localiz~nuc,mapk),Hog1(Localiz~nuc,Tyr176~p,TF,CD~
    pbd1) <-> Ptp(Localiz~nuc,mapk!1),Hog1(Localiz~nuc,Tyr176~p,TF,CD~pbd1!1) @
    0.000005,1.0
'Ptp_Binds_Hog1_nuc Ref2' Ptp(Localiz~nuc,mapk),Hog1(Localiz~nuc,TF!_,CD~pbd1,
    Tyr176~p) <-> Ptp(Localiz~nuc,mapk!1),Hog1(Localiz~nuc,TF!_,CD~pbd1!1,Tyr176
    ~p) @ 0.0000001,1.0
'Ptp_Binds_Hog1_nuc_pbd2' Ptp(Localiz~nuc,mapk!1),Hog1(Localiz~nuc,Tyr176,CD~
    pbd1!1) <-> Ptp(Localiz~nuc,mapk!1),Hog1(Localiz~nuc,Tyr176,CD~pbd2!1) @
    1,0.1
'Ptp_Dephosph_Hog1Tyr_nuc' Ptp(Localiz~nuc,mapk!1),Hog1(Tyr176~p,Localiz~nuc,CD~
    pbd2!1) -> Ptp(Localiz~nuc,mapk),Hog1(Tyr176~u,Localiz~nuc,CD~pbd1) @ 1
'Hog1ppmembr_Binds_Ste50membr' Hog1(Thr174,Ste50,Tyr176,Localiz~membr),Ste50(
    Localiz~membr,S155_196_202_248,Hog1) <-> Hog1(Thr174,Ste50!1,Tyr176,Localiz~
    membr),Ste50(Localiz~membr,S155_196_202_248,Hog1!1) @ 0.0,0.5
'Hog1ppmembr_Binds_Ste50membr Ref1' Hog1(Thr174~p,Ste50,Tyr176~p,Localiz~membr),
    Ste50(Localiz~membr,S155_196_202_248,Hog1,Sho1) <-> Hog1(Thr174~p,Ste50!1,
    Tyr176~p,Localiz~membr),Ste50(Localiz~membr,S155_196_202_248,Hog1!1,Sho1) @
    0.00007,0.5
'Hog1ppcyto_Binds_Ste50cyto' Hog1(Thr174,Ste50,Localiz~cyto,Tyr176),Ste50(
    Localiz~cyto,Hog1) <-> Hog1(Thr174,Ste50!1,Localiz~cyto,Tyr176),Ste50(
    Localiz~cyto,Hog1!1) @ 0.0,0.5
# 'Hog1ppcyto_Binds_Ste50cyto Ref1' Hog1(Ste50,Localiz~cyto,Tyr176~p,Thr174~p),
    Ste50(Localiz~cyto,Sho1,Hog1) <-> Hog1(Ste50!1,Localiz~cyto,Tyr176~p,Thr174~
    p),Ste50(Localiz~cyto,Sho1,Hog1!1) @ 0.000007,0.5 # Inactive rule
'Hog1ppcyto_Binds_Ste50cyto Ref2' Hog1(Thr174,Ste50,Localiz~cyto,Tyr176~p),Ste50
    (Localiz~cyto,Hog1) <-> Hog1(Thr174,Ste50!1,Localiz~cyto,Tyr176~p),Ste50(
    Localiz~cyto,Hog1!1) @ 0.000007,0.5
# 'Hog1pp_PhosphProc_Ste50' Hog1(Thr174~p,Ste50!1,Tyr176~p,Localiz),Ste50(
    Localiz,S155_196_202_248~u,Hog1!1) -> Hog1(Thr174~p,Ste50!1,Tyr176~p,Localiz
    ),Ste50(Localiz,S155_196_202_248~pppp,Hog1!1) @ 10 # Inactive rule
'Hog1pp_Phosph_Ste50' Hog1(Thr174~p,Ste50!1,Tyr176~p,Localiz),Ste50(Localiz,
    S155_196_202_248~u,Hog1!1) -> Hog1(Thr174~p,Ste50!1,Tyr176~p,Localiz),Ste50(
    Localiz,S155_196_202_248~p,Hog1!1) @ 10
'Hog1pp_Phosph_Ste50p' Hog1(Thr174~p,Ste50!1,Tyr176~p,Localiz),Ste50(Localiz,
```

```
    S155_196_202_248~p,Hog1!1) -> Hog1(Thr174~p,Ste50!1,Tyr176~p,Localiz),Ste50(
    Localiz,S155_196_202_248~pp,Hog1!1) @ 10
'Hog1pp_Phosph_Ste50pp' Hog1(Thr174~p,Ste50!1,Tyr176~p,Localiz),Ste50(Localiz,
    S155_196_202_248~pp,Hog1!1) -> Hog1(Thr174~p,Ste50!1,Tyr176~p,Localiz),Ste50
    (Localiz,S155_196_202_248~ppp,Hog1!1) @ 10
'Hog1pp_Phosph_Ste50ppp' Hog1(Thr174~p,Ste50!1,Tyr176~p,Localiz),Ste50(Localiz,
    S155_196_202_248~ppp,Hog1!1) -> Hog1(Thr174~p,Ste50!1,Tyr176~p,Localiz),
    Ste50(Localiz,S155_196_202_248~pppp,Hog1!1) @ 10
'Ste50pppp_ppp_Membr' Ste50(S155_196_202_248~pppp,Localiz~membr) -> Ste50(
    S155_196_202_248~ppp,Localiz~membr) @ 0.00001
'Ste50ppp_ppu_Membr' Ste50(S155_196_202_248~ppp,Localiz~membr) -> Ste50(
    S155_196_202_248~pp,Localiz~membr) @ 0.00001
'Ste50ppu_puu_Membr' Ste50(S155_196_202_248~pp,Localiz~membr) -> Ste50(
    S155_196_202_248~p,Localiz~membr) @ 0.00001
'Ste50puu_uuu_Membr' Ste50(S155_196_202_248~p,Localiz~membr) -> Ste50(
    S155_196_202_248~u,Localiz~membr) @ 0.00001
'Ste50pppp_ppp_Cyto' Ste50(S155_196_202_248~pppp,Localiz~cyto) -> Ste50(
    S155_196_202_248~ppp,Localiz~cyto) @ 0.00001
'Ste50ppp_ppu_Cyto' Ste50(S155_196_202_248~ppp,Localiz~cyto) -> Ste50(
    S155_196_202_248~pp,Localiz~cyto) @ 0.00001
'Ste50ppu_puu_Cyto' Ste50(S155_196_202_248~pp,Localiz~cyto) -> Ste50(
    S155_196_202_248~p,Localiz~cyto) @ 0.00001
'Ste50puu_uuu_Cyto' Ste50(S155_196_202_248~p,Localiz~cyto) -> Ste50(
    S155_196_202_248~u,Localiz~cyto) @ 0.00001

#######
# Chapter 'OsmoSystem'
'TriggerOsm' Trigger() -> Osm(r,Localiz~e) @ 0.
'TriggerOsm2' Trigger2() -> Osm(r,Localiz~e) @ 0
'OsmEx_Change_OsmInt' Osm(Localiz~e,r) -> Osm(Localiz~i,r) @ 0
'OsmEx_Del' Osm(r,Localiz~e) -> @ 0
'Sln1SensD_Act' Sln1_SensD(X~ia,Localiz~membr) -> Sln1_SensD(X~a,Localiz~membr)
    @ 0
'Osm_Binds_Sln1SensD' Osm(Localiz~e,r),Sln1_SensD(Osm) <-> Osm(Localiz~e,r!1),
    Sln1_SensD(Osm!1) @ 10,1
'OsmEx_DEactivates_Sln1SensD' Sln1_SensD(Osm!1,X~a),Osm(Localiz~e,r!1) ->
    Sln1_SensD(Osm!1,X~ia),Osm(Localiz~e,r!1) @ 10
'OsmIntern_Binds_Sln1SensDE' Osm(Localiz~i,r),Sln1_SensD(Osm,OsmIntern) <-> Osm(
    Localiz~i,r!1),Sln1_SensD(Osm!1,OsmIntern) @ 10,0.8
'OsmInt_Activates_Sln1SensDI' Osm(Localiz~i,r!1),Sln1_SensD(X~ia,Osm!1,OsmIntern
    ) -> Osm(Localiz~i,r!1),Sln1_SensD(X~a,Osm!1,OsmIntern) @ 10
'OsmEx_Binds_Hkr1' Hkr1(STR),Osm(r,Localiz~e) <-> Hkr1(STR!1),Osm(r!1,Localiz~e)
```

```
     @ 10,1
'OsmEx_Activates_HKR1' Osm(r!1,Localiz~e),Hkr1(STR!1,X~ia) -> Osm(r!1,Localiz~e)
    ,Hkr1(STR!1,X~a) @ 2
'OsmInt_Binds_Hkr1E' Osm(r,Localiz~i),Hkr1(OsmIntern,STR) <-> Osm(r!1,Localiz~i)
    ,Hkr1(OsmIntern,STR!1) @ 0.05,0.1
'OsmInt_BindsCoo_Hkr1E' Hkr1(OsmIntern,STR),Osm(r,Localiz~i) <-> Hkr1(OsmIntern
    !2,STR),Osm(r!2,Localiz~i) @ 0,0.1
'OsmInt_BindsCoo_Hkr1E Ref1' Osm(r!1,Localiz~i),Hkr1(OsmIntern,STR!1),Osm(r,
    Localiz~i) <-> Osm(r!1,Localiz~i),Hkr1(OsmIntern!2,STR!1),Osm(r!2,Localiz~i)
     @ 3,0.1
# 'OsmInt_Deactivates _HKR1E' Osm(r!1,Localiz~i),Hkr1(X~A,OsmIntern,STR!1) ->
    Osm(r!1,Localiz~i),Hkr1(X~IA,OsmIntern,STR!1) @ 5 # Inactive rule
'OsmIntCoo_Deactivates _HKR1E' Osm(r!1,Localiz~i),Hkr1(X~a,OsmIntern!2,STR!1),
    Osm(r!2,Localiz~i) -> Osm(r!1,Localiz~i),Hkr1(X~ia,OsmIntern!2,STR!1),Osm(r
    !2,Localiz~i) @ 10
'OsmEx_Binds_Msb2' Msb2(STR),Osm(r,Localiz~e) <-> Msb2(STR!1),Osm(r!1,Localiz~e)
     @ 10,1
'OsmEx_Activates_Msb2' Msb2(STR!1,X~ia),Osm(r!1,Localiz~e) -> Msb2(STR!1,X~a),
    Osm(r!1,Localiz~e) @ 2
'OsmInt_Binds_Msb2E' Msb2(STR,OsmIntern),Osm(r,Localiz~i) <-> Msb2(STR!1,
    OsmIntern),Osm(r!1,Localiz~i) @ 0.05,0.1
'OsmInt_BindsCoo_Msb2E' Msb2(STR,OsmIntern),Osm(r,Localiz~i) <-> Msb2(STR,
    OsmIntern!2),Osm(r!2,Localiz~i) @ 0,0.1
'OsmInt_BindsCoo_Msb2E Ref1' Msb2(STR!1,OsmIntern),Osm(r!1,Localiz~i),Osm(r,
    Localiz~i) <-> Msb2(STR!1,OsmIntern!2),Osm(r!1,Localiz~i),Osm(r!2,Localiz~i)
     @ 3,0.1
# 'OsmInt_Deactivates_Msb2E' Msb2(X~A,OsmIntern,STR!1),Osm(r!1,Localiz~i) ->
    Msb2(X~IA,OsmIntern,STR!1),Osm(r!1,Localiz~i) @ 5 # Inactive rule
'OsmIntCoo_Deactivates_Msb2E' Msb2(X~a,OsmIntern!2,STR!1),Osm(r!1,Localiz~i),Osm
    (r!2,Localiz~i) -> Msb2(X~ia,OsmIntern!2,STR!1),Osm(r!1,Localiz~i),Osm(r!2,
    Localiz~i) @ 10
'Fps1_Transp_Osm1' Fps1(Localiz~membr,Osm,A~a),Osm(Localiz~i,r) -> Fps1(Localiz~
    membr,Osm!1,A~a),Osm(Localiz~i,r!1) @ 100
# 'Fps1_Transp_Osm2' Fps1(Localiz~membr,Osm!1,A~a),Osm(Localiz~i,r!1) -> Fps1(
    Localiz~membr,Osm,A~a),Osm(Localiz~e,r) @ 10 # Inactive rule
'Fps1_Transp_Osm2 Ref1' Fps1(Localiz~memr,Osm!1,A~a),Osm(Localiz~i,r!1) -> Fps1(
    Localiz~membr,Osm,A~a) @ 10
'Sho1Act_Deactiv_Fps1_WOBinding' Fps1(Localiz~membr,A~a,Hog1),Sho1(X~a) -> Fps1(
    Localiz~membr,A~ia,Hog1),Sho1(X~a) @ 0.5
'Fps1_Activation' Fps1(Localiz~membr,Hog1,A~ia) -> Fps1(Localiz~membr,Hog1,A~a)
    @ 0
'Sln1Deact_Deactivates_Fps1' Fps1(Localiz~membr,Hog1,A~a),Sln1_SensD(X~ia) ->
```

```
    Fps1(Localiz~membr,Hog1,A~ia),Sln1_SensD(X~ia) @ 0.5
'Hog1_Binds_Sho1' Sho1(Localiz~membr,Hog1),Hog1(Sho1) <-> Sho1(Localiz~membr,
    Hog1!1),Hog1(Sho1!1) @ 0.0,1
'Hog1_Binds_Sho1 Ref1' Sho1(Localiz~membr,Hog1,X~a),Hog1(Tyr176~p,Localiz~cyto,
    Phosphat,Kinase,Sho1,Ptc23,Ste50,Thr174~p,TF,CD) <-> Sho1(Localiz~membr,Hog1
    !1,X~a),Hog1(Tyr176~p,Localiz~cyto,Phosphat,Kinase,Sho1!1,Ptc23,Ste50,Thr174
    ~p,TF,CD) @ 0.000005,1.0
'Hog1_Phosph_Sho1' Sho1(Localiz~membr,Hog1~u!1,X~a),Hog1(Tyr176~p,Localiz~cyto,
    Phosphat,Kinase,Sho1!1,Ptc23,Ste50,Thr174~p,TF,CD) -> Sho1(Localiz~membr,
    Hog1~p!1,X~a),Hog1(Tyr176~p,Localiz~cyto,Phosphat,Kinase,Sho1!1,Ptc23,Ste50,
    Thr174~p,TF,CD) @ 1
'Sho1_Dephosph' Sho1(Localiz~membr,Hog1~p) -> Sho1(Localiz~membr,Hog1~u) @
    0.0005
'OsmDel' Osm(Localiz~e),Osm(Localiz~i) -> @ 0.0

#######
# Chapter 'GeneralRules'
'Hog1pp_Binds_GlycFeedback' Hog1(Localiz~cyto,Kinase),GlycFeedback(Hog1,X) <->
    Hog1(Localiz~cyto,Kinase!1),GlycFeedback(Hog1!1,X) @ 0.0,0.1
'Hog1pp_Binds_GlycFeedback Ref2' GlycFeedback(Hog1,GlycProd,X~ia),Hog1(Localiz~
    cyto,Kinase,TF,Thr174,CD,Ste50,Tyr176,Phosphat,Ptc23,Sho1) <-> GlycFeedback(
    Hog1!1,GlycProd,X~ia),Hog1(Localiz~cyto,Kinase!1,TF,Thr174,CD,Ste50,Tyr176,
    Phosphat,Ptc23,Sho1) @ 0.005,0.1
'Hog1pp_Activ_GlycFeedb' GlycFeedback(Hog1!1,X~ia),Hog1(Thr174~p,Tyr176~p,
    Localiz~cyto,Kinase!1) -> GlycFeedback(Hog1!1,X~a),Hog1(Thr174~p,Tyr176~p,
    Localiz~cyto,Kinase!1) @ 1
# 'Hog1pp_Activ_GlycFeedb Ref1' GlycFeedback(Hog1!1,X~ia),Hog1(TF,Thr174~p,CD,
    Ste50,Tyr176~p,Localiz~cyto,Phosphat,Ptc23,Kinase!1,Sho1) -> GlycFeedback(
    Hog1!1,X~a),Hog1(TF,Thr174~p,CD,Ste50,Tyr176~p,Localiz~cyto,Phosphat,Ptc23,
    Kinase!1,Sho1) @ 1.0 # Inactive rule
'GlycFeedB_Binds_GlycProd' GlycFeedback(X,GlycProd),GlycProd(Hog1,X) <->
    GlycFeedback(X,GlycProd!1),GlycProd(Hog1!1,X) @ 0.0,0.1
# 'GlycFeedB_Binds_GlycProd Ref1' GlycProd(Hog1,X~ia),GlycFeedback(X~a,GlycProd,
    Hog1) <-> GlycProd(Hog1!1,X~ia),GlycFeedback(X~a,GlycProd!1,Hog1) @
    0.001,0.1 # Inactive rule
'GlycFeedB_Binds_GlycProd Ref2' GlycFeedback(X,GlycProd,Hog1),GlycProd(Hog1,X~ia
    ) <-> GlycFeedback(X,GlycProd!1,Hog1),GlycProd(Hog1!1,X~ia) @ 0.005,0.1
'GlycFeedB_Act_GlycProd' GlycFeedback(X~a,GlycProd!1),GlycProd(Hog1!1,X~ia) ->
    GlycFeedback(X~a,GlycProd!1),GlycProd(Hog1!1,X~a) @ 1
'GlycProd_Synt_OsmInt' GlycProd(Hog1,X~a) -> GlycProd(Hog1,X~a),Osm(r,Localiz~i)
     @ 0.08
'GlycProd_BasalSynt_OsmInt' GlycProd(Hog1,X~ia) -> GlycProd(Hog1,X~ia),Osm(r,
```

```
    Localiz~i) @ 0.0005
'GlycFeedB_Deact' GlycFeedback(X~a) -> GlycFeedback(X~ia) @ 0.01
'GlycProd_Deact' GlycProd(Hog1,X~a) -> GlycProd(Hog1,X~ia) @ 0.004
# 'GlycProd_Deact Ref1' GlycProd(Hog1,X~a),Sho1(A~IA) -> GlycProd(Hog1,X~ia),
    Sho1(A~IA) @ 1.0 # Inactive rule
'Hog1ppnuc_Binds_TFGen' TF(Hopg1pp),Hog1(Localiz~nuc,TF) <-> TF(Hopg1pp!1),Hog1(
    Localiz~nuc,TF!1) @ 0.0,0.01
'Hog1ppnuc_Binds_TFGen Ref2' Hog1(Localiz~nuc,TF,Ste50,Tyr176~p,Thr174~p),TF(
    Hopg1pp,Gene!_) <-> Hog1(Localiz~nuc,TF!1,Ste50,Tyr176~p,Thr174~p),TF(
    Hopg1pp!1,Gene!_) @ 0.00005,0.01
'Hog1ppnuc_Binds_TFGen Ref1' TF(Hopg1pp),Hog1(Localiz~nuc,TF,Tyr176~u) <-> TF(
    Hopg1pp!1),Hog1(Localiz~nuc,TF!1,Tyr176~u) @ 0.0,1.0
'Hog1ppnuc_Binds_TFGen Ref3' TF(Hopg1pp),Hog1(Localiz~nuc,TF,Thr174~u) <-> TF(
    Hopg1pp!1),Hog1(Localiz~nuc,TF!1,Thr174~u) @ 0.0,1.0
'Hog1ppnuc_Binds_TFSko1' Sko1(Hog1pp),Hog1(Localiz~nuc,TF) <-> Sko1(Hog1pp!1),
    Hog1(Localiz~nuc,TF!1) @ 0.0,0.1
'Hog1ppnuc_Binds_TFSko1 Ref1' Sko1(Hog1pp),Hog1(Localiz~nuc,TF,Ste50,Tyr176~p,
    Thr174~p) <-> Sko1(Hog1pp!1),Hog1(Localiz~nuc,TF!1,Ste50,Tyr176~p,Thr174~p)
    @ 0.00005,0.1
'Hog1ppnuc_Binds_TFSko1 Ref2' Sko1(Hog1pp),Hog1(Localiz~nuc,TF,Thr174~u) <->
    Sko1(Hog1pp!1),Hog1(Localiz~nuc,TF!1,Thr174~u) @ 0.0,1.0
'Hog1ppnuc_Binds_TFSko1 Ref3' Sko1(Hog1pp),Hog1(Localiz~nuc,TF,Tyr176~u) <->
    Sko1(Hog1pp!1),Hog1(Localiz~nuc,TF!1,Tyr176~u) @ 0.0,1.0
'TF_Binds_GeneVenus' Venus(TF),TF(Gene) <-> Venus(TF!1),TF(Gene!1) @ 0.0,0.1
'TF_Binds_GeneVenus Ref1' TF(Hopg1pp,Gene),Venus(TF,PolII) <-> TF(Hopg1pp,Gene
    !1),Venus(TF!1,PolII) @ 0.001,0.1
# 'TF_Binds_GeneVenus Ref2' TF(Hopg1pp!1,Gene),Hog1(TF!1,Thr174~p,Tyr176~p,
    Localiz~nuc),Venus(TF,PolII) <-> TF(Hopg1pp!2,Gene!1),Hog1(TF!2,Thr174~p,
    Tyr176~p,Localiz~nuc),Venus(TF!1,PolII) @ 0.0002,0.01 # Inactive rule
'TF_Binds_GeneCherry' TF(Gene),mCherry(TF) <-> TF(Gene!1),mCherry(TF!1) @
    0.0,0.1
'TF_Binds_GeneCherry Ref1' TF(Hopg1pp,Gene),mCherry(TF,PolII) <-> TF(Hopg1pp,
    Gene!1),mCherry(TF!1,PolII) @ 0.001,0.1
# 'TF_Binds_GeneCherry Ref2' TF(Hopg1pp!1,Gene),Hog1(TF!1,Tyr176~p,Thr174~p,
    Localiz~nuc),mCherry(TF,PolII) <-> TF(Hopg1pp!2,Gene!1),Hog1(TF!2,Tyr176~p,
    Thr174~p,Localiz~nuc),mCherry(TF!1,PolII) @ 0.0002,0.01 # Inactive rule
'TF_Binds_Gene' gene(TF),TF(Gene) <-> gene(TF!1),TF(Gene!1) @ 0.0,0.1
'TF_Binds_Gene Ref1' gene(TF,PolII),TF(Hopg1pp,Gene) <-> gene(TF!1,PolII),TF(
    Hopg1pp,Gene!1) @ 0.001,0.1
# 'TF_Binds_Gene Ref2' gene(TF,PolII),TF(Hopg1pp!1,Gene),Hog1(TF!1,Tyr176~p,
    Thr174~p,Localiz~nuc) <-> gene(TF!1,PolII),TF(Hopg1pp!2,Gene!1),Hog1(TF!2,
    Tyr176~p,Thr174~p,Localiz~nuc) @ 0.0002,0.01 # Inactive rule
```

```
'PolII_Binds_VenTFHog' Venus(PolII),PolII(Gene) <-> Venus(PolII!2),PolII(Gene!2)
     @ 0.0,0.005
'PolII_Binds_InitComplex Ref1' Hog1(Thr174~p,Tyr176~p,Localiz~nuc,TF!1),TF(
    Hopg1pp!1,Gene!2),Venus(TF!2,PolII),PolII(Remodel,Gene) <-> Hog1(Thr174~p,
    Tyr176~p,Localiz~nuc,TF!1),TF(Hopg1pp!1,Gene!3),Venus(TF!3,PolII!2),PolII(
    Remodel,Gene!2) @ 0.005,0.01
'PolII_Binds_InitComplex Ref2' PolII(Gene),TF(Hopg1pp,Gene!1),Venus(PolII,TF!1)
    <-> PolII(Gene!2),TF(Hopg1pp,Gene!1),Venus(PolII!2,TF!1) @ 0.0,1
'PolII_Binds_InitComplex Ref3' PolII(Gene),Venus(PolII,TF) <-> PolII(Gene!2),
    Venus(PolII!2,TF) @ 0.0,1
'PolII_Binds_CherTFHog' PolII(Gene),mCherry(PolII) <-> PolII(Gene!2),mCherry(
    PolII!2) @ 0.0,0.005
'PolII_Binds_InitComplexCherry Ref1' Hog1(Thr174~p,Tyr176~p,Localiz~nuc,TF!1),TF
    (Hopg1pp!1,Gene!2),PolII(Remodel,Gene),mCherry(PolII,TF!2) <-> Hog1(Thr174~p
    ,Tyr176~p,Localiz~nuc,TF!1),TF(Hopg1pp!1,Gene!3),PolII(Remodel,Gene!2),
    mCherry(PolII!2,TF!3) @ 0.005,0.01
'PolII_Binds_InitComplexCherry Ref2' PolII(Gene),TF(Hopg1pp,Gene!1),mCherry(TF
    !1,PolII) <-> PolII(Gene!2),TF(Hopg1pp,Gene!1),mCherry(TF!1,PolII!2) @
    0.0,1.0
'PolII_Binds_InitComplexCherry Ref3' PolII(Gene),mCherry(TF,PolII) <-> PolII(
    Gene!2),mCherry(TF,PolII!2) @ 0.0,1.0
'PolII_Binds_GeneTFHog' gene(PolII),PolII(Gene) <-> gene(PolII!2),PolII(Gene!2)
    @ 0.0,0.005
'PolII_Binds_InitComplexGene Ref1' Hog1(Thr174~p,Tyr176~p,Localiz~nuc,TF!1),TF(
    Hopg1pp!1,Gene!2),gene(TF!2,PolII),PolII(Remodel,Gene) <-> Hog1(Thr174~p,
    Tyr176~p,Localiz~nuc,TF!1),TF(Hopg1pp!1,Gene!3),gene(TF!3,PolII!2),PolII(
    Remodel,Gene!2) @ 0.005,0.02
'PolII_Binds_InitComplexGene Ref2' PolII(Gene),TF(Hopg1pp,Gene!1),gene(PolII,TF
    !1) <-> PolII(Gene!2),TF(Hopg1pp,Gene!1),gene(PolII!2,TF!1) @ 0.0,1.0
'PolII_Binds_InitComplexGene Ref3' PolII(Gene),gene(PolII,TF) <-> PolII(Gene!2),
    gene(PolII!2,TF) @ 0.0,1.0
'Remod_Binds_InitComplVen' Venus(Remodel),Remodeler(PolII) <-> Venus(Remodel!1),
    Remodeler(PolII!1) @ 0.0,0.0001
'Remod_Binds_InitCompl_2 Ref1' Hog1(Thr174~p,Tyr176~p,Localiz~nuc,TF!1),TF(
    Hopg1pp!1,Gene!2),Venus(TF!2,PolII!3,Remodel),PolII(Remodel,Gene!3),
    Remodeler(PolII) <-> Hog1(Thr174~p,Tyr176~p,Localiz~nuc,TF!1),TF(Hopg1pp!1,
    Gene!2),Venus(TF!2,PolII!3,Remodel!4),PolII(Remodel,Gene!3),Remodeler(PolII
    !4) @ 0.001,0.0001
'Remod_Binds_InitComplCherry' Remodeler(PolII),mCherry(Remodel) <-> Remodeler(
    PolII!1),mCherry(Remodel!1) @ 0.0,0.0001
'Remod_Binds_InitComplCherry_2 Ref1' Hog1(Thr174~p,Tyr176~p,Localiz~nuc,TF!1),TF
    (Hopg1pp!1,Gene!2),PolII(Remodel,Gene!3),Remodeler(PolII),mCherry(TF!2,PolII
```

```
    !3,Remodel) <-> Hog1(Thr174~p,Tyr176~p,Localiz~nuc,TF!1),TF(Hopg1pp!1,Gene
    !2),PolII(Remodel,Gene!3),Remodeler(PolII!4),mCherry(TF!2,PolII!3,Remodel!4)
     @ 0.001,0.0001
'Remod_Binds_InitComplGene' Remodeler(PolII),gene(Remodel) <-> Remodeler(PolII
    !4),gene(Remodel!4) @ 0.0,0.0001
'Remod_Binds_InitComplGene2 Ref1' TF(Hopg1pp!1,Gene!2),Hog1(Thr174~p,Tyr176~p,
    Localiz~nuc,TF!1),PolII(Remodel,Gene!3),Remodeler(PolII),gene(TF!2,PolII!3,
    Remodel) <-> TF(Hopg1pp!1,Gene!2),Hog1(Thr174~p,Tyr176~p,Localiz~nuc,TF!1),
    PolII(Remodel,Gene!3),Remodeler(PolII!4),gene(TF!2,PolII!3,Remodel!4) @
    0.001,0.0001
'TrscrptCmplx_cr_VenusmRNA2' Hog1(Thr174~p,Tyr176~p,Localiz~nuc,TF!1),TF(Hopg1pp
    !1,Gene!2),Venus(TF!2,PolII!3,Remodel!4),PolII(Remodel,Gene!3),Remodeler(
    PolII!4) -> Hog1(Thr174~p,Tyr176~p,Localiz~nuc,TF!1),TF(Hopg1pp!1,Gene!2),
    Venus(TF!2,PolII!3,Remodel!4),PolII(Remodel,Gene!3),Remodeler(PolII!4),
    mRNA_Venus(Localiz~nuc) @ 0.5
'TrscrptCmplx_cr_GeneRNA2' Hog1(Thr174~p,Tyr176~p,Localiz~nuc,TF!1),TF(Hopg1pp
    !1,Gene!2),PolII(Remodel,Gene!3),Remodeler(PolII!4),gene(PolII!3,TF!2,
    Remodel!4) -> Hog1(Thr174~p,Tyr176~p,Localiz~nuc,TF!1),TF(Hopg1pp!1,Gene!2),
    PolII(Remodel,Gene!3),Remodeler(PolII!4),gene(PolII!3,TF!2,Remodel!4),
    mRNA_Gene(Localiz~nuc) @ 0.5
'TrscrptCmplx_cr_mCherryRNA2' Hog1(Thr174~p,Tyr176~p,Localiz~nuc,TF!1),TF(
    Hopg1pp!1,Gene!2),PolII(Remodel,Gene!3),Remodeler(PolII!4),mCherry(PolII!3,
    TF!2,Remodel!4) -> Hog1(Thr174~p,Tyr176~p,Localiz~nuc,TF!1),TF(Hopg1pp!1,
    Gene!2),PolII(Remodel,Gene!3),Remodeler(PolII!4),mCherry(PolII!3,TF!2,
    Remodel!4),mRNA_mCherry(Localiz~nuc) @ 0.5

#######
# Chapter 'Sln1 Branch'
# 'SensorSln1_Assembly' Sln1_RecD(h),Sln1_SensD(K),Sln1_HkD(s,r) -> Sln1_RecD(h
    !2),Sln1_SensD(K!1),Sln1_HkD(s!1,r!2) @ 1.0 # Inactive rule
# 'Sln1SensD_Act_Inact' Sln1_SensD(X~a) -> Sln1_SensD(X~ia) @ 0.0 # Inactive
    rule
# 'Sln1_Dimerization' Sln1_HkD(DimR),Sln1_HkD(DimR) <-> Sln1_HkD(DimR!1),
    Sln1_HkD(DimR!1) @ 0.00016,0.1 # Inactive rule
'Sln1_Dimerization Ref1' Sln1_HkD(DimR,s!1),Sln1_SensD(K!1,X~ia),Sln1_HkD(DimR,s
    !2),Sln1_SensD(K!2,X~ia) <-> Sln1_HkD(DimR!1,s!2),Sln1_SensD(K!2,X~ia),
    Sln1_HkD(DimR!1,s!3),Sln1_SensD(K!3,X~ia) @ 0.00016,0.1
'Sln1_Dimerization Ref2' Sln1_HkD(DimR,s!1),Sln1_SensD(K!1,X~a),Sln1_HkD(DimR,s
    !2),Sln1_SensD(K!2,X~a) <-> Sln1_HkD(DimR!1,s!2),Sln1_SensD(K!2,X~a),
    Sln1_HkD(DimR!1,s!3),Sln1_SensD(K!3,X~a) @ 0.00016,0.1
'Sln1SensAct_AutPhosph_Sln1His' Sln1_SensD(K!1,X~a),Sln1_HkD(His576~u,s!1) ->
    Sln1_SensD(K!1,X~a),Sln1_HkD(His576~p,s!1) @ 1
```

```
'Sln1_CrossPh' Sln1_HkD(s!4,r!2,DimR!5),Sln1_RecD(h!2,Asp1144~u),Sln1_RecD(h!1),
    Sln1_HkD(r!1,DimR!5,His576~p,s!3),Sln1_SensD(K!3),Sln1_SensD(K!4) ->
    Sln1_HkD(s!4,r!1,DimR!5),Sln1_RecD(h!1,Asp1144~p),Sln1_RecD(h!3),Sln1_HkD(r
    !3,DimR!5,His576~u,s!2),Sln1_SensD(K!2),Sln1_SensD(K!4) @ 40
# 'Sln1HkD_Phosph_Sln1RecD' Sln1_HkD(His576~p,r!1),Sln1_RecD(Asp1144~u,h!1) ->
    Sln1_HkD(His576~u,r!1),Sln1_RecD(Asp1144~p,h!1) @ 50 # Inactive rule
'Ypd1_Binds_Sln1RecD' Sln1_RecD(Ypd1),Ypd1(Sln1,Localiz~membr) <-> Sln1_RecD(
    Ypd1!1),Ypd1(Sln1!1,Localiz~membr) @ 0.0,0.0001
'Ypd1_Binds_Sln1RecD Ref1' Sln1_RecD(Ypd1),Ypd1(Sln1,Localiz~membr,His64~u) <->
    Sln1_RecD(Ypd1!1),Ypd1(Sln1!1,Localiz~membr,His64~u) @ 0.0005,0.0001
# 'Ypd1_Binds_Sln1RecD Ref2' Sln1_RecD(Ypd1),Ypd1(Sln1,Localiz~membr,Ssk1,His64~
    u) <-> Sln1_RecD(Ypd1!1),Ypd1(Sln1!1,Localiz~membr,Ssk1,His64~u) @
    0.0005,0.0001 # Inactive rule
# 'Ypd1_Binds_Sln1RecD Ref3' Ypd1(Sln1,Localiz~membr,His64~u),Sln1_RecD(Ypd1,
    Asp1144~p) <-> Ypd1(Sln1!1,Localiz~membr,His64~u),Sln1_RecD(Ypd1!1,Asp1144~p
    ) @ 0.0005,0.0001 # Inactive rule
# 'Sln1RecD_Unbind_Ypd1P' Sln1_RecD(Ypd1!1),Ypd1(Sln1!1,His64~p,Localiz~membr)
    -> Sln1_RecD(Ypd1),Ypd1(Sln1,His64~p,Localiz~membr) @ 0.0001 # Inactive rule
# 'Sln1RecD_Unbind_Ypd1' Sln1_RecD(Ypd1!1),Ypd1(Sln1!1,Localiz~membr) ->
    Sln1_RecD(Ypd1),Ypd1(Sln1,Localiz~membr) @ 0.0001 # Inactive rule
'Sln1RecD_P_YPD1' Ypd1(Sln1!1,Localiz~membr,His64~u),Sln1_RecD(Asp1144~p,Ypd1!1)
     -> Ypd1(Sln1!1,Localiz~membr,His64~p),Sln1_RecD(Asp1144~u,Ypd1!1) @ 50
'Sln1RecD_CrossPh_Ypd1' Sln1_HkD(r!2,DimR!5),Sln1_HkD(r!1,DimR!5),Sln1_RecD(h!1,
    Ypd1!3),Sln1_RecD(h!2,Asp1144~p),Ypd1(Localiz~membr,His64~u,Sln1!3) ->
    Sln1_HkD(r!1,DimR!5),Sln1_HkD(r!3,DimR!5),Sln1_RecD(h!3,Ypd1!2),Sln1_RecD(h
    !1,Asp1144~u),Ypd1(Localiz~membr,His64~p,Sln1!2) @ 50
'Ssk1_Desphos' Ssk1(Asp554~p) -> Ssk1(Asp554~u) @ 0.008
'Ssk1Dimerpu_Dephosph' Ssk1(Dimer!1,Asp554~u),Ssk1(Dimer!1,Asp554~p) -> Ssk1(
    Dimer!1,Asp554~u),Ssk1(Dimer!1,Asp554~u) @ 20
# 'Ssk1Dimerpu_Dephosph Ref1' Ssk1(Dimer!1,Asp554~u,Ssk2_22,Ypd1),Ssk1(Dimer!1,
    Asp554~p,Ssk2_22,Ypd1) -> Ssk1(Dimer!1,Asp554~u,Ssk2_22,Ypd1),Ssk1(Dimer!1,
    Asp554~u,Ssk2_22,Ypd1) @ 0.0 # Inactive rule
# 'Ypd1_Binds_Ssk1Membr' Ypd1(Localiz~membr,Ssk1,His64),Ssk1(Ypd1,Localiz~membr)
     <-> Ypd1(Localiz~membr,Ssk1!1,His64),Ssk1(Ypd1!1,Localiz~membr) @
    0.00005,0.0 # Inactive rule
'Ypd1_Binds_Ssk1Membr Ref1' Ypd1(Localiz~membr,Ssk1,His64~p),Ssk1(Ypd1,Localiz~
    membr,Asp554~u) <-> Ypd1(Localiz~membr,Ssk1!1,His64~p),Ssk1(Ypd1!1,Localiz~
    membr,Asp554~u) @ 0.05,0.0
# 'Ypd1_Binds_Ssk1Membr Ref2' Sln1_RecD(Ypd1!1),Ssk1(Ypd1,Localiz~membr,Asp554),
    Ypd1(Localiz~membr,Ssk1,Sln1!1,His64) <-> Sln1_RecD(Ypd1!2),Ssk1(Ypd1!1,
    Localiz~membr,Asp554),Ypd1(Localiz~membr,Ssk1!1,Sln1!2,His64) @ 0.05,0.0 #
    Inactive rule
```

```
# 'Ypd1_Binds_Ssk1Membr Ref3' Sln1_RecD(Ypd1!1),Ypd1(Localiz~membr,Ssk1,His64~p,
    Sln1!1),Ssk1(Ypd1,Ssk2_22,Dimer,Localiz~membr,Asp554~p),Ssk1(Ypd1,Localiz~
    membr,Asp554~u,Dimer) <-> Sln1_RecD(Ypd1!2),Ypd1(Localiz~membr,Ssk1!1,His64~
    p,Sln1!2),Ssk1(Ypd1,Ssk2_22,Dimer,Localiz~membr,Asp554~p),Ssk1(Ypd1!1,
    Localiz~membr,Asp554~u,Dimer) @ 0.0005,0.0 # Inactive rule
'Ypd1_UnBind_Ssk1_membr' Ypd1(Localiz~membr,Ssk1!1),Ssk1(Asp554,Ypd1!1,Localiz~
    membr) -> Ypd1(Localiz~membr,Ssk1),Ssk1(Asp554,Ypd1,Localiz~membr) @ 0.01
# 'Ypd1_UnBind_Ssk1_membr Ref1' Ssk1(Asp554~p,Ypd1!1,Localiz~membr),Ypd1(Localiz
    ~membr,Ssk1!1,His64~u) -> Ssk1(Asp554~p,Ypd1,Localiz~membr),Ypd1(Localiz~
    membr,Ssk1,His64~u) @ 0.1 # Inactive rule
'Ypd1_UnBind_Ssk1_membr Ref2' Ypd1(Localiz~membr,Ssk1!1),Ssk1(Asp554~u,Ypd1!1,
    Localiz~membr) -> Ypd1(Localiz~membr,Ssk1),Ssk1(Asp554~u,Ypd1,Localiz~membr)
     @ 0.01
'Ypd1_UnBind_Ssk1_membr Ref3' Ypd1(Localiz~membr,Ssk1!1),Ssk1(Asp554~p,Ypd1!1,
    Localiz~membr) -> Ypd1(Localiz~membr,Ssk1),Ssk1(Asp554~p,Ypd1,Localiz~membr)
     @ 0.1
'Ypd1_Phos_Ssk1_membr' Ypd1(Localiz~membr,His64~p,Ssk1!1),Ssk1(Asp554~u,Ypd1!1,
    Localiz~membr) <-> Ypd1(Localiz~membr,His64~u,Ssk1!1),Ssk1(Asp554~p,Ypd1!1,
    Localiz~membr) @ 50,0
'Ypd1_Phos_Ssk1Dimer_membr' Ypd1(Localiz~membr,His64~p,Ssk1!1),Ssk1(Asp554~p,
    Ypd1!1,Localiz~membr,Dimer!2),Ssk1(Dimer!2,Localiz~membr,Asp554~u) -> Ypd1(
    Localiz~membr,His64~u,Ssk1!1),Ssk1(Asp554~p,Ypd1!1,Localiz~membr,Dimer!2),
    Ssk1(Dimer!2,Localiz~membr,Asp554~p) @ 30
# 'Ssk1_Dimerization' Ssk1(Dimer),Ssk1(Dimer) <-> Ssk1(Dimer!1),Ssk1(Dimer!1) @
    0.0001,0.001 # Inactive rule
'Ssk1_Dimerization Ref1' Ssk1(Dimer,Localiz~membr),Ssk1(Dimer,Localiz~membr) <->
     Ssk1(Dimer!1,Localiz~membr),Ssk1(Dimer!1,Localiz~membr) @ 0.005,0.01
'Ssk1_Dimerization Ref2' Ssk1(Dimer,Localiz~cyto),Ssk1(Dimer,Localiz~cyto) <->
    Ssk1(Dimer!1,Localiz~cyto),Ssk1(Dimer!1,Localiz~cyto) @ 0.0005,0.01
# 'Ssk1_Dimerization Ref3' Ssk1(Dimer,Ypd1,Ssk2_22,Localiz~cyto),Ssk1(Dimer,Ypd1
    ,Ssk2_22,Localiz~cyto) <-> Ssk1(Dimer!1,Ypd1,Ssk2_22,Localiz~cyto),Ssk1(
    Dimer!1,Ypd1,Ssk2_22,Localiz~cyto) @ 0.000005,0.01 # Inactive rule
# 'Ssk1_Dimerization Ref4' Ssk1(Dimer,Ypd1,Ssk2_22,Localiz~membr),Ssk1(Dimer,
    Ypd1,Ssk2_22,Localiz~membr) <-> Ssk1(Dimer!1,Ypd1,Ssk2_22,Localiz~membr),
    Ssk1(Dimer!1,Ypd1,Ssk2_22,Localiz~membr) @ 0.0005,0.01 # Inactive rule
'SSk1Mono_Binds_Ssk22_Cyto' Ssk1(Localiz~cyto,Ssk2_22),Ssk22(98_179,Localiz~cyto
    ) <-> Ssk1(Localiz~cyto,Ssk2_22!1),Ssk22(98_179!1,Localiz~cyto) @ 0.0,0.1
# 'Ssk1cyto_Binds_Ssk22_cyto' Ssk22(98_179,Localiz~cyto),Ssk1(Dimer!1,Localiz~
    cyto,Asp554),Ssk1(Ssk2_22,Dimer!1,Localiz~cyto,Asp554) <-> Ssk22(98_179!1,
    Localiz~cyto),Ssk1(Dimer!2,Localiz~cyto,Asp554),Ssk1(Ssk2_22!1,Dimer!2,
    Localiz~cyto,Asp554) @ 0.0,0.1 # Inactive rule
# 'Ssk1cyto_Binds_Ssk22_cyto Ref1' Ssk22(98_179,Localiz~cyto),Ssk1(Ssk2_22,Dimer
```

```
    !1,Localiz~cyto,Asp554~u),Ssk1(Dimer!1,Localiz~cyto,Asp554~u) <-> Ssk22(98
    _179!1,Localiz~cyto),Ssk1(Ssk2_22!1,Dimer!2,Localiz~cyto,Asp554~u),Ssk1(
    Dimer!2,Localiz~cyto,Asp554~u) @ 0.000005,0.1 # Inactive rule
'Ssk1cyto_Binds_Ssk22_cyto Ref3' Ssk1(Dimer!1,Localiz~cyto,Asp554~u),Ssk1(
    Ssk2_22,Dimer!1,Localiz~cyto,Asp554~u),Ssk22(98_179,Localiz~cyto,Thr1460~u)
    <-> Ssk1(Dimer!2,Localiz~cyto,Asp554~u),Ssk1(Ssk2_22!1,Dimer!2,Localiz~cyto,
    Asp554~u),Ssk22(98_179!1,Localiz~cyto,Thr1460~u) @ 0.000005,0.1
'Ssk1Mono_Binds_Ssk2_Cyto' Ssk1(Ssk2_22,Localiz~cyto),Ssk2(294_413,Localiz~cyto)
     <-> Ssk1(Ssk2_22!1,Localiz~cyto),Ssk2(294_413!1,Localiz~cyto) @ 0.0,0.1
# 'Ssk1_Binds_Ssk2_cyto' Ssk1(Dimer!1,Localiz~cyto,Asp554),Ssk1(Ssk2_22,Dimer!1,
    Localiz~cyto,Asp554),Ssk2(294_413,Localiz~cyto,Thr1460) <-> Ssk1(Dimer!1,
    Localiz~cyto,Asp554),Ssk1(Ssk2_22!2,Dimer!1,Localiz~cyto,Asp554),Ssk2(294
    _413!2,Localiz~cyto,Thr1460) @ 0.0,0.1 # Inactive rule
# 'Ssk1_Binds_Ssk2_cyto Ref1' Ssk1(Dimer!1,Localiz~cyto,Asp554~u),Ssk1(Ssk2_22,
    Dimer!1,Localiz~cyto,Asp554~u),Ssk2(294_413,Localiz~cyto,Thr1460) <-> Ssk1(
    Dimer!1,Localiz~cyto,Asp554~u),Ssk1(Ssk2_22!2,Dimer!1,Localiz~cyto,Asp554~u)
    ,Ssk2(294_413!2,Localiz~cyto,Thr1460) @ 1.0,1.0 # Inactive rule
'Ssk1_Binds_Ssk2_cyto Ref2' Ssk1(Dimer!1,Localiz~cyto,Asp554~u),Ssk1(Ssk2_22,
    Dimer!1,Localiz~cyto,Asp554~u),Ssk2(294_413,Localiz~cyto,Thr1460~u) <-> Ssk1
    (Dimer!1,Localiz~cyto,Asp554~u),Ssk1(Ssk2_22!2,Dimer!1,Localiz~cyto,Asp554~u
    ),Ssk2(294_413!2,Localiz~cyto,Thr1460~u) @ 0.000005,0.1
# 'Ypd1u_creates_Ssk1Dimuu' Ypd1(His64~u) -> Ypd1(His64~u),Ssk1(Ssk2_22,Localiz~
    cyto,Asp554~u,Dimer!1,Ypd1),Ssk1(Ypd1,Ssk2_22,Dimer!1,Localiz~cyto,Asp554~u)
     @ 0.00005 # Inactive rule

#######
# Chapter 'Sho1 Branch'
'Hkr1_Binds_Sho1' Hkr1(Sho1),Sho1(Mucin) <-> Hkr1(Sho1!1),Sho1(Mucin!1) @
    0.0,0.1
'Hkr1_Binds_Sho1 Ref1' Hkr1(Sho1,X~a),Sho1(Mucin) <-> Hkr1(Sho1!1,X~a),Sho1(
    Mucin!1) @ 0.0001,0.1
'Hkr1_Binds_Sho1 Ref2' Sho1(Mucin),Hkr1(Sho1,X~ia) <-> Sho1(Mucin!1),Hkr1(Sho1
    !1,X~ia) @ 0.0001,0.1
# 'Hkr1_Binds_Sho1 Ref3' Hkr1(Sho1,X~ia),Sho1(Mucin,X~a) <-> Hkr1(Sho1!1,X~ia),
    Sho1(Mucin!1,X~a) @ 0.01,0.001 # Inactive rule
# 'Hkr1_Activates_Sho1' Sho1(Mucin!1,X~ia),Hkr1(Sho1!1,X~a) -> Sho1(Mucin!1,X~a)
    ,Hkr1(Sho1!1,X~a) @ 1 # Inactive rule
'Hkr1_Activates_Sho1 Ref1' Hkr1(Sho1!1,X~a),Sho1(Mucin!1,Hog1~u,X~ia) -> Hkr1(
    Sho1!1,X~a),Sho1(Mucin!1,Hog1~u,X~a) @ 1.0
'Hkr1_DEActivates_Sho1' Sho1(Mucin!1,X~a),Hkr1(Sho1!1,X~ia) -> Sho1(Mucin!1,X~ia
    ),Hkr1(Sho1!1,X~ia) @ 10
# 'Hkr1_DEActivates_Sho1MUCIN2' Sho1(Mucin2!1,X~a),Hkr1(Sho1!1,X~ia) -> Sho1(
```

```
    Mucin2!1,X~ia),Hkr1(Sho1!1,X~ia) @ 10 # Inactive rule
# 'HKR1_Deactivation' Hkr1(X~a) -> Hkr1(X~ia) @ 0.00005 # Inactive rule
'Msb2_Binds_Sho1' Sho1(Mucin),Msb2(Sho1) <-> Sho1(Mucin!1),Msb2(Sho1!1) @
    0.0,0.1
'Msb2_Binds_Sho1 Ref1' Msb2(Sho1,X~a),Sho1(Mucin) <-> Msb2(Sho1!1,X~a),Sho1(
    Mucin!1) @ 0.0001,0.1
'Msb2_Binds_Sho1 Ref2' Sho1(Mucin),Msb2(Sho1,X~ia) <-> Sho1(Mucin!1),Msb2(Sho1
    !1,X~ia) @ 0.0001,0.1
# 'Msb2_Binds_Sho1 Ref3' Msb2(Sho1,X~ia),Sho1(Mucin,X~a) <-> Msb2(Sho1!1,X~ia),
    Sho1(Mucin!1,X~a) @ 0.001,0.001 # Inactive rule
# 'Msb2_Activates_Sho1' Sho1(Mucin!1,X~ia),Msb2(Sho1!1,X~a) -> Sho1(Mucin!1,X~a)
    ,Msb2(Sho1!1,X~a) @ 10 # Inactive rule
'Msb2_Activates_Sho1 Ref1' Msb2(Sho1!1,X~a),Sho1(Mucin!1,Hog1~u,X~ia) -> Msb2(
    Sho1!1,X~a),Sho1(Mucin!1,Hog1~u,X~a) @ 1.0
'Msb2_DEActivates_Sho1' Sho1(Mucin!1,X~a),Msb2(Sho1!1,X~ia) -> Sho1(Mucin!1,X~ia
    ),Msb2(Sho1!1,X~ia) @ 10
# 'Msb2_DEActivates_Sho1MUCIN2' Msb2(Sho1!1,X~ia),Sho1(Mucin2!1,X~a) -> Msb2(
    Sho1!1,X~ia),Sho1(Mucin2!1,X~ia) @ 10 # Inactive rule
# 'Msb2_Deactivation' Msb2(X~a) -> Msb2(X~ia) @ 0.00005 # Inactive rule
# 'Sho1_Deactivation' Sho1(X~a) -> Sho1(X~ia) @ 0.000001 # Inactive rule
# 'Cdc24_Activation' Cdc24(sho1,DH,Localiz,PhosphSites,X~ia,PB1,PH) -> Cdc24(
    sho1,DH,Localiz,PhosphSites,X~a,PB1,PH) @ 0.005 # Inactive rule
# 'Cdc24membr_Binds_Sho1' Cdc24(sho1,Localiz~membr),Sho1(cdc24,Localiz~membr,X~a
    ) <-> Cdc24(sho1!1,Localiz~membr),Sho1(cdc24!1,Localiz~membr,X~a) @
    0.00005,0.1 # Inactive rule
# 'Sho1_Activates_Cdc24' Sho1(cdc24!1,Localiz~membr,X~a),Cdc24(sho1!1,Localiz~
    membr,X~ia) -> Sho1(cdc24!1,Localiz~membr,X~a),Cdc24(sho1!1,Localiz~membr,X~
    a) @ 1 # Inactive rule
'Sho1_Activates_Cdc24Unbound' Sho1(cdc24,Localiz~membr,X~a),Cdc24(sho1,Localiz~
    membr,X~ia) -> Sho1(cdc24,Localiz~membr,X~a),Cdc24(sho1,Localiz~membr,X~a) @
     0.0001
'Cdc24_Deactivation' Cdc24(X~a) -> Cdc24(X~ia) @ 0.005
'Cla4membr_Binds_Ste11membr' Cla4(Localiz~membr,Ste11),Ste11(Cla4,Localiz~membr)
     <-> Cla4(Localiz~membr,Ste11!1),Ste11(Cla4!1,Localiz~membr) @ 0,0.1
# 'Cla4membr_Binds_Ste11membr Ref1' Cla4(X~a,Localiz~membr,Ste11),Ste11(Cla4,
    Localiz~membr) <-> Cla4(X~a,Localiz~membr,Ste11!1),Ste11(Cla4!1,Localiz~
    membr) @ 0.0000005,0.1 # Inactive rule
'Cla4membr_Binds_Ste11membr Ref2' Cla4(Localiz~membr,Ste11,X~a),Ste11(Cla4,
    Localiz~membr,Ste20) <-> Cla4(Localiz~membr,Ste11!1,X~a),Ste11(Cla4!1,
    Localiz~membr,Ste20) @ 0.0000005,0.1
'Cla4cyto_Binds_Ste11cyto' Cla4(Localiz~cyto,Ste11),Ste11(Cla4,Localiz~cyto) <->
     Cla4(Localiz~cyto,Ste11!1),Ste11(Cla4!1,Localiz~cyto) @ 0,0.1
```

```
# 'Cla4cyto_Binds_Ste11cyto Ref1' Cla4(X~a,Localiz~cyto,Ste11),Ste11(Cla4,
    Localiz~cyto) <-> Cla4(X~a,Localiz~cyto,Ste11!1),Ste11(Cla4!1,Localiz~cyto)
    @ 0.00000005,0.1 # Inactive rule
'Cla4cyto_Binds_Ste11cyto Ref2' Cla4(Localiz~cyto,Ste11,X~a),Ste11(Cla4,Localiz~
    cyto,Ste20) <-> Cla4(Localiz~cyto,Ste11!1,X~a),Ste11(Cla4!1,Localiz~cyto,
    Ste20) @ 0.00000005,0.1
'Cla4_PhosphDirect_Ste11u_Membr' Cla4(X~a,Ste11!1,Localiz~membr),Ste11(
    Ser281Ser285Thr286~u,Cla4!1,Localiz~membr) -> Cla4(X~a,Ste11!1,Localiz~membr
    ),Ste11(Ser281Ser285Thr286~p,Cla4!1,Localiz~membr) @ 0.1
'Cla4_PhosphDirect_Ste11u_Cyto' Cla4(X~a,Ste11!1,Localiz~cyto),Ste11(
    Ser281Ser285Thr286~u,Cla4!1,Localiz~cyto) -> Cla4(X~a,Ste11!1,Localiz~cyto),
    Ste11(Ser281Ser285Thr286~p,Cla4!1,Localiz~cyto) @ 0.1
'Cla4_PhosphDirect_Ste11p_Membr' Cla4(X~a,Ste11!1,Localiz~membr),Ste11(
    Ser281Ser285Thr286~p,Cla4!1,Localiz~membr) -> Cla4(X~a,Ste11!1,Localiz~membr
    ),Ste11(Ser281Ser285Thr286~pp,Cla4!1,Localiz~membr) @ 0.1
'Cla4_PhosphDirect_Ste11p_Cyto' Cla4(X~a,Ste11!1,Localiz~cyto),Ste11(
    Ser281Ser285Thr286~p,Cla4!1,Localiz~cyto) -> Cla4(X~a,Ste11!1,Localiz~cyto),
    Ste11(Ser281Ser285Thr286~pp,Cla4!1,Localiz~cyto) @ 0.1
'Cla4_PhosphDirect_Ste11pp_Membr' Cla4(X~a,Ste11!1,Localiz~membr),Ste11(
    Ser281Ser285Thr286~pp,Cla4!1,Localiz~membr) -> Cla4(X~a,Ste11!1,Localiz~
    membr),Ste11(Ser281Ser285Thr286~ppp,Cla4!1,Localiz~membr) @ 0.1
'Cla4_PhosphDirect_Ste11pp_Cyto' Cla4(X~a,Ste11!1,Localiz~cyto),Ste11(
    Ser281Ser285Thr286~pp,Cla4!1,Localiz~cyto) -> Cla4(X~a,Ste11!1,Localiz~cyto)
    ,Ste11(Ser281Ser285Thr286~ppp,Cla4!1,Localiz~cyto) @ 0.1
# 'Cla4_Deactivation_cyto' Cla4(X~a,Localiz~cyto) -> Cla4(X~ia,Localiz~cyto) @
    0.008 # Inactive rule
'Cla4_Deactivation_cyto Ref1' Cla4(X~a,Localiz~cyto,CRIB) -> Cla4(X~ia,Localiz~
    cyto,CRIB) @ 0.008
# 'Cla4_Deactivation_membr' Cla4(X~a,Localiz~membr) -> Cla4(X~ia,Localiz~membr)
    @ 0.008 # Inactive rule
'Cla4_Deactivation_membr Ref1' Cla4(X~a,Localiz~membr,CRIB) -> Cla4(X~ia,Localiz
    ~membr,CRIB) @ 0.008
'Ste20membr_Binds_Ste11membr' Ste20(Localiz~membr,Ste11),Ste11(Localiz~membr,
    Ste20) <-> Ste20(Localiz~membr,Ste11!1),Ste11(Localiz~membr,Ste20!1) @
    0.0,0.1
# 'Ste20membr_Binds_Ste11membr Ref1' Ste20(Localiz~membr,Ste11,X~a),Ste11(
    Localiz~membr,Ste20) <-> Ste20(Localiz~membr,Ste11!1,X~a),Ste11(Localiz~
    membr,Ste20!1) @ 0.000005,0.1 # Inactive rule
'Ste20membr_Binds_Ste11membr Ref2' Ste20(Localiz~membr,Ste11,X~a),Ste11(Localiz~
    membr,Ste20,Cla4) <-> Ste20(Localiz~membr,Ste11!1,X~a),Ste11(Localiz~membr,
    Ste20!1,Cla4) @ 0.000005,0.1
'Ste20cyto_Binds_Ste11cyto' Ste20(Localiz~cyto,Ste11),Ste11(Localiz~cyto,Ste20)
```

```
    <-> Ste20(Localiz~cyto,Ste11!1),Ste11(Localiz~cyto,Ste20!1) @ 0.0,0.1
# 'Ste20cyto_Binds_Ste11cyto Ref1' Ste20(Localiz~cyto,Ste11,X~a),Ste11(Localiz~
    cyto,Ste20) <-> Ste20(Localiz~cyto,Ste11!1,X~a),Ste11(Localiz~cyto,Ste20!1)
    @ 0.0000005,0.1 # Inactive rule
'Ste20cyto_Binds_Ste11cyto Ref2' Ste20(Localiz~cyto,Ste11,X~a),Ste11(Localiz~
    cyto,Ste20,Cla4) <-> Ste20(Localiz~cyto,Ste11!1,X~a),Ste11(Localiz~cyto,
    Ste20!1,Cla4) @ 0.0000005,0.1
'Ste20_PhosphDirect_Ste11u_Membr' Ste20(Localiz~membr,Ste11!1,X~a),Ste11(Localiz
    ~membr,Ste20!1,Ser281Ser285Thr286~u) -> Ste20(Localiz~membr,Ste11!1,X~a),
    Ste11(Localiz~membr,Ste20!1,Ser281Ser285Thr286~p) @ 0.1
'Ste20_PhosphDirect_Ste11u_Cyto' Ste20(Localiz~cyto,Ste11!1,X~a),Ste11(Localiz~
    cyto,Ste20!1,Ser281Ser285Thr286~u) -> Ste20(Localiz~cyto,Ste11!1,X~a),Ste11(
    Localiz~cyto,Ste20!1,Ser281Ser285Thr286~p) @ 0.1
'Ste20_PhosphDirect_Ste11p_Membr' Ste20(Localiz~membr,Ste11!1,X~a),Ste11(Localiz
    ~membr,Ste20!1,Ser281Ser285Thr286~p) -> Ste20(Localiz~membr,Ste11!1,X~a),
    Ste11(Localiz~membr,Ste20!1,Ser281Ser285Thr286~pp) @ 0.1
'Ste20_PhosphDirect_Ste11p_Cyto' Ste20(Localiz~cyto,Ste11!1,X~a),Ste11(Localiz~
    cyto,Ste20!1,Ser281Ser285Thr286~p) -> Ste20(Localiz~cyto,Ste11!1,X~a),Ste11(
    Localiz~cyto,Ste20!1,Ser281Ser285Thr286~pp) @ 0.1
'Ste20_PhosphDirect_Ste11pp_Membr' Ste20(Localiz~membr,Ste11!1,X~a),Ste11(
    Localiz~membr,Ste20!1,Ser281Ser285Thr286~pp) -> Ste20(Localiz~membr,Ste11!1,
    X~a),Ste11(Localiz~membr,Ste20!1,Ser281Ser285Thr286~ppp) @ 0.1
'Ste20_PhosphDirect_Ste11pp_Cyto' Ste20(Localiz~cyto,Ste11!1,X~a),Ste11(Localiz~
    cyto,Ste20!1,Ser281Ser285Thr286~pp) -> Ste20(Localiz~cyto,Ste11!1,X~a),Ste11
    (Localiz~cyto,Ste20!1,Ser281Ser285Thr286~ppp) @ 0.1
# 'Ste20_Deactivation_Membr' Ste20(Localiz~membr,X~a) -> Ste20(Localiz~membr,X~
    ia) @ 0.008 # Inactive rule
'Ste20_Deactivation_Membr Ref1' Ste20(Localiz~membr,CRIB,X~a) -> Ste20(Localiz~
    membr,CRIB,X~ia) @ 0.0008
# 'Ste20_Deactivation_Cyto' Ste20(Localiz~cyto,X~a) -> Ste20(Localiz~cyto,X~ia)
    @ 0.008 # Inactive rule
'Ste20_Deactivation_Cyto Ref1' Ste20(Localiz~cyto,CRIB,X~a) -> Ste20(Localiz~
    cyto,CRIB,X~ia) @ 0.0008
'Bem1membr_Binds_Cla4membr' Cla4(Localiz~membr,PRR),Bem1(Localiz~membr,SH3b) <->
     Cla4(Localiz~membr,PRR!1),Bem1(Localiz~membr,SH3b!1) @ 0.0005,0.1
'Bem1cyto_Binds_Cla4cyto' Bem1(Localiz~cyto,SH3b),Cla4(Localiz~cyto,PRR) <->
    Bem1(Localiz~cyto,SH3b!1),Cla4(Localiz~cyto,PRR!1) @ 0.00005,0.1
'Bem1membr_Binds_Ste20membr' Bem1(Localiz~membr,SH3b),Ste20(Localiz~membr,PRR)
    <-> Bem1(Localiz~membr,SH3b!1),Ste20(Localiz~membr,PRR!1) @ 0.0,0.01
'Bem1membr_Binds_Ste20membr Ref1' Bem1(Localiz~membr,SH3b),Ste20(Localiz~membr,
    PRR,X~ia) <-> Bem1(Localiz~membr,SH3b!1),Ste20(Localiz~membr,PRR!1,X~ia) @
    0.0005,0.01
```

```
'Bem1membr_Binds_Ste20membr Ref2' Bem1(Localiz~membr,SH3b),Ste20(Localiz~membr,
    PRR,X~a) <-> Bem1(Localiz~membr,SH3b!1),Ste20(Localiz~membr,PRR!1,X~a) @
    0.00005,0.1
'Bem1cyto_Binds_Ste20cyto' Bem1(Localiz~cyto,SH3b),Ste20(Localiz~cyto,PRR) <->
    Bem1(Localiz~cyto,SH3b!1),Ste20(Localiz~cyto,PRR!1) @ 0.00,0.01
'Bem1cyto_Binds_Ste20cyto Ref1' Bem1(Localiz~cyto,SH3b),Ste20(Localiz~cyto,PRR,X
    ~ia) <-> Bem1(Localiz~cyto,SH3b!1),Ste20(Localiz~cyto,PRR!1,X~ia) @
    0.00005,0.01
'Bem1cyto_Binds_Ste20cyto Ref2' Bem1(Localiz~cyto,SH3b),Ste20(Localiz~cyto,PRR)
    <-> Bem1(Localiz~cyto,SH3b!1),Ste20(Localiz~cyto,PRR!1) @ 0.000005,0.1
'Bem1membr_Binds_Cdc24membr' Bem1(Localiz~membr,PB1),Cdc24(Localiz~membr,PB1)
    <-> Bem1(Localiz~membr,PB1!1),Cdc24(Localiz~membr,PB1!1) @ 0.0,0.1
'Bem1membr_Binds_Cdc24membr Ref1' Bem1(Localiz~membr,PB1),Cdc24(Localiz~membr,
    PB1,PhosphSites~u) <-> Bem1(Localiz~membr,PB1!1),Cdc24(Localiz~membr,PB1!1,
    PhosphSites~u) @ 0.00005,1.0
'Bem1cyto_Binds_Cdc24cyto' Bem1(Localiz~cyto,PB1),Cdc24(Localiz~cyto,PB1) <->
    Bem1(Localiz~cyto,PB1!1),Cdc24(Localiz~cyto,PB1!1) @ 0.0,0.1
'Bem1cyto_Binds_Cdc24cyto Ref1' Bem1(Localiz~cyto,PB1),Cdc24(Localiz~cyto,PB1,
    PhosphSites~u) <-> Bem1(Localiz~cyto,PB1!1),Cdc24(Localiz~cyto,PB1!1,
    PhosphSites~u) @ 0.000005,0.1
'Cla4_Phosph_Cdc24_membr' Cla4(X~a,Localiz~membr,PRR!1),Bem1(PB1!2,Localiz~membr
    ,SH3b!1),Cdc24(Localiz~membr,PB1!2,PhosphSites~u) -> Cla4(X~a,Localiz~membr,
    PRR!1),Bem1(PB1!2,Localiz~membr,SH3b!1),Cdc24(Localiz~membr,PB1!2,
    PhosphSites~p) @ 1
'Cla4_Phosph_Cdc24_cyto' Bem1(Localiz~cyto,SH3b!1,PB1!2),Cla4(X~a,Localiz~cyto,
    PRR!1),Cdc24(Localiz~cyto,PB1!2,PhosphSites~u) -> Bem1(Localiz~cyto,SH3b!1,
    PB1!2),Cla4(X~a,Localiz~cyto,PRR!1),Cdc24(Localiz~cyto,PB1!2,PhosphSites~p)
    @ 1
'Cdc24Phomembr_Diss_Bem1membr' Bem1(Localiz~membr,PB1!1),Cdc24(Localiz~membr,PB1
    !1,PhosphSites~p) -> Bem1(Localiz~membr,PB1),Cdc24(Localiz~membr,PB1,
    PhosphSites~p) @ 10
'Cdc24Phocyto_Diss_Bem1cyto' Bem1(Localiz~cyto,PB1!1),Cdc24(Localiz~cyto,PB1!1,
    PhosphSites~p) -> Bem1(Localiz~cyto,PB1),Cdc24(Localiz~cyto,PB1,PhosphSites~
    p) @ 10
# 'Bem1membr_Activ_Cdc24membr' Bem1(Localiz~membr,PB1!1),Cdc24(Localiz~membr,X~
    ia,PB1!1) -> Bem1(Localiz~membr,PB1!1),Cdc24(Localiz~membr,X~a,PB1!1) @
    0.001 # Inactive rule
# 'Bem1cyto_Activ_Cdc24cyto' Bem1(Localiz~cyto,PB1!1),Cdc24(Localiz~cyto,X~ia,
    PB1!1) -> Bem1(Localiz~cyto,PB1!1),Cdc24(Localiz~cyto,X~a,PB1!1) @ 0.001 #
    Inactive rule
'Ste11ppp_U_Membr' Ste11(Ser281Ser285Thr286~ppp,Localiz~membr) -> Ste11(
    Ser281Ser285Thr286~pp,Localiz~membr) @ 0.001
```

```
'Ste11ppp_U_Cyto' Ste11(Ser281Ser285Thr286~ppp,Localiz~cyto) -> Ste11(
    Ser281Ser285Thr286~pp,Localiz~cyto) @ 0.001
'Ste11pp_U_Membr' Ste11(Ser281Ser285Thr286~pp,Localiz~membr) -> Ste11(
    Ser281Ser285Thr286~p,Localiz~membr) @ 0.0005
'Ste11pp_U_Cyto' Ste11(Ser281Ser285Thr286~pp,Localiz~Cyto) -> Ste11(
    Ser281Ser285Thr286~p,Localiz~Cyto) @ 0.0005
'Ste11p_U_Membr' Ste11(Ser281Ser285Thr286~p,Localiz~membr) -> Ste11(
    Ser281Ser285Thr286~u,Localiz~membr) @ 0.0005
'Ste11p_U_Cyto' Ste11(Ser281Ser285Thr286~p,Localiz~cyto) -> Ste11(
    Ser281Ser285Thr286~u,Localiz~cyto) @ 0.0005
'Cdc24pmembr_u' Cdc24(Localiz~membr,PhosphSites~p) -> Cdc24(Localiz~membr,
    PhosphSites~u) @ 0.0005
'Cdc24pcyto_u' Cdc24(Localiz~cyto,PhosphSites~p) -> Cdc24(Localiz~cyto,
    PhosphSites~u) @ 0.0005
# 'Cdc24_Oligo_DH-DH' Cdc24(DH),Cdc24(DH) <-> Cdc24(DH!1),Cdc24(DH!1) @ 0.0,0.0
    # Inactive rule

#######
# Chapter 'Cdc42 Interaction'
'Msb2_Binds_Cdc42' Msb2(Cyt_Cdc42,Localiz~membr),Cdc42(Msb2,Localiz~membr) <->
    Msb2(Cyt_Cdc42!1,Localiz~membr),Cdc42(Msb2!1,Localiz~membr) @ 0.0000,0.1
'Msb2_Binds_Cdc42 Ref1' Msb2(Cyt_Cdc42,Localiz~membr,X~A),Cdc42(Msb2,Localiz~
    membr) <-> Msb2(Cyt_Cdc42!1,Localiz~membr,X~A),Cdc42(Msb2!1,Localiz~membr) @
     0.00005,0.1
'Cdc24membr_Binds_Cdc42' Cdc42(Localiz~membr,GEF),Cdc24(Localiz~membr,DH) <->
    Cdc42(Localiz~membr,GEF!1),Cdc24(Localiz~membr,DH!1) @ 0.0,0.1
'Cdc24membr_Binds_Cdc42 Ref1' Cdc42(Localiz~membr,GEF,X~GDP),Cdc24(X~a,Localiz~
    membr,DH) <-> Cdc42(Localiz~membr,GEF!1,X~GDP),Cdc24(X~a,Localiz~membr,DH!1)
     @ 0.000005,0.1
'Cdc24membr_Binds_Cdc42 Ref2' Cdc42(Localiz~membr,GEF,X~GDP),Bem1(PB1!1),Cdc24(X
    ~a,Localiz~membr,PB1!1,DH) <-> Cdc42(Localiz~membr,GEF!1,X~GDP),Bem1(PB1!2),
    Cdc24(X~a,Localiz~membr,PB1!2,DH!1) @ 0.0005,0.1
'Cdc24membr_Binds_Cdc42 Ref3' Cdc42(Localiz~membr,GEF,X~GTP),Cdc24(Localiz~membr
    ,DH) <-> Cdc42(Localiz~membr,GEF!1,X~GTP),Cdc24(Localiz~membr,DH!1) @ 0.0,10
'Cdc24membr_Activates_Cdc42' Cdc42(GEF!1,Localiz~membr,X~GDP),Cdc24(X~a,Localiz~
    membr,DH!1) -> Cdc42(GEF!1,Localiz~membr,X~GTP),Cdc24(X~a,Localiz~membr,DH
    !1) @ 10
'Bem1membr_Binds_Cdc42GTP' Cdc42(Bem1),Bem1(Localiz~membr,CI) <-> Cdc42(Bem1!1),
    Bem1(Localiz~membr,CI!1) @ 0.0,0.1
'Bem1membr_Binds_Cdc42GTP Ref1' Bem1(Localiz~membr,CI),Cdc42(Bem1,ToCRIB,X~GTP)
    <-> Bem1(Localiz~membr,CI!1),Cdc42(Bem1!1,ToCRIB,X~GTP) @ 0.000005,0.1
'Bem1_Scaffolds_Cdc24Cdc42' Bem1(Localiz~membr,CI,PB1!1),Cdc42(GEF!2,Bem1,X~GTP)
```

```
    ,Cdc24(PB1!1,DH!2) -> Bem1(Localiz~membr,CI!1,PB1!2),Cdc42(GEF,Bem1!1,X~GTP)
    ,Cdc24(PB1!2,DH) @ 5
'Cdc42_Activates_Ste20membrViaBem1_I' Cdc42(ToCRIB,Localiz~membr,Bem1!1,X~GTP),
    Bem1(Localiz~membr,CI!1,SH3b!2),Ste20(CRIB,Localiz~membr,PRR!2,X~ia) ->
    Cdc42(ToCRIB!3,Localiz~membr,Bem1!1,X~GTP),Bem1(Localiz~membr,CI!1,SH3b!2),
    Ste20(CRIB!3,Localiz~membr,PRR!2,X~ia) @ 50
'Cdc42_Activates_Ste20membrViaBem1_II' Bem1(Localiz~membr,SH3b!1,CI!2),Ste20(
    Localiz~membr,CRIB!3,PRR!1,X~ia),Cdc42(Localiz~membr,Bem1!2,ToCRIB!3,X~GTP)
    -> Bem1(Localiz~membr,SH3b!3,CI),Ste20(Localiz~membr,CRIB!2,PRR!3,X~a),Cdc42
    (Localiz~membr,Bem1,ToCRIB!2,X~GTP) @ 100
'Cdc42_Binds_Ste20membr' Cdc42(ToCRIB,Localiz~membr),Ste20(Localiz~membr,CRIB)
    <-> Cdc42(ToCRIB!1,Localiz~membr),Ste20(Localiz~membr,CRIB!1) @ 0.0,0.01
'Cdc42_Binds_Ste20membr Ref2' Cdc42(ToCRIB,Localiz~membr,X~GTP),Ste20(Localiz~
    membr,CRIB) <-> Cdc42(ToCRIB!1,Localiz~membr,X~GTP),Ste20(Localiz~membr,CRIB
    !1) @ 0.0005,0.01
# 'Cdc42_Binds_Ste20membr Ref1' Cdc42(ToCRIB,Localiz~membr,X~GTP),Ste20(Localiz~
    membr,CRIB,PRR) <-> Cdc42(ToCRIB!1,Localiz~membr,X~GTP),Ste20(Localiz~membr,
    CRIB!1,PRR) @ 0.00005,0.01 # Inactive rule
'Cdc42_Activates_Ste20membr' Cdc42(ToCRIB!1,Localiz~membr,X~GTP),Ste20(Localiz~
    membr,X~ia,CRIB!1) -> Cdc42(ToCRIB!1,Localiz~membr,X~GTP),Ste20(Localiz~
    membr,X~a,CRIB!1) @ 10
'Ste50membr_Binds_Cdc42' Ste50(Localiz~membr,RA),Cdc42(Localiz~membr,ste50) <->
    Ste50(Localiz~membr,RA!1),Cdc42(Localiz~membr,ste50!1) @ 0.0,0.1
'Ste50membr_Binds_Cdc42 Ref4' Ste50(Localiz~membr,RA),Cdc42(Localiz~membr,ste50)
     <-> Ste50(Localiz~membr,RA!1),Cdc42(Localiz~membr,ste50!1) @ 0.0005,0.1
'Ste50membr_Binds_Cdc42 Ref5' Ste50(Localiz~membr,RA),Cdc42(Localiz~membr,ste50,
    Msb2!1),Msb2(Cyt_Cdc42!1) <-> Ste50(Localiz~membr,RA!1),Cdc42(Localiz~membr,
    ste50!1,Msb2!2),Msb2(Cyt_Cdc42!2) @ 0.0005,1.0
'Ste20_Phosph_Ste11' Ste20(CRIB!1,X~a),Cdc42(ToCRIB!1,ste50!2,X~GTP),Ste50(SAM
    !3,RA!2),Ste11(SAM!3,Ser281Ser285Thr286~u) -> Ste20(CRIB!1,X~a),Cdc42(ToCRIB
    !1,ste50!2,X~GTP),Ste50(SAM!3,RA!2),Ste11(SAM!3,Ser281Ser285Thr286~p) @ 10
'Ste20_Phosph_Ste11p' Ste20(CRIB!1,X~a),Cdc42(ToCRIB!1,ste50!2,X~GTP),Ste50(SAM
    !3,RA!2),Ste11(SAM!3,Ser281Ser285Thr286~p) -> Ste20(CRIB!1,X~a),Cdc42(ToCRIB
    !1,ste50!2,X~GTP),Ste50(SAM!3,RA!2),Ste11(SAM!3,Ser281Ser285Thr286~pp) @ 10
'Ste20_Phosph_Ste11pp' Ste20(CRIB!1,X~a),Cdc42(ToCRIB!1,ste50!2,X~GTP),Ste50(SAM
    !3,RA!2),Ste11(SAM!3,Ser281Ser285Thr286~pp) -> Ste20(CRIB!1,X~a),Cdc42(
    ToCRIB!1,ste50!2,X~GTP),Ste50(SAM!3,RA!2),Ste11(SAM!3,Ser281Ser285Thr286~ppp
    ) @ 10
'Ste20_Phosph_Ste11/Oligo' Ste20(CRIB!1,X~a),Cdc42(ToCRIB!1,ste50!2,X~GTP),Ste50
    (SAMoligo!3,RA!2),Ste11(SAM!3,Ser281Ser285Thr286~u) -> Ste20(CRIB!1,X~a),
    Cdc42(ToCRIB!1,ste50!2,X~GTP),Ste50(SAMoligo!3,RA!2),Ste11(SAM!3,
    Ser281Ser285Thr286~p) @ 10
```

```
'Ste20_Phosph_Ste11p/Oligo' Ste20(CRIB!1,X~a),Cdc42(ToCRIB!1,ste50!2,X~GTP),
    Ste50(SAMoligo!3,RA!2),Ste11(SAM!3,Ser281Ser285Thr286~p) -> Ste20(CRIB!1,X~a
    ),Cdc42(ToCRIB!1,ste50!2,X~GTP),Ste50(SAMoligo!3,RA!2),Ste11(SAM!3,
    Ser281Ser285Thr286~pp) @ 10
'Ste20_Phosph_Ste11pp/Oligo' Ste20(CRIB!1,X~a),Cdc42(ToCRIB!1,ste50!2,X~GTP),
    Ste50(SAMoligo!3,RA!2),Ste11(SAM!3,Ser281Ser285Thr286~pp) -> Ste20(CRIB!1,X~
    a),Cdc42(ToCRIB!1,ste50!2,X~GTP),Ste50(SAMoligo!3,RA!2),Ste11(SAM!3,
    Ser281Ser285Thr286~ppp) @ 10
# 'Ste20_Phosph_Ste11ViaBem1' Ste20(PRR!1,X~a),Ste50(SAM!3,RA!2),Cdc42(ste50!2,
    Bem1!4,X~GTP),Ste11(SAM!3,Ser281Ser285Thr286~u),Bem1(Localiz~membr,CI!4,SH3b
    !1) -> Ste20(PRR!4,X~a),Ste50(SAM!3,RA!2),Cdc42(ste50!2,Bem1!1,X~GTP),Ste11(
    SAM!3,Ser281Ser285Thr286~p),Bem1(Localiz~membr,CI!1,SH3b!4) @ 0 # Inactive
    rule
'Cdc42_Binds_Cla4membr' Cdc42(ToCRIB,Localiz~membr),Cla4(CRIB,Localiz~membr) <->
     Cdc42(ToCRIB!1,Localiz~membr),Cla4(CRIB!1,Localiz~membr) @ 0.0005,0.0
'Cdc42_Binds_Cla4membr Ref1' Cla4(CRIB,Localiz~membr),Cdc42(ToCRIB,Localiz~membr
    ,X~GTP) <-> Cla4(CRIB!1,Localiz~membr),Cdc42(ToCRIB!1,Localiz~membr,X~GTP) @
     0.0005,0.01
'Cdc42_Activates_Cla4membr' Cdc42(ToCRIB!1,Localiz~membr,X~GTP),Cla4(CRIB!1,X~ia
    ,Localiz~membr) -> Cdc42(ToCRIB!1,Localiz~membr,X~GTP),Cla4(CRIB!1,X~a,
    Localiz~membr) @ 10
'Cla4_Phosph_Ste11' Cla4(CRIB!1,X~a),Cdc42(ToCRIB!1,ste50!2,X~GTP),Ste50(SAM!3,
    RA!2),Ste11(SAM!3,Ser281Ser285Thr286~u) -> Cla4(CRIB!1,X~a),Cdc42(ToCRIB!1,
    ste50!2,X~GTP),Ste50(SAM!3,RA!2),Ste11(SAM!3,Ser281Ser285Thr286~p) @ 10
'Cla4_Phosph_Ste11p' Cla4(CRIB!1,X~a),Cdc42(ToCRIB!1,ste50!2,X~GTP),Ste50(SAM!3,
    RA!2),Ste11(SAM!3,Ser281Ser285Thr286~p) -> Cla4(CRIB!1,X~a),Cdc42(ToCRIB!1,
    ste50!2,X~GTP),Ste50(SAM!3,RA!2),Ste11(SAM!3,Ser281Ser285Thr286~pp) @ 10
'Cla4_Phosph_Ste11pp' Cla4(CRIB!1,X~a),Cdc42(ToCRIB!1,ste50!2,X~GTP),Ste50(SAM
    !3,RA!2),Ste11(SAM!3,Ser281Ser285Thr286~pp) -> Cla4(CRIB!1,X~a),Cdc42(ToCRIB
    !1,ste50!2,X~GTP),Ste50(SAM!3,RA!2),Ste11(SAM!3,Ser281Ser285Thr286~ppp) @ 10
'Cla4_Phosph_Ste11/Oligo' Cla4(CRIB!1,X~a),Cdc42(ToCRIB!1,ste50!2,X~GTP),Ste50(
    SAMoligo!3,RA!2),Ste11(SAM!3,Ser281Ser285Thr286~u) -> Cla4(CRIB!1,X~a),Cdc42
    (ToCRIB!1,ste50!2,X~GTP),Ste50(SAMoligo!3,RA!2),Ste11(SAM!3,
    Ser281Ser285Thr286~p) @ 10
'Cla4_Phosph_Ste11p/Oligo' Cla4(CRIB!1,X~a),Cdc42(ToCRIB!1,ste50!2,X~GTP),Ste50(
    SAMoligo!3,RA!2),Ste11(SAM!3,Ser281Ser285Thr286~p) -> Cla4(CRIB!1,X~a),Cdc42
    (ToCRIB!1,ste50!2,X~GTP),Ste50(SAMoligo!3,RA!2),Ste11(SAM!3,
    Ser281Ser285Thr286~pp) @ 10
'Cla4_Phosph_Ste11pp/Oligo' Cla4(CRIB!1,X~a),Cdc42(ToCRIB!1,ste50!2,X~GTP),Ste50
    (SAMoligo!3,RA!2),Ste11(SAM!3,Ser281Ser285Thr286~pp) -> Cla4(CRIB!1,X~a),
    Cdc42(ToCRIB!1,ste50!2,X~GTP),Ste50(SAMoligo!3,RA!2),Ste11(SAM!3,
    Ser281Ser285Thr286~ppp) @ 10
```

```
'Cla4_Phosph_Ste11ViaBem1' Bem1(Localiz~membr,SH3b!1,CI!4),Cla4(X~a,PRR!1),Cdc42
    (ste50!2,Bem1!4,X~GTP),Ste11(SAM!3,Ser281Ser285Thr286~u),Ste50(SAM!3,RA!2)
    -> Bem1(Localiz~membr,SH3b!1,CI!4),Cla4(X~a,PRR!1),Cdc42(ste50!2,Bem1!4,X~
    GTP),Ste11(SAM!3,Ser281Ser285Thr286~p),Ste50(SAM!3,RA!2) @ 0
# 'Cdc42 activation' Cdc42(X~GDP) -> Cdc42(X~GTP) @ 0.000005 # Inactive rule
'Cdc42_DEactivation' Cdc42(X~GTP) -> Cdc42(X~GDP) @ 0.005

#######
# Chapter 'Sho1 Interaction'
'Sho1IA_Binds_Pbs2' Pbs2(Localiz~membr,sho91_102),Sho1(Localiz~membr,SH3_338,X~
    ia) <-> Pbs2(Localiz~membr,sho91_102!1),Sho1(Localiz~membr,SH3_338!1,X~ia) @
     0.0,0.1
# 'Sho1IA_Binds_Pbs2 Ref1' Sho1(SH3_342_346,Localiz~membr,SH3_338,X~ia),Pbs2(
    AA389~m,Localiz~membr,Ste11,sho91_102,Thr518~u,Ser514~u) <-> Sho1(
    SH3_342_346,Localiz~membr,SH3_338!1,X~ia),Pbs2(AA389~m,Localiz~membr,Ste11,
    sho91_102!1,Thr518~u,Ser514~u) @ 0.0005,1.0 # Inactive rule
# 'Sho1IA_Binds_Pbs2 Ref2' Sho1(Localiz~membr,SH3_338,X~ia),Pbs2(Localiz~membr,
    sho91_102,Ser514~u,Thr518~p,Ste11) <-> Sho1(Localiz~membr,SH3_338!1,X~ia),
    Pbs2(Localiz~membr,sho91_102!1,Ser514~u,Thr518~p,Ste11) @ 0.0005,1.0 #
    Inactive rule
# 'Sho1IA_Binds_Pbs2 Ref3' Sho1(Localiz~membr,SH3_338,X~ia),Pbs2(Localiz~membr,
    sho91_102,Ser514~p,Thr518~u,Ste11) <-> Sho1(Localiz~membr,SH3_338!1,X~ia),
    Pbs2(Localiz~membr,sho91_102!1,Ser514~p,Thr518~u,Ste11) @ 0.0005,1.0 #
    Inactive rule
# 'Sho1IA_Binds_Pbs2 Ref4' Sho1(Localiz~membr,SH3_338,X~ia),Pbs2(Localiz~membr,
    sho91_102,Ser514~u,Thr518~u,Ste11) <-> Sho1(Localiz~membr,SH3_338!1,X~ia),
    Pbs2(Localiz~membr,sho91_102!1,Ser514~u,Thr518~u,Ste11) @ 0.0005,1.0 #
    Inactive rule
'Sho1_BindsPbs2uu' Pbs2(Thr518,Ser514,Localiz~membr,sho91_102),Sho1(Localiz~
    membr,SH3_338,X) <-> Pbs2(Thr518,Ser514,Localiz~membr,sho91_102!1),Sho1(
    Localiz~membr,SH3_338!1,X) @ 0.0,0.0001
'Sho1A_BindsPbs2uu Ref1' Sho1(Localiz~membr,SH3_338,X~a),Pbs2(Thr518~u,Ser514~u,
    Localiz~membr,Ste11,sho91_102) <-> Sho1(Localiz~membr,SH3_338!1,X~a),Pbs2(
    Thr518~u,Ser514~u,Localiz~membr,Ste11,sho91_102!1) @ 0.0005,0.1
'Sho1A_BindsPbs2uu Ref2' Pbs2(Thr518~p,Ser514~p,Localiz~membr,Ste11,sho91_102),
    Sho1(Localiz~membr,SH3_338,X~a) <-> Pbs2(Thr518~p,Ser514~p,Localiz~membr,
    Ste11,sho91_102!1),Sho1(Localiz~membr,SH3_338!1,X~a) @ 0.0,1.0
'Sho1A_BindsPbs2uu Ref3' Pbs2(Localiz~membr,Ste11,sho91_102,Ser514~u,Thr518~u),
    Sho1(Localiz~membr,SH3_338,X~ia) <-> Pbs2(Localiz~membr,Ste11,sho91_102!1,
    Ser514~u,Thr518~u),Sho1(Localiz~membr,SH3_338!1,X~ia) @ 0.0005,0.1
'Sho1A_BindsPbs2uu Ref4' Pbs2(Localiz~membr,Ste11,sho91_102,Ser514~u,Thr518~p),
    Sho1(Localiz~membr,SH3_338,X) <-> Pbs2(Localiz~membr,Ste11,sho91_102!1,
```

```
    Ser514~u,Thr518~p),Sho1(Localiz~membr,SH3_338!1,X) @ 0.0005,0.1
'Sho1A_BindsPbs2uu Ref5' Pbs2(Localiz~membr,Ste11,sho91_102,Ser514~p,Thr518~u),
    Sho1(Localiz~membr,SH3_338,X) <-> Pbs2(Localiz~membr,Ste11,sho91_102!1,
    Ser514~p,Thr518~u),Sho1(Localiz~membr,SH3_338!1,X) @ 0.0005,0.1
'T_Ste11membr_Binds_Sho1' Sho1(SH3_342_346,Localiz~membr),Ste11(Localiz~membr,
    Sho1) <-> Sho1(SH3_342_346!1,Localiz~membr),Ste11(Localiz~membr,Sho1!1) @
    0.0,0.1
# 'T_Ste11membr_Binds_Sho1 Ref1' Sho1(SH3_342_346,Localiz~membr,X~ia),Ste11(
    Localiz~membr,Sho1,Ser281Ser285Thr286~ppp) <-> Sho1(SH3_342_346!1,Localiz~
    membr,X~ia),Ste11(Localiz~membr,Sho1!1,Ser281Ser285Thr286~ppp) @ 0.005,0.1 #
     Inactive rule
# 'T_Ste11membr_Binds_Sho1 Ref3' Sho1(SH3_342_346,Localiz~membr,X~a),Ste11(
    Localiz~membr,Sho1,Ser281Ser285Thr286~ppp) <-> Sho1(SH3_342_346!1,Localiz~
    membr,X~a),Ste11(Localiz~membr,Sho1!1,Ser281Ser285Thr286~ppp) @ 0.005,0.1 #
    Inactive rule
'T_Ste11membr_Binds_Sho1 Ref2' Sho1(SH3_342_346,Localiz~membr,X~a),Ste11(Localiz
    ~membr,Sho1,Ser281Ser285Thr286~ppp,SAM!_) <-> Sho1(SH3_342_346!1,Localiz~
    membr,X~a),Ste11(Localiz~membr,Sho1!1,Ser281Ser285Thr286~ppp,SAM!_) @
    0.0005,0.1
# 'T_Ste11membr_Binds_Sho1 Ref4' Sho1(SH3_342_346,Localiz~membr,X~ia),Ste11(
    Localiz~membr,Sho1,Ser281Ser285Thr286~ppp,SAM!_) <-> Sho1(SH3_342_346!1,
    Localiz~membr,X~ia),Ste11(Localiz~membr,Sho1!1,Ser281Ser285Thr286~ppp,SAM!_)
     @ 0.0005,0.1 # Inactive rule
'T2_Ste50_Binds_Sho1' Ste50(Localiz~membr,Sho1),Sho1(Localiz~membr,Ste50) <->
    Ste50(Localiz~membr,Sho1!1),Sho1(Localiz~membr,Ste50!1) @ 0.0,0.1
# 'T2_Ste50_Binds_Sho1 Ref1' Sho1(Localiz~membr,Ste50,X~a),Ste50(Localiz~membr,
    Sho1) <-> Sho1(Localiz~membr,Ste50!1,X~a),Ste50(Localiz~membr,Sho1!1) @
    0.0005,0.1 # Inactive rule
'T2_Ste50_Binds_Sho1 Ref2' Sho1(Localiz~membr,Ste50,X),Ste50(Localiz~membr,Sho1,
    SAM) <-> Sho1(Localiz~membr,Ste50!1,X),Ste50(Localiz~membr,Sho1!1,SAM) @
    0.0,1.0
'T2_Ste50_Binds_Sho1 Ref3' Sho1(Localiz~membr,Ste50,X),Ste50(Localiz~membr,Sho1,
    SAMoligo) <-> Sho1(Localiz~membr,Ste50!1,X),Ste50(Localiz~membr,Sho1!1,
    SAMoligo) @ 0.0,1.0
'T2_Ste50_Binds_Sho1 Ref4' Ste50(Localiz~membr,Sho1),Sho1(Localiz~membr,Ste50,X~
    ia) <-> Ste50(Localiz~membr,Sho1!1),Sho1(Localiz~membr,Ste50!1,X~ia) @
    0.0,1.0
# 'Ste50Sam_Attach_Ste11_Sho1' Sho1(Localiz~membr,Ste50!1,SH3_342_346,X),Ste11(
    Sho1,SAM!2,Localiz~membr),Ste50(Sho1!1,Localiz~membr,SAM!2) <-> Sho1(Localiz
    ~membr,Ste50!1,SH3_342_346!2,X),Ste11(Sho1!2,SAM!3,Localiz~membr),Ste50(Sho1
    !1,Localiz~membr,SAM!3) @ 10,0.01 # Inactive rule
# 'Ste50OligSam_Attach_Ste11_Sho1' Sho1(Localiz~membr,Ste50!1,SH3_342_346,X),
```

```
    Ste11(Sho1,SAM!3,Localiz~membr),Ste50(Sho1!1,Localiz~membr,SAMoligo!3) <->
    Sho1(Localiz~membr,Ste50!1,SH3_342_346!2,X),Ste11(Sho1!2,SAM!3,Localiz~membr
    ),Ste50(Sho1!1,Localiz~membr,SAMoligo!3) @ 10,0.01 # Inactive rule
'Ste11Sam_Attach_Ste50_Sho1' Sho1(Localiz~membr,Ste50,SH3_342_346!2,X~a),Ste11(
    Sho1!2,SAM!1,Localiz~membr),Ste50(Sho1,Localiz~membr,SAM!1) <-> Sho1(Localiz
    ~membr,Ste50!1,SH3_342_346!2,X~a),Ste11(Sho1!2,SAM!3,Localiz~membr),Ste50(
    Sho1!1,Localiz~membr,SAM!3) @ 20,0.0
'Ste11OligSam_Attach_Ste50_Sho1' Sho1(Localiz~membr,Ste50,SH3_342_346!2,X~a),
    Ste11(Sho1!2,SAM!3,Localiz~membr),Ste50(Sho1,Localiz~membr,SAMoligo!3) <->
    Sho1(Localiz~membr,Ste50!1,SH3_342_346!2,X~a),Ste11(Sho1!2,SAM!3,Localiz~
    membr),Ste50(Sho1!1,Localiz~membr,SAMoligo!3) @ 20,0.0
'Sho1_Brks_Ste50Ste11Sam' Sho1(Localiz~membr,Ste50!1,SH3_342_346!2,X),Ste11(Sho1
    !2,SAM!3,Localiz~membr),Ste50(Sho1!1,Localiz~membr,SAM!3) <-> Sho1(Localiz~
    membr,Ste50!1,SH3_342_346!2,X),Ste11(Sho1!2,SAM,Localiz~membr),Ste50(Sho1!1,
    Localiz~membr,SAM) @ 50,0.0
'Sho1_Brks_Ste50Ste11OligSam' Sho1(Localiz~membr,Ste50!1,SH3_342_346!2,X),Ste11(
    Sho1!2,SAM!3,Localiz~membr),Ste50(Sho1!1,Localiz~membr,SAMoligo!3) <-> Sho1(
    Localiz~membr,Ste50!1,SH3_342_346!2,X),Ste11(Sho1!2,SAM,Localiz~membr),Ste50
    (Sho1!1,Localiz~membr,SAMoligo) @ 50,0.0
'Sho1_QarternComplxSam' Sho1(SH3_342_346!2,Ste50,Localiz~membr,SH3_338!4,X),
    Ste11(Sho1!2,Pbs2),Pbs2(sho91_102!4,Ste11) -> Sho1(SH3_342_346!2,Ste50,
    Localiz~membr,SH3_338!4,X),Ste11(Sho1!2,Pbs2!5),Pbs2(sho91_102!4,Ste11!5) @
    1
# 'Sho1_quarternComplx2Sam' Sho1(A,SH3_338!3,Localiz~membr,Ste50!1),Ste50(Sho1
    !1,SAM!2),Pbs2(sho91_102!3,Ste11),Ste11(SAM!2,Pbs2) -> Sho1(A,SH3_338!3,
    Localiz~membr,Ste50!1),Ste50(Sho1!1,SAM!2),Pbs2(sho91_102!3,Ste11!4),Ste11(
    SAM!2,Pbs2!4) @ 20 # Inactive rule
# 'Sho1_quarternComplxOlig2Sam' Sho1(A,SH3_338!3,Localiz~membr,Ste50!1),Pbs2(
    sho91_102!3,Ste11),Ste11(SAM!2,Pbs2),Ste50(Sho1!1,SAMoligo!2) -> Sho1(A,
    SH3_338!3,Localiz~membr,Ste50!1),Pbs2(sho91_102!3,Ste11!4),Ste11(SAM!2,Pbs2
    !4),Ste50(Sho1!1,SAMoligo!2) @ 20 # Inactive rule
# 'Sho1_quarternComplx2Sam_Del1' Sho1(A,SH3_338!3,Localiz~membr,Ste50!1),Ste50(
    Sho1!1,SAM!2),Ste11(SAM!2,Pbs2!4),Pbs2(Ste11!4,sho91_102!3) -> Sho1(A,
    SH3_338!3,Localiz~membr,Ste50),Ste50(Sho1,SAM!2),Ste11(SAM!2,Pbs2!4),Pbs2(
    Ste11!4,sho91_102!3) @ 20 # Inactive rule
# 'Sho1_quarternComplx2OligSam_Del1' Sho1(A,SH3_338!3,Localiz~membr,Ste50!1),
    Pbs2(sho91_102!3,Ste11!4),Ste11(SAM!2,Pbs2!4),Ste50(Sho1!1,SAMoligo!2) ->
    Sho1(A,SH3_338!3,Localiz~membr,Ste50),Pbs2(sho91_102!3,Ste11!4),Ste11(SAM!2,
    Pbs2!4),Ste50(Sho1,SAMoligo!2) @ 20 # Inactive rule
'Ste11Pbs2_Unbind_Sho1' Sho1(SH3_342_346!2,Localiz~membr,SH3_338!4,X),Pbs2(
    sho91_102!4,Ste11!5),Ste11(Sho1!2,Pbs2!5) -> Sho1(SH3_342_346,Localiz~membr,
    SH3_338!4,X),Pbs2(sho91_102!4,Ste11!1),Ste11(Sho1,Pbs2!1) @ 50
```

```
'Sho1_Unbind_Pbs2Ste11' Sho1(SH3_338!1),Pbs2(sho91_102!1,Ste11!2),Ste11(Pbs2!2)
    -> Sho1(SH3_338),Pbs2(sho91_102,Ste11!1),Ste11(Pbs2!1) @ 1
# 'Ste11umembr_Binds_Sho1act' Sho1(Localiz~membr,SH3_342_346,X~A),Ste11(Localiz~
    membr,Sho1) <-> Sho1(Localiz~membr,SH3_342_346!1,X~A),Ste11(Localiz~membr,
    Sho1!1) @ 0.00005,0.1 # Inactive rule

#######
# Chapter 'Opy2 Interactions'
'Ste50membr_Binds_Opy2CRA' Opy2(CR-A,Localiz~membr),Ste50(RA,Localiz~membr,
    S155_196_202_248) <-> Opy2(CR-A!1,Localiz~membr),Ste50(RA!1,Localiz~membr,
    S155_196_202_248) @ 0.0,0.01
'Ste50membr_Binds_Opy2CRA Ref1' Opy2(CR-A,Localiz~membr),Ste50(RA,Localiz~membr,
    S155_196_202_248~u) <-> Opy2(CR-A!1,Localiz~membr),Ste50(RA!1,Localiz~membr,
    S155_196_202_248~u) @ 0.00005,0.01
# 'Ste50membr_Binds_Opy2CRA Ref2' Opy2(CR-A,Localiz~membr),Ste50(RA,Localiz~
    membr,S155_196_202_248,SAM!_) <-> Opy2(CR-A!1,Localiz~membr),Ste50(RA!1,
    Localiz~membr,S155_196_202_248,SAM!_) @ 0.0,0.1 # Inactive rule
# 'Ste50membr_Binds_Opy2CRA Ref3' Opy2(CR-A,Localiz~membr),Ste50(RA,Localiz~
    membr,S155_196_202_248,SAMoligo!_) <-> Opy2(CR-A!1,Localiz~membr),Ste50(RA
    !1,Localiz~membr,S155_196_202_248,SAMoligo!_) @ 0.0,0.1 # Inactive rule
# 'Ste50membr_Binds_Opy2CRA Ref4' Opy2(CR-A,Localiz~membr),Ste50(RA,Localiz~
    membr,S155_196_202_248~u,SAM,SAMoligo) <-> Opy2(CR-A!1,Localiz~membr),Ste50(
    RA!1,Localiz~membr,S155_196_202_248~u,SAM,SAMoligo) @ 0.00005,0.01 #
    Inactive rule
'Ste50pmembr_Binds_Opy2CRA' Opy2(CR-A,Localiz~membr),Ste50(RA,Localiz~membr,
    S155_196_202_248~p) <-> Opy2(CR-A!1,Localiz~membr),Ste50(RA!1,Localiz~membr,
    S155_196_202_248~p) @ 0.0000005,0.1
'Ste50ppmembr_Binds_Opy2CRA' Opy2(CR-A,Localiz~membr),Ste50(RA,Localiz~membr,
    S155_196_202_248~pp) <-> Opy2(CR-A!1,Localiz~membr),Ste50(RA!1,Localiz~membr
    ,S155_196_202_248~pp) @ 0.00000005,0.1
'Ste50pppmembr_Binds_Opy2CRA' Opy2(CR-A,Localiz~membr),Ste50(RA,Localiz~membr,
    S155_196_202_248~ppp) <-> Opy2(CR-A!1,Localiz~membr),Ste50(RA!1,Localiz~
    membr,S155_196_202_248~ppp) @ 0.000000005,0.1
'Ste50ppppmembr_Binds_Opy2CRA' Opy2(CR-A,Localiz~membr),Ste50(RA,Localiz~membr,
    S155_196_202_248~pppp) <-> Opy2(CR-A!1,Localiz~membr),Ste50(RA!1,Localiz~
    membr,S155_196_202_248~pppp) @ 0,1
'Ste50membr_Binds_Opy2CRB' Opy2(Localiz~membr,CR-B),Ste50(RA,Localiz~membr,
    S155_196_202_248) <-> Opy2(Localiz~membr,CR-B!1),Ste50(RA!1,Localiz~membr,
    S155_196_202_248) @ 0.0,0.01
'Ste50membr_Binds_Opy2CRB Ref1' Opy2(Localiz~membr,CR-B~p),Ste50(RA,Localiz~
    membr,S155_196_202_248~u) <-> Opy2(Localiz~membr,CR-B~p!1),Ste50(RA!1,
    Localiz~membr,S155_196_202_248~u) @ 0.00005,0.01
```

```
# 'Ste50membr_Binds_Opy2CRB Ref2' Opy2(Localiz~membr,CR-B~p),Ste50(RA,Localiz~
    membr,S155_196_202_248,SAM!_) <-> Opy2(Localiz~membr,CR-B~p!1),Ste50(RA!1,
    Localiz~membr,S155_196_202_248,SAM!_) @ 0.0,0.1 # Inactive rule
# 'Ste50membr_Binds_Opy2CRB Ref3' Opy2(Localiz~membr,CR-B~p),Ste50(RA,Localiz~
    membr,S155_196_202_248,SAMoligo!_) <-> Opy2(Localiz~membr,CR-B~p!1),Ste50(RA
    !1,Localiz~membr,S155_196_202_248,SAMoligo!_) @ 0.0,0.1 # Inactive rule
# 'Ste50membr_Binds_Opy2CRB Ref4' Opy2(Localiz~membr,CR-B),Ste50(RA,Localiz~
    membr,S155_196_202_248~u) <-> Opy2(Localiz~membr,CR-B!1),Ste50(RA!1,Localiz~
    membr,S155_196_202_248~u) @ 0.00005,0.01 # Inactive rule
'Ste50pmembr_Binds_Opy2CRB' Opy2(Localiz~membr,CR-B~p),Ste50(RA,Localiz~membr,
    S155_196_202_248~p) <-> Opy2(Localiz~membr,CR-B~p!1),Ste50(RA!1,Localiz~
    membr,S155_196_202_248~p) @ 0.0000005,0.1
'Ste50ppmembr_Binds_Opy2CRB' Opy2(Localiz~membr,CR-B~p),Ste50(RA,Localiz~membr,
    S155_196_202_248~pp) <-> Opy2(Localiz~membr,CR-B~p!1),Ste50(RA!1,Localiz~
    membr,S155_196_202_248~pp) @ 0.00000005,0.1
'Ste50pppmembr_Binds_Opy2CRB' Opy2(Localiz~membr,CR-B~p),Ste50(RA,Localiz~membr,
    S155_196_202_248~ppp) <-> Opy2(Localiz~membr,CR-B~p!1),Ste50(RA!1,Localiz~
    membr,S155_196_202_248~ppp) @ 0.000000005,0.1
'Ste50ppppmembr_Binds_Opy2CRB' Opy2(Localiz~membr,CR-B~p),Ste50(RA,Localiz~membr
    ,S155_196_202_248~pppp) <-> Opy2(Localiz~membr,CR-B~p!1),Ste50(RA!1,Localiz~
    membr,S155_196_202_248~pppp) @ 0,1
'Ste50membr_Binds_Opy2CRD' Opy2(CR-D,Localiz~membr),Ste50(RA,Localiz~membr,
    S155_196_202_248) <-> Opy2(CR-D!1,Localiz~membr),Ste50(RA!1,Localiz~membr,
    S155_196_202_248) @ 0.0,0.01
'Ste50membr_Binds_Opy2CRD Ref1' Opy2(CR-D,Localiz~membr),Ste50(RA,Localiz~membr,
    S155_196_202_248~u) <-> Opy2(CR-D!1,Localiz~membr),Ste50(RA!1,Localiz~membr,
    S155_196_202_248~u) @ 0.00005,0.01
# 'Ste50membr_Binds_Opy2CRD Ref2' Opy2(CR-D,Localiz~membr),Ste50(RA,Localiz~
    membr,S155_196_202_248,SAM!_) <-> Opy2(CR-D!1,Localiz~membr),Ste50(RA!1,
    Localiz~membr,S155_196_202_248,SAM!_) @ 0.0,0.1 # Inactive rule
# 'Ste50membr_Binds_Opy2CRD Ref3' Opy2(CR-D,Localiz~membr),Ste50(RA,Localiz~
    membr,S155_196_202_248,SAMoligo!_) <-> Opy2(CR-D!1,Localiz~membr),Ste50(RA
    !1,Localiz~membr,S155_196_202_248,SAMoligo!_) @ 0.0,0.1 # Inactive rule
# 'Ste50membr_Binds_Opy2CRD Ref4' Opy2(CR-D,Localiz~membr),Ste50(RA,Localiz~
    membr,S155_196_202_248~u,SAMoligo,SAM) <-> Opy2(CR-D!1,Localiz~membr),Ste50(
    RA!1,Localiz~membr,S155_196_202_248~u,SAMoligo,SAM) @ 0.00005,0.01 #
    Inactive rule
'Ste50pmembr_Binds_Opy2CRD' Opy2(CR-D,Localiz~membr),Ste50(RA,Localiz~membr,
    S155_196_202_248~p) <-> Opy2(CR-D!1,Localiz~membr),Ste50(RA!1,Localiz~membr,
    S155_196_202_248~p) @ 0.0000005,0.1
'Ste50ppmembr_Binds_Opy2CRD' Opy2(CR-D,Localiz~membr),Ste50(RA,Localiz~membr,
    S155_196_202_248~pp) <-> Opy2(CR-D!1,Localiz~membr),Ste50(RA!1,Localiz~membr
```

```
    ,S155_196_202_248~pp) @ 0.00000005,0.1
'Ste50pppmembr_Binds_Opy2CRD' Opy2(CR-D,Localiz~membr),Ste50(RA,Localiz~membr,
    S155_196_202_248~ppp) <-> Opy2(CR-D!1,Localiz~membr),Ste50(RA!1,Localiz~
    membr,S155_196_202_248~ppp) @ 0.000000005,0.1
'Ste50ppppmembr_Binds_Opy2CRD' Opy2(CR-D,Localiz~membr),Ste50(RA,Localiz~membr,
    S155_196_202_248~pppp) <-> Opy2(CR-D!1,Localiz~membr),Ste50(RA!1,Localiz~
    membr,S155_196_202_248~pppp) @ 0,1
# 'Yck1_2_Activation' Yck1_2(A~ia) -> Yck1_2(A~a) @ 0 # Inactive rule
'Yck1_2_Binds_Opy2' Yck1_2(Opy2),Opy2(Localiz~membr,Yck) <-> Yck1_2(Opy2!1),Opy2
    (Localiz~membr,Yck!1) @ 0.000005,0.1
# 'Yck1_2_Binds_Opy2 Ref1' Opy2(Localiz~membr,Yck),Yck1_2(Opy2,A~a) <-> Opy2(
    Localiz~membr,Yck!1),Yck1_2(Opy2!1,A~a) @ 0.000005,0.1 # Inactive rule
'Yck1_2_Phosph_Opy2CRB' Opy2(Localiz~membr,CR-B~u,Yck!1),Yck1_2(Opy2!1,A~a) ->
    Opy2(Localiz~membr,CR-B~p,Yck!1),Yck1_2(Opy2!1,A~a) @ 10

#######
# Chapter 'Ste50Ste11OligDim'
'2Ste50Dim2_Oligomerization1' Ste50(SAMoligo,SAM!1,Localiz~cyto),Ste50(SAM,
    SAMoligo!2,Localiz~cyto),Ste50(SAM!1,Localiz~cyto),Ste50(SAMoligo!2,Localiz~
    cyto) <-> Ste50(SAMoligo!2,SAM!1,Localiz~cyto),Ste50(SAM!2,SAMoligo!3,
    Localiz~cyto),Ste50(SAM!1,Localiz~cyto),Ste50(SAMoligo!3,Localiz~cyto) @
    0.0000001,0.0001
'2Ste50OligoDim_Oligomerization2' Ste50(SAM,SAMoligo!1,Localiz~cyto),Ste50(SAM
    !1,Localiz~cyto),Ste50(SAMoligo,SAM!2,Localiz~cyto),Ste50(SAMoligo!2,Localiz
    ~cyto) <-> Ste50(SAM!2,SAMoligo!1,Localiz~cyto),Ste50(SAM!1,Localiz~cyto),
    Ste50(SAMoligo!2,SAM!3,Localiz~cyto),Ste50(SAMoligo!3,Localiz~cyto) @
    0.0000001,0.0001
'2Ste50OligoDim_Oligomerization1' Ste50(SAM,SAMoligo!1,Localiz~cyto),Ste50(SAM
    !1,Localiz~cyto),Ste50(SAMoligo!2,SAM,Localiz~cyto),Ste50(SAM!2,Localiz~cyto
    ) <-> Ste50(SAM!2,SAMoligo!1,Localiz~cyto),Ste50(SAM!1,Localiz~cyto),Ste50(
    SAMoligo!3,SAM!2,Localiz~cyto),Ste50(SAM!3,Localiz~cyto) @ 0.0000001,0.0001
'2Ste50OligoDim2_Oligomerization2' Ste50(SAM,SAMoligo!1,Localiz~cyto),Ste50(SAM
    !1,Localiz~cyto),Ste50(SAMoligo,SAM!2,Localiz~cyto),Ste50(SAM!2,Localiz~cyto
    ) <-> Ste50(SAM!2,SAMoligo!1,Localiz~cyto),Ste50(SAM!1,Localiz~cyto),Ste50(
    SAMoligo!2,SAM!3,Localiz~cyto),Ste50(SAM!3,Localiz~cyto) @ 0.0000001,0.0001
'2Ste50OligoDim2_Oligomerization1' Ste50(SAM,SAMoligo!1,Localiz~cyto),Ste50(SAM
    !1,Localiz~cyto),Ste50(SAMoligo!2,SAM,Localiz~cyto),Ste50(SAMoligo!2,Localiz
    ~cyto) <-> Ste50(SAM!2,SAMoligo!1,Localiz~cyto),Ste50(SAM!1,Localiz~cyto),
    Ste50(SAMoligo!3,SAM!2,Localiz~cyto),Ste50(SAMoligo!3,Localiz~cyto) @
    0.0000001,0.0001
'2Ste50Dim_Oligomerization2' Ste50(SAM!1,Localiz~cyto),Ste50(SAMoligo,SAM!2,
    Localiz~cyto),Ste50(SAM!1,SAMoligo,Localiz~cyto),Ste50(SAM!2,Localiz~cyto)
```

```
    <-> Ste50(SAM!1,Localiz~cyto),Ste50(SAMoligo!2,SAM!3,Localiz~cyto),Ste50(SAM
    !1,SAMoligo!2,Localiz~cyto),Ste50(SAM!3,Localiz~cyto) @ 0.0000001,0.0001
'Ste50_Oligomerization2_new' Ste50(SAM!1,SAMoligo,Localiz~cyto),Ste50(SAM!1,
    Localiz~cyto),Ste50(SAMoligo,SAM,Localiz~cyto) <-> Ste50(SAM!1,SAMoligo!2,
    Localiz~cyto),Ste50(SAM!1,Localiz~cyto),Ste50(SAMoligo!2,SAM,Localiz~cyto) @
     0.0000001,0.0001
'Ste50Oligo_Oligomerization2' Ste50(SAM,SAMoligo!1,Localiz~cyto),Ste50(SAM!1,
    Localiz~cyto),Ste50(SAMoligo,SAM,Localiz~cyto) <-> Ste50(SAM!2,SAMoligo!1,
    Localiz~cyto),Ste50(SAM!1,Localiz~cyto),Ste50(SAMoligo!2,SAM,Localiz~cyto) @
     0.0000001,0.0001
'2Ste50Dim_Oligomerization1' Ste50(SAM!1,SAMoligo,Localiz~cyto),Ste50(SAM,
    SAMoligo!2,Localiz~cyto),Ste50(SAM!1,Localiz~cyto),Ste50(SAM!2,Localiz~cyto)
     <-> Ste50(SAM!1,SAMoligo!2,Localiz~cyto),Ste50(SAM!2,SAMoligo!3,Localiz~
    cyto),Ste50(SAM!1,Localiz~cyto),Ste50(SAM!3,Localiz~cyto) @ 0.0000001,0.0001
'2Ste50Dim2_Oligomerization2' Ste50(SAMoligo,SAM!2,Localiz~cyto),Ste50(SAMoligo,
    SAM!1,Localiz~cyto),Ste50(SAM!1,Localiz~cyto),Ste50(SAMoligo!2,Localiz~cyto)
     <-> Ste50(SAMoligo!2,SAM!3,Localiz~cyto),Ste50(SAMoligo!2,SAM!1,Localiz~
    cyto),Ste50(SAM!1,Localiz~cyto),Ste50(SAMoligo!3,Localiz~cyto) @
    0.0000001,0.0001
'Ste50Oligo_Oligomerization1' Ste50(SAM,SAMoligo!1,Localiz~cyto),Ste50(SAM!1,
    Localiz~cyto),Ste50(SAMoligo,SAM,Localiz~cyto) <-> Ste50(SAM!2,SAMoligo!1,
    Localiz~cyto),Ste50(SAM!1,Localiz~cyto),Ste50(SAMoligo,SAM!2,Localiz~cyto) @
     0.0000001,0.0001
'Ste50_Oligomerization1_new' Ste50(SAM!1,Localiz~cyto),Ste50(SAM!1,SAMoligo,
    Localiz~cyto),Ste50(SAM,SAMoligo,Localiz~cyto) <-> Ste50(SAM!1,Localiz~cyto)
    ,Ste50(SAM!1,SAMoligo!2,Localiz~cyto),Ste50(SAM!2,SAMoligo,Localiz~cyto) @
    0.0000001,0.0001
'Ste50_Dimerization_new' Ste50(SAM,SAMoligo,Localiz~cyto),Ste50(SAM,SAMoligo,
    Localiz~cyto) <-> Ste50(SAM!1,SAMoligo,Localiz~cyto),Ste50(SAM!1,SAMoligo,
    Localiz~cyto) @ 0.0000001,0.0001
'Ste50_Dimerization_Oligo' Ste50(SAM,SAMoligo,Localiz~cyto),Ste50(SAM,SAMoligo,
    Localiz~cyto) <-> Ste50(SAM!1,SAMoligo,Localiz~cyto),Ste50(SAM,SAMoligo!1,
    Localiz~cyto) @ 0.0000001,0.0001
'Ste50_Binds_Ste11_SAM' Ste11(SAM,Localiz~cyto),Ste50(SAM,SAMoligo,Localiz~cyto)
     <-> Ste11(SAM!2,Localiz~cyto),Ste50(SAM!2,SAMoligo,Localiz~cyto) @
    0.000005,0.0001
'Ste50_Binds_Ste11_OligoSAM' Ste50(SAMoligo,SAM,Localiz~cyto),Ste11(SAM,Localiz~
    cyto) <-> Ste50(SAMoligo!1,SAM,Localiz~cyto),Ste11(SAM!1,Localiz~cyto) @
    0.000005,0.0001
'Ste11_Binds_Ste50Ste11' Ste11(SAM,Localiz~cyto),Ste50(SAM,SAMoligo!1,Localiz~
    cyto),Ste11(SAM!1,Localiz~cyto) <-> Ste11(SAM!2,Localiz~cyto),Ste50(SAM!2,
    SAMoligo!1,Localiz~cyto),Ste11(SAM!1,Localiz~cyto) @ 0.000005,0.00001
```

```
'Ste11_Binds_Ste50Ste11_Oligo' Ste11(SAM!1,Localiz~cyto),Ste50(SAMoligo,SAM!1,
    Localiz~cyto),Ste11(SAM,Localiz~cyto) <-> Ste11(SAM!2,Localiz~cyto),Ste50(
    SAMoligo!1,SAM!2,Localiz~cyto),Ste11(SAM!1,Localiz~cyto) @ 0.000005,0.00001
'Ste50Oligmembr_Binds_Ste11membr' Ste11(SAM,Localiz~membr),Ste50(Localiz~membr,
    SAMoligo,SAM) <-> Ste11(SAM!1,Localiz~membr),Ste50(Localiz~membr,SAMoligo!1,
    SAM) @ 0.00005,0.0001
'Ste50OligmembrOpyCRA_Binds_Ste11membr' Opy2(CR-A!1),Ste11(SAM,Localiz~membr),
    Ste50(Localiz~membr,SAMoligo,RA!1) <-> Opy2(CR-A!1),Ste11(SAM!2,Localiz~
    membr),Ste50(Localiz~membr,SAMoligo!2,RA!1) @ 0.0,0.0001
'Ste50OligmembrOpyCRA_Binds_Ste11membr Ref1' Opy2(CR-A!1),Ste50(Localiz~membr,
    SAMoligo,RA!1),Ste11(SAM,Localiz~membr,Pbs2,Sho1) <-> Opy2(CR-A!1),Ste50(
    Localiz~membr,SAMoligo!2,RA!1),Ste11(SAM!2,Localiz~membr,Pbs2,Sho1) @
    0.005,0.0001
'Ste50OligmembrOpyCRB_Binds_Ste11membr' Ste11(SAM,Localiz~membr),Ste50(Localiz~
    membr,SAMoligo,RA!1),Opy2(CR-B~p!1) <-> Ste11(SAM!1,Localiz~membr),Ste50(
    Localiz~membr,SAMoligo!1,RA!2),Opy2(CR-B~p!2) @ 0.0,0.0001
'Ste50OligmembrOpyCRB_Binds_Ste11membr Ref1' Ste50(Localiz~membr,SAMoligo,RA!1),
    Opy2(CR-B~p!1),Ste11(SAM,Localiz~membr,Sho1,Pbs2) <-> Ste50(Localiz~membr,
    SAMoligo!1,RA!2),Opy2(CR-B~p!2),Ste11(SAM!1,Localiz~membr,Sho1,Pbs2) @
    0.005,0.0001
'Ste50OligmembrOpyCRD_Binds_Ste11membr' Ste11(SAM,Localiz~membr),Ste50(Localiz~
    membr,SAMoligo,RA!1),Opy2(CR-D!1) <-> Ste11(SAM!1,Localiz~membr),Ste50(
    Localiz~membr,SAMoligo!1,RA!2),Opy2(CR-D!2) @ 0.0,0.0001
'Ste50OligmembrOpyCRD_Binds_Ste11membr Ref1' Ste50(Localiz~membr,SAMoligo,RA!1),
    Opy2(CR-D!1),Ste11(SAM,Localiz~membr,Sho1,Pbs2) <-> Ste50(Localiz~membr,
    SAMoligo!1,RA!2),Opy2(CR-D!2),Ste11(SAM!1,Localiz~membr,Sho1,Pbs2) @
    0.005,0.0001
'Ste50membr_Binds_Ste11membr' Ste11(SAM,Localiz~membr),Ste50(Localiz~membr,SAM,
    SAMoligo) <-> Ste11(SAM!2,Localiz~membr),Ste50(Localiz~membr,SAM!2,SAMoligo)
     @ 0.00005,0.0001
'Ste50membrOpy2A_Binds_Ste11membr' Ste11(SAM,Localiz~membr),Opy2(CR-A!1),Ste50(
    Localiz~membr,SAM,RA!1) <-> Ste11(SAM!2,Localiz~membr),Opy2(CR-A!1),Ste50(
    Localiz~membr,SAM!2,RA!1) @ 0.0,0.0001
'Ste50membrOpy2A_Binds_Ste11membr Ref1' Opy2(CR-A!1),Ste50(Localiz~membr,SAM,RA
    !1),Ste11(SAM,Localiz~membr,Sho1,Pbs2) <-> Opy2(CR-A!1),Ste50(Localiz~membr,
    SAM!2,RA!1),Ste11(SAM!2,Localiz~membr,Sho1,Pbs2) @ 0.005,0.0001
'Ste50membrOpy2B_Binds_Ste11membr' Ste11(SAM,Localiz~membr),Opy2(CR-B~p!1),Ste50
    (Localiz~membr,SAM,RA!1) <-> Ste11(SAM!2,Localiz~membr),Opy2(CR-B~p!1),Ste50
    (Localiz~membr,SAM!2,RA!1) @ 0.0,0.0001
'Ste50membrOpy2B_Binds_Ste11membr Ref1' Opy2(CR-B~p!1),Ste50(Localiz~membr,SAM,
    RA!1),Ste11(SAM,Localiz~membr,Sho1,Pbs2) <-> Opy2(CR-B~p!1),Ste50(Localiz~
    membr,SAM!2,RA!1),Ste11(SAM!2,Localiz~membr,Sho1,Pbs2) @ 0.005,0.0001
```

```
'Ste50membrOpy2D_Binds_Ste11membr' Ste11(SAM,Localiz~membr),Ste50(Localiz~membr,
    SAM,RA!1),Opy2(CR-D!1) <-> Ste11(SAM!2,Localiz~membr),Ste50(Localiz~membr,
    SAM!2,RA!1),Opy2(CR-D!1) @ 0.0,0.0001
'Ste50membrOpy2D_Binds_Ste11membr Ref1' Ste50(Localiz~membr,SAM,RA!1),Opy2(CR-D
    !1),Ste11(SAM,Localiz~membr,Sho1,Pbs2) <-> Ste50(Localiz~membr,SAM!2,RA!1),
    Opy2(CR-D!1),Ste11(SAM!2,Localiz~membr,Sho1,Pbs2) @ 0.005,0.0001
'Ste11membr_Binds_Ste50Ste11membr' Ste11(SAM,Localiz~membr),Ste50(SAM,SAMoligo
    !1,Localiz~membr),Ste11(SAM!1,Localiz~membr) <-> Ste11(SAM!2,Localiz~membr),
    Ste50(SAM!2,SAMoligo!1,Localiz~membr),Ste11(SAM!1,Localiz~membr) @
    0.00005,0.00001
'Ste11membr_Binds_Ste50Ste11membr_Oligo' Ste11(SAM!1,Localiz~membr),Ste50(
    SAMoligo,SAM!1,Localiz~membr),Ste11(SAM,Localiz~membr) <-> Ste11(SAM!2,
    Localiz~membr),Ste50(SAMoligo!1,SAM!2,Localiz~membr),Ste11(SAM!1,Localiz~
    membr) @ 0.00005,0.00001

#######
# Chapter 'Transport_CytoMembr'
'Ptc23_Transp_CytoNuc' Ptc23(Localiz~cyto,P,Hog1) <-> Ptc23(Localiz~nuc,P,Hog1)
    @ 1,10
'Ptc1_Transp_CytNuc' Ptc1(Nbp2,Localiz~cyto,KIM,Hog,P) <-> Ptc1(Nbp2,Localiz~nuc
    ,KIM,Hog,P) @ 1,10
# 'Hog1pp_Tr_Nuc' Hog1(Thr174,Tyr176,Localiz~cyto,Ste50,CD,TF,Phosphat,Ptc23,
    Kinase) -> Hog1(Thr174,Tyr176,Localiz~nuc,Ste50,CD,TF,Phosphat,Ptc23,Kinase)
     @ 1 # Inactive rule
'Hog1pp_Tr_Nuc Ref1' Hog1(Thr174~p,Tyr176~p,Localiz~cyto,Ste50,CD,TF,Phosphat,
    Ptc23,Kinase,Sho1) -> Hog1(Thr174~p,Tyr176~p,Localiz~nuc,Ste50,CD,TF,
    Phosphat,Ptc23,Kinase,Sho1) @ 30
'Hog1nuc_Tr_Cyto' Hog1(Localiz~nuc,TF,CD,Ste50,Thr174,Tyr176,Phosphat,Ptc23,
    Kinase,Sho1) -> Hog1(Localiz~cyto,TF,CD,Ste50,Thr174,Tyr176,Phosphat,Ptc23,
    Kinase,Sho1) @ 10
# 'PtpCyto_Nuc' Ptp(mapk,Localiz~cyto,Hog1,P) <-> Ptp(mapk,Localiz~nuc,Hog1,P) @
     1.0,10 # Inactive rule
'Bem1membr_cyto' Bem1(Localiz~membr,Ste5,SH3b,PB1,CI) <-> Bem1(Localiz~cyto,Ste5
    ,SH3b,PB1,CI) @ 10,1
'Ypd1membr_Cyto' Ypd1(Localiz~membr,Sln1,Ssk1) <-> Ypd1(Localiz~cyto,Sln1,Ssk1)
    @ 10,1
'Ssk1DimerMembr_Cyto' Ssk1(Localiz~membr,Dimer!1,Ssk2_22,Ypd1),Ssk1(Ypd1,Dimer
    !1,Ssk2_22,Localiz~membr) <-> Ssk1(Localiz~cyto,Dimer!1,Ssk2_22,Ypd1),Ssk1(
    Ypd1,Dimer!1,Ssk2_22,Localiz~cyto) @ 10,1
'Ssk1Membr_Cyto' Ssk1(Ypd1,Ssk2_22,Dimer,Localiz~membr) <-> Ssk1(Ypd1,Ssk2_22,
    Dimer,Localiz~cyto) @ 10,1
'Pbs2membr_Transp_cyto' Pbs2(sho91_102,Localiz~membr,Ste11,HBD-I,ssk44_57,Nbp2)
```

```
    <-> Pbs2(sho91_102,Localiz~cyto,Ste11,HBD-I,ssk44_57,Nbp2) @ 20,2
# 'Pbs2Ssk2membr_Transp_cyto' Pbs2(sho91_102,Localiz~membr,Ste11,Nbp2,HBD-I,
    ssk44_57!1),Ssk2(Thr1460~u,Localiz~membr,pbs2!1,294_413) <-> Pbs2(sho91_102,
    Localiz~cyto,Ste11,Nbp2,HBD-I,ssk44_57!1),Ssk2(Thr1460~u,Localiz~cyto,pbs2
    !1,294_413) @ 0.001,0.0001 # Inactive rule
# 'Pbs2Ssk22membr_Transp_cyto' Pbs2(sho91_102,Localiz~membr,Ste11,Nbp2,HBD-I,
    ssk44_57!1),Ssk22(Thr1460~u,Localiz~membr,98_179,pbs2!1) <-> Pbs2(sho91_102,
    Localiz~cyto,Ste11,Nbp2,HBD-I,ssk44_57!1),Ssk22(Thr1460~u,Localiz~cyto,98
    _179,pbs2!1) @ 0.01,0.01 # Inactive rule
'Pbs2Ste11membr_Transp_cyto' Pbs2(Nbp2,Localiz~membr,HBD-I,sho91_102,Ste11!1,
    ssk44_57,Ptc23),Ste11(Sho1,Cla4,SAM,Localiz~membr,Ste20,Pbs2!1) <-> Pbs2(
    Nbp2,Localiz~cyto,HBD-I,sho91_102,Ste11!1,ssk44_57,Ptc23),Ste11(Sho1,Cla4,
    SAM,Localiz~cyto,Ste20,Pbs2!1) @ 10,1
'Pbs2Ste11Ste50Ste11membr_Transp_cyto' Pbs2(Nbp2,Localiz~membr,HBD-I,sho91_102,
    Ste11!1),Ste11(Sho1,Cla4,SAM!2,Localiz~membr,Ste20,Pbs2!1),Ste50(Sho1,RA,
    Localiz~membr,SAM!2,SAMoligo!3),Ste11(Sho1,Pbs2,Cla4,Localiz~membr,Ste20,SAM
    !3) <-> Pbs2(Nbp2,Localiz~cyto,HBD-I,sho91_102,Ste11!1),Ste11(Sho1,Cla4,SAM
    !2,Localiz~cyto,Ste20,Pbs2!1),Ste50(Sho1,RA,Localiz~cyto,SAM!2,SAMoligo!3),
    Ste11(Sho1,Pbs2,Cla4,Localiz~cyto,Ste20,SAM!3) @ 10,1
'Pbs2Ste11oligSte50Ste11membr_Transp_cyto' Pbs2(HBD-I,Nbp2,Localiz~membr,
    ssk44_57,sho91_102,Ste11!1),Ste11(Sho1,Cla4,SAM!2,Localiz~membr,Ste20,Pbs2
    !1),Ste50(Sho1,SAM!3,Localiz~membr,RA,SAMoligo!2),Ste11(Sho1,Pbs2,Cla4,
    Localiz~membr,Ste20,SAM!3) <-> Pbs2(HBD-I,Nbp2,Localiz~cyto,ssk44_57,
    sho91_102,Ste11!1),Ste11(Sho1,Cla4,SAM!2,Localiz~cyto,Ste20,Pbs2!1),Ste50(
    Sho1,SAM!3,Localiz~cyto,RA,SAMoligo!2),Ste11(Sho1,Pbs2,Cla4,Localiz~cyto,
    Ste20,SAM!3) @ 10,1
'Pbs2Nbp2Ptc1membr_Transp_cyto' Pbs2(HBD-I,ssk44_57,Ste11,Localiz~membr,
    sho91_102,Nbp2!2),Ptc1(Localiz~membr,KIM,Nbp2!1),Nbp2(Localiz~membr,Ptc1!1,
    Pbs2!2) <-> Pbs2(HBD-I,ssk44_57,Ste11,Localiz~cyto,sho91_102,Nbp2!2),Ptc1(
    Localiz~cyto,KIM,Nbp2!1),Nbp2(Localiz~cyto,Ptc1!1,Pbs2!2) @ 10,1
'Pbs2Ste11Ste50oligTransp_cyto' Pbs2(HBD-I,Localiz~membr,ssk44_57,sho91_102,
    Ste11!1,Nbp2,Ptc23),Ste11(Sho1,Cla4,SAM!2,Localiz~membr,Ste20,Pbs2!1),Ste50(
    Sho1,SAM,Localiz~membr,RA,SAMoligo!2) <-> Pbs2(HBD-I,Localiz~cyto,ssk44_57,
    sho91_102,Ste11!1,Nbp2,Ptc23),Ste11(Sho1,Cla4,SAM!2,Localiz~cyto,Ste20,Pbs2
    !1),Ste50(Sho1,SAM,Localiz~cyto,RA,SAMoligo!2) @ 10,1
'Pbs2Ste11Ste50_Transp_cyto' Ste11(Sho1,Cla4,SAM!2,Localiz~membr,Ste20,Pbs2!1),
    Ste50(Sho1,RA,Localiz~membr,SAM!2,SAMoligo),Pbs2(Localiz~membr,HBD-I,
    sho91_102,Ste11!1,Nbp2,Ptc23) <-> Ste11(Sho1,Cla4,SAM!2,Localiz~cyto,Ste20,
    Pbs2!1),Ste50(Sho1,RA,Localiz~cyto,SAM!2,SAMoligo),Pbs2(Localiz~cyto,HBD-I,
    sho91_102,Ste11!1,Nbp2,Ptc23) @ 10,1
'Cdc24membr_cyto' Cdc24(sho1,DH,Localiz~membr,PB1,PhosphSites,PH) <-> Cdc24(sho1
    ,DH,Localiz~cyto,PB1,PhosphSites,PH) @ 10,1
```

```
'Ste20membr_cyto' Ste20(Localiz~membr,Ste11,CRIB,PRR) <-> Ste20(Localiz~cyto,
    Ste11,CRIB,PRR) @ 10,1
'Cla4membr_cyto' Cla4(CRIB,Localiz~membr,Ste11,PRR) <-> Cla4(CRIB,Localiz~cyto,
    Ste11,PRR) @ 10,1
'Ste11membr_cyto' Ste11(SAM,Localiz~membr,Sho1,Pbs2,Cla4,Ste20) <-> Ste11(SAM,
    Localiz~cyto,Sho1,Pbs2,Cla4,Ste20) @ 10,1
'Ste50Ste11Ste11membr_cyto' Ste50(RA,Localiz~membr,SAM!3,Sho1,SAMoligo!2),Ste11(
    SAM!2,Localiz~membr,Sho1,Cla4,Pbs2,Ste20),Ste11(SAM!3,Localiz~membr,Sho1,
    Pbs2,Cla4,Ste20) <-> Ste50(RA,Localiz~cyto,SAM!3,Sho1,SAMoligo!2),Ste11(SAM
    !2,Localiz~cyto,Sho1,Cla4,Pbs2,Ste20),Ste11(SAM!3,Localiz~cyto,Sho1,Pbs2,
    Cla4,Ste20) @ 10,1
'Ste50Ste11membr_cyto' Ste11(SAM!2,Localiz~membr,Sho1,Pbs2,Cla4,Ste20),Ste50(RA,
    Localiz~membr,SAMoligo,SAM!2,Sho1) <-> Ste11(SAM!2,Localiz~cyto,Sho1,Pbs2,
    Cla4,Ste20),Ste50(RA,Localiz~cyto,SAMoligo,SAM!2,Sho1) @ 10,1
'Ste50Ste11OligMembr_cyto' Ste11(SAM!2,Localiz~membr,Sho1,Pbs2,Cla4,Ste20),Ste50
    (RA,Localiz~membr,SAMoligo!2,SAM,Sho1) <-> Ste11(SAM!2,Localiz~cyto,Sho1,
    Pbs2,Cla4,Ste20),Ste50(RA,Localiz~cyto,SAMoligo!2,SAM,Sho1) @ 10,1
'Ste50membr_cyto' Ste50(RA,Localiz~membr,SAMoligo,SAM,Sho1) <-> Ste50(RA,Localiz
    ~cyto,SAMoligo,SAM,Sho1) @ 10,1


# Initial Conditions:
%init: 0 * (Osm(Localiz~e,r))
%init: 1160 * (Opy2(CR-A,Localiz~membr,CR-D,CR-B~p,CR-C,Yck))
%init: 521 * (Nbp2(Ptc1,Pbs2,Localiz~cyto))
%init: 0 * (mRNA_mCherry(Localiz~nuc))
%init: 0 * (mRNA_Venus(Localiz~nuc))
%init: 10 * (Remodeler(PolII))
%init: 200000 * (Trigger())
%init: 907 * (Fps1(Localiz~membr,Osm,A~a,Hog1))
%init: 1630 * (Yck1_2(Opy2,A~a))
%init: 18000 * (Ptc23(Hog1,P~ia,Localiz~cyto))
%init: 5000 * (GlycProd(Hog1,X~ia))
%init: 768 * (Ptp(mapk,Localiz~cyto,Hog1,P~a))
%init: 1520 * (Ptc1(Localiz~cyto,KIM,Nbp2,Hog,P~a))
%init: 150 * (GlycFeedback(Hog1,X~ia,GlycProd))
%init: 1000 * (Ssk22(98_179,Thr1460~u,Localiz~cyto,pbs2,phosphatas))
%init: 217 * (Ssk2(294_413,Thr1460~u,Localiz~cyto,pbs2,Ssk,phosphatas))
%init: 100 * (FeedbackDummy(Hog1pp,Ptc23,X~ia))
%init: 336 * (Ste11(Sho1,Pbs2,Cla4,SAM,Ste20,Ser281Ser285Thr286~u,Localiz~membr)
    )
%init: 149 * (Ptp(Localiz~nuc,mapk,Hog1,P~a))
```

```
%init: 5330 * (Ypd1(Ssk1,Sln1,His64~u,Localiz~membr))
%init: 1000 * (Ypd1(Localiz~membr,Sln1,Ssk1,His64~p))
%init: 1200 * (Ssk1(Asp554~p,Ypd1,Dimer,Localiz~membr,Ssk2_22))
%init: 10 * (gene(TF,PolII,Remodel))
%init: 1 * (Venus(TF,PolII,Remodel))
%init: 1 * (mCherry(TF,PolII,Remodel))
%init: 500 * (Sko1(Hog1pp))
%init: 6490 * (Bem1(CI,PB1,Ste5,SH3b,Localiz~membr))
%init: 150 * (TF(Hopg1pp,Gene))
%init: 2000 * (Hkr1(STR,HMH,Sho1,X~ia,OsmIntern,Localiz~membr))
%init: 1360 * (Msb2(STR,HMH,Cyt_Cdc42,Sho1,OsmIntern,Localiz~membr,X~ia))
%init: 2160 * (Pbs2(HBD-I,Ser514~u,Thr518~u,ssk44_57,sho91_102,Localiz~cyto,
    Ste11,Nbp2,Ptc23))
%init: 200 * (PolII(Remodel,Gene))
%init: 900 * (Cdc42(GEF,ToCRIB,ste50,Msb2,Bem1,Localiz~membr,X~GDP))
%init: 656 * (Sln1_RecD(Asp1144~p,h!1,Ypd1),Sln1_SensD(OsmIntern,Osm,X~a,K!2,
    Localiz~membr),Sln1_HkD(r!1,DimR,His576~p,s!2))
%init: 2330 * (Sho1(cdc24,Mucin,SH3_342_346,SH3_338,Ste50,Localiz~membr,Mucin2,
    Hog1~u,X~ia))
%init: 259 * (Ste20(CRIB,Localiz~cyto,Ste11,PRR,X~ia))
%init: 259 * (Cla4(CRIB,Ste11,X~ia,Localiz~cyto,PRR))
%init: 6780 * (Hog1(Thr174~u,Tyr176~u,Localiz~cyto,TF,Ste50,Phosphat,Ptc23,
    Kinase,Sho1,CD~pbd1))
%init: 1010 * (Cdc24(sho1,DH,Localiz~cyto,PhosphSites~u,PB1,PH,X~ia))
%init: 2000 * (Ptc23(Hog1,P~ia,Localiz~nuc))
%init: 200 * (Ste50(S155_196_202_248~u,Sho1,SAMoligo!2,SAM!1,RA,Localiz~cyto,
    Hog1),Ste11(Sho1,Pbs2,Cla4,SAM!1,Ste20,Ser281Ser285Thr286~u,Localiz~cyto),
    Ste11(Sho1,Pbs2,Cla4,SAM!2,Ste20,Ser281Ser285Thr286~u,Localiz~cyto))
%init: 590 * (Ste50(S155_196_202_248~u,Sho1,SAMoligo,SAM,RA,Localiz~cyto,Hog1))
%init: 110 * (Ste50(Localiz~cyto,RA,S155_196_202_248~u,SAM!1,SAMoligo!0,Sho1,
    Hog1),Ste50(Localiz~cyto,RA,S155_196_202_248~u,SAM!1,SAMoligo!2,Sho1,Hog1),
    Ste50(Localiz~cyto,RA,S155_196_202_248~u,SAM,SAMoligo!2,Sho1,Hog1),Ste50(
    Localiz~cyto,RA,S155_196_202_248~u,SAM!0,SAMoligo,Sho1,Hog1))
%init: 110 * (Ste50(Localiz~cyto,RA,S155_196_202_248~u,SAM!0,SAMoligo,Sho1,Hog1)
    ,Ste50(Localiz~cyto,RA,S155_196_202_248~u,SAM!1,SAMoligo!0,Sho1,Hog1),Ste50(
    Localiz~cyto,RA,S155_196_202_248~u,SAM!1,SAMoligo!2,Sho1,Hog1),Ste50(Localiz
    ~cyto,RA,S155_196_202_248~u,SAM!2,SAMoligo,Sho1,Hog1))
%init: 0 * (mRNA_Gene(Localiz~nuc))

# Simulation:
%obs: Hkr1(X~a)
%obs: Msb2(X~a,Localiz~membr)
```

```
%obs: Cdc24(X~a)
%obs: Cdc42(X~GTP)
%obs: Ste20(X~a)
%obs: Cla4(X~a)
%obs: Ste11(Ser281Ser285Thr286~ppp)
%obs: Pbs2(Thr518~p,Ser514~p)
%obs: Ptc1(P~a,Localiz~cyto)
%obs: GlycFeedback(X~a)
%obs: Ptc23(P~a,Localiz~cyto)
%obs: Ptc23(P~a,Localiz~nuc)
%obs: Ptp(P~a)
%obs: Hog1(Thr174~p,Tyr176~p,Localiz~cyto)
%obs: Hog1(Thr174~p,Tyr176~p,Localiz~nuc)
%obs: Sln1_SensD(X~ia,Localiz~membr)
%obs: Sln1_HkD(His576~p)
%obs: Sln1_RecD(Asp1144~p)
%obs: Ypd1(His64~u)
%obs: Ssk1(Asp554~u,Dimer!1),Ssk1(Asp554~u,Dimer!1)
%obs: Ssk2(Thr1460~u)
%obs: Ssk22(Thr1460~u)
%obs: mRNA_Gene()
%obs: mRNA_mCherry()
%obs: mRNA_Venus()
%obs: Sho1(X~a,Localiz~membr)
%obs: Ptc1(KIM,P~a,Nbp2!1,Localiz~cyto),Nbp2(Pbs2!2,Localiz~cyto,Ptc1!1),Pbs2(
    Nbp2!2,Localiz~cyto)
%obs: FeedbackDummy(X~a)
%obs: Trigger()
%obs: Osm(Localiz~e)
%obs: Osm(r,Localiz~i)
%obs: Ste50(SAMoligo!1),Ste11(SAM!1)
%obs: Ste11(SAM!1),Ste50(SAM!1)
%obs: Ptc1(P~a,Localiz~nuc)
%obs: GlycProd(X~a)
%obs: Ste50(SAMoligo!2,SAM!1),Ste11(SAM!1),Ste11(SAM!2)
%obs: Ste11(Sho1,Pbs2,Cla4,SAM!1,Ste20),Ste50(Sho1,Hog1,SAMoligo,SAM!1,RA)
%obs: Ste50(Sho1,Hog1,SAMoligo!1,SAM,RA),Ste11(Sho1,Pbs2,Cla4,SAM!1,Ste20)
%obs: Ste50(Sho1,Hog1,SAMoligo!2,SAM!1,RA),Ste11(Sho1,Pbs2,Cla4,SAM!1,Ste20),
    Ste11(Sho1,Pbs2,Cla4,SAM!2,Ste20)
%obs: Ste11(Sho1,Pbs2,Cla4,SAM!1,Ste20,Ser281Ser285Thr286),Ste50(
    S155_196_202_248,Sho1,Hog1,SAMoligo,SAM!1,RA)
%obs: Ste50(S155_196_202_248,Sho1,Hog1,SAMoligo!2,SAM!1,RA),Ste11(Sho1,Pbs2,Cla4
```

```
    ,SAM!1,Ste20,Ser281Ser285Thr286),Ste11(Sho1,Pbs2,Cla4,SAM!2,Ste20,
    Ser281Ser285Thr286)
%obs: Ste50(S155_196_202_248,SAMoligo!2,SAM!1),Ste11(SAM!1,Ser281Ser285Thr286),
    Ste11(SAM!2,Ser281Ser285Thr286)
%obs: Fps1(A~ia,Localiz~membr)
%obs: mCherry(Remodel!4,PolII,TF),Remodeler(PolII!4)
%obs: Venus(TF,Remodel!1,PolII),Remodeler(PolII!1)
%obs: Hog1(TF!1,Thr174~p,Tyr176~p,Localiz~nuc),TF(Hopg1pp!1,Gene)
%obs: Hog1(TF!1,Thr174~p,Tyr176~p,Localiz~nuc),Sko1(Hog1pp!1)
%obs: Hog1(TF!1,Thr174~p,Tyr176~p,Localiz~nuc),TF(Gene!2,Hopg1pp!1),mCherry(
    Remodel!4,PolII!3,TF!2),PolII(Remodel,Gene!3),Remodeler(PolII!4)
%obs: Hog1(TF!1,Thr174~p,Tyr176~p,Localiz~nuc),TF(Gene!4,Hopg1pp!1),Venus(
    Remodel!2,PolII!3,TF!4),PolII(Remodel,Gene!3),Remodeler(PolII!2)
%obs: Hog1(TF!1,Thr174~p,Tyr176~p,Localiz~nuc),TF(Gene!4,Hopg1pp!1),gene(PolII
    !2,TF!4,Remodel!3),PolII(Remodel,Gene!2),Remodeler(PolII!3)
%obs: Ste50(Localiz~membr)
%obs: Ste50(Sho1,Hog1,RA,Localiz~membr)
%obs: Ste11(Localiz~membr)
%obs: Ste11(Sho1,Pbs2,Cla4,Ste20,Ser281Ser285Thr286~ppp,Localiz~membr)
%obs: Ste50(Sho1,Hog1,SAMoligo,SAM,RA,Localiz~membr)
%obs: Ste11(Sho1,Pbs2,Cla4,SAM,Ste20,Localiz~membr)

# Perturbation:
%mod: $T>800 do 'TriggerOsm':=$INF
```

A.8 The High-Osmolarity Glycerol Web in Yeast (Spatial)

```
# John Wilson-Kanamori and Peter Krenn
#
# Model of the HOG pathway in Yeast - last update 11-07-2014 (based on version
    02-2013)
#
# Based on work by Peter Krenn (peter.krenn@edu.uni-graz.at).
#      Assembly and experimental validation of a rule based model for the high
    osmolarity glycerol (HOG) pathway in yeast.
# Revised and transcribed by John Wilson-Kanamori (j.r.wilson-kanamori@sms.ed.ac
    .uk).


### Spatial Structures:
%compartment: Membrane
%compartment: Cytosol
%compartment: Nucleus
%channel: Memb_Cyto :Membrane -> :Cytosol
%channel: Cyto_Memb :Cytosol -> :Membrane
%channel: Cyto_Nuc :Cytosol -> :Nucleus
%channel: Nuc_Cyto :Nucleus -> :Cytosol


### Agents:
%agent: Bem1(CI,PB1,Ste5,SH3b)
%agent: Cdc24(Sho1,DH,PB1,PH,X~ia~a,PhosphSites~u~p)
%agent: Cdc42(GEF,ToCRIB,Ste50,Msb2,Bem1,X~GDP~GTP)
%agent: Cla4(PRR,Ste11,CRIB,X~ia~a)
%agent: FeedbackDummy(Hog1pp,Ptc23,X~ia~a)
%agent: Fps1(Osm,Hog1,A~ia~a)
%agent: gene(TF,PolII,Remodel)
%agent: GlycFeedback(Hog1,GlycProd,X~ia~a)
%agent: GlycProd(Hog1,X~ia~a)
%agent: Hkr1(Sho1,STR,HMH,OsmIntern,X~ia~a)
%agent: Hog1(Ste50,Sho1,Ptc23,CD~pbd1~pbd2,TF,Phosphat,Kinase,Thr174~u~p,Tyr176~
    u~p)
%agent: mCherry(TF,PolII,Remodel)
%agent: mRNA_Gene()
%agent: mRNA_mCherry()
%agent: mRNA_Venus()
%agent: Msb2(Cyt_Cdc42,Sho1,STR,HMH,OsmIntern,X~ia~a)
```

```
%agent: Nbp2(Ptc1,Pbs2)
%agent: Opy2(CR-A,CR-B~u~p,CR-C,CR-D,Yck)
%agent: Osm(Localiz~e~i,r)
%agent: Pbs2(ssk44_57,sho91_102,Ptc23,Nbp2,Ste11,HBD-I,Ser514~u~p,Thr518~u~p)
%agent: PolII(Gene,Remodel)
%agent: Ptc1(Hog1,Nbp2,KIM,P~ia~a)
%agent: Ptc23(Hog1,P~ia~a)
%agent: Ptp(Hog1,mapk,P~ia~a)
%agent: Remodeler(PolII)
%agent: Sho1(Cdc24,Ste50,SH3_338,SH3_342_346,Mucin,Mucin2,Hog1~u~p,X~a~ia)
%agent: Sko1(Hog1pp)
%agent: Sln1_HkD(DimR,s,r,His576~u~p)
%agent: Sln1_RecD(Ypd1,h,Asp1144~u~p)
%agent: Sln1_SensD(Osm,OsmIntern,k,X~ia~a)
%agent: Ssk1(Dimer,Ypd1,Ssk2_22,Asp554~u~p)
%agent: Ssk2(Pbs2,Ssk,phosphatas,s294_413,Thr1460~u~p)
%agent: Ssk22(Pbs2,phosphatas,s98_179,Thr1460~u~p)
%agent: Ste11(Pbs2,Cla4,Ste20,Sho1,SAM,Ser281Ser285Thr286~u~p~pp~ppp)
%agent: Ste20(PRR,Ste11,CRIB,X~ia~a)
%agent: Ste50(Sho1,RA,SAM,SAMoligo,Hog1,S155_196_202_248~u~p~pp~ppp~pppp)
%agent: TF(Gene,Hog1pp)
%agent: Trigger()
%agent: Venus(TF,PolII,Remodel)
%agent: Yck1_2(Opy2,A~a)
%agent: Ypd1(Sln1,Ssk1,His64~u~p)


### Rules:

# MAPK System
'Ssk1Dimer_AutPhosph_Ssk2' :Cytosol Ssk1(Dimer!1,Asp554~u),Ssk2(s294_413!2,
    Thr1460~u),Ssk1(Dimer!1,Ssk2_22!2,Asp554~u) -> :Cytosol Ssk1(Dimer!1,Asp554~
    u),Ssk2(s294_413!2,Thr1460~p),Ssk1(Dimer!1,Ssk2_22!2,Asp554~u) @ 10
'Ptc1aNbp2_Dephos_Ssk2' :Cytosol Ptc1(P~a,Nbp2!3),Nbp2(Ptc1!3,Pbs2!2),Pbs2(Nbp2
    !2,ssk44_57!1),Ssk2(Pbs2!1,Thr1460~p) -> :Cytosol Ptc1(P~a,Nbp2!2),Nbp2(Ptc1
    !2,Pbs2!3),Pbs2(Nbp2!3,ssk44_57!1),Ssk2(Pbs2!1,Thr1460~u) @ 5
'Ssk2cyto_Binds_Pbs2cyto' :Cytosol Ssk2(Pbs2),Pbs2(ssk44_57) -> :Cytosol Ssk2(
    Pbs2!1),Pbs2(ssk44_57!1) @ 0.00005
'Ssk2cyto_Unbinds_Pbs2cyto' :Cytosol Ssk2(Pbs2!1),Pbs2(ssk44_57!1) -> :Cytosol
    Ssk2(Pbs2),Pbs2(ssk44_57) @ 0.1
'Ssk2_Phosph_Pbs2Thr' Ssk2(Pbs2!1,Thr1460~p),Pbs2(ssk44_57!1,Thr518~u) -> Ssk2(
    Pbs2!1,Thr1460~p),Pbs2(ssk44_57!1,Thr518~p) @ 10
```

```
'Ssk2_Phosph_Pbs2Ser' Ssk2(Pbs2!1,Thr1460~p),Pbs2(ssk44_57!1,Ser514~u) -> Ssk2(
    Pbs2!1,Thr1460~p),Pbs2(ssk44_57!1,Ser514~p) @ 10
'Ssk1Dimer_AutPhosph_Ssk22cyt' :Cytosol Ssk1(Asp554~u,Dimer!1),Ssk1(Asp554~u,
    Dimer!1,Ssk2_22!2),Ssk22(s98_179!2,Thr1460~u) -> :Cytosol Ssk1(Asp554~u,
    Dimer!1),Ssk1(Asp554~u,Dimer!1,Ssk2_22!2),Ssk22(s98_179!2,Thr1460~p) @ 10
'Ssk22cyto_Binds_Pbs2cyto' :Cytosol Ssk22(Pbs2),Pbs2(ssk44_57) -> :Cytosol Ssk22
    (Pbs2!1),Pbs2(ssk44_57!1) @ 0.00005
'Ssk22cyto_Unbinds_Pbs2cyto' :Cytosol Ssk22(Pbs2!1),Pbs2(ssk44_57!1) -> :Cytosol
     Ssk22(Pbs2),Pbs2(ssk44_57) @ 0.1
'Ssk22_Phosph_Pbs2Thr' Ssk22(Pbs2!1,Thr1460~p),Pbs2(ssk44_57!1,Thr518~u) ->
    Ssk22(Pbs2!1,Thr1460~p),Pbs2(ssk44_57!1,Thr518~p) @ 10
'Ssk22_Phosph_Pbs2Ser' Ssk22(Pbs2!1,Thr1460~p),Pbs2(ssk44_57!1,Ser514~u) ->
    Ssk22(Pbs2!1,Thr1460~p),Pbs2(ssk44_57!1,Ser514~p) @ 10
'Ptc1aNbp2_Dephos_Ssk22' :Cytosol Nbp2(Ptc1!3,Pbs2!2),Ptc1(P~a,Nbp2!3),Pbs2(Nbp2
    !2,ssk44_57!1),Ssk22(Pbs2!1,Thr1460~p) -> :Cytosol Nbp2(Ptc1!2,Pbs2!3),Ptc1(
    P~a,Nbp2!2),Pbs2(Nbp2!3,ssk44_57!1),Ssk22(Pbs2!1,Thr1460~u) @ 5
'Pbs2pp_Unbind_Sho1' Sho1(SH3_338!1,X),Pbs2(sho91_102!1,Thr518~p,Ser514~p) ->
    Sho1(SH3_338,X),Pbs2(sho91_102,Thr518~p,Ser514~p) @ 10
'Ste11membr_Unbinds_Pbs2membr' :Membrane Pbs2(Ste11!1),Ste11(Pbs2!1) -> :
    Membrane Pbs2(Ste11),Ste11(Pbs2) @ 0.1
'Ste11membr_Binds_Pbs2Umembr Ref1' :Membrane Ste11(Ser281Ser285Thr286~ppp,Pbs2),
    Pbs2(Ste11,sho91_102,Thr518~u,Ser514~u) -> :Membrane Ste11(
    Ser281Ser285Thr286~ppp,Pbs2!1),Pbs2(Ste11!1,sho91_102,Thr518~u,Ser514~u) @
    0.00000005
'Ste11membr_Unbinds_Pbs2Umembr Ref1' :Membrane Ste11(Ser281Ser285Thr286~ppp,Pbs2
    !1),Pbs2(Ste11!1,sho91_102,Thr518~u,Ser514~u) -> :Membrane Ste11(
    Ser281Ser285Thr286~ppp,Pbs2),Pbs2(Ste11,sho91_102,Thr518~u,Ser514~u) @ 0.1
'Ste11membr_Binds_Pbs2Umembr Ref2' :Membrane Pbs2(Ste11,Ser514~p,Thr518~u,
    sho91_102),Ste11(Pbs2,Ser281Ser285Thr286~ppp) -> :Membrane Pbs2(Ste11!1,
    Ser514~p,Thr518~u,sho91_102),Ste11(Pbs2!1,Ser281Ser285Thr286~ppp) @ 0.000005
'Ste11membr_Unbinds_Pbs2Umembr Ref2' :Membrane Pbs2(Ste11!1,Ser514~p,Thr518~u,
    sho91_102),Ste11(Pbs2!1,Ser281Ser285Thr286~ppp) -> :Membrane Pbs2(Ste11,
    Ser514~p,Thr518~u,sho91_102),Ste11(Pbs2,Ser281Ser285Thr286~ppp) @ 0.1
'Ste11membr_Binds_Pbs2Umembr Ref3' :Membrane Pbs2(Ste11,Thr518~p,Ser514~u,
    sho91_102),Ste11(Pbs2,Ser281Ser285Thr286~ppp) -> :Membrane Pbs2(Ste11!1,
    Thr518~p,Ser514~u,sho91_102),Ste11(Pbs2!1,Ser281Ser285Thr286~ppp) @ 0.000005
'Ste11membr_Unbinds_Pbs2Umembr Ref3' :Membrane Pbs2(Ste11!1,Thr518~p,Ser514~u,
    sho91_102),Ste11(Pbs2!1,Ser281Ser285Thr286~ppp) -> :Membrane Pbs2(Ste11,
    Thr518~p,Ser514~u,sho91_102),Ste11(Pbs2,Ser281Ser285Thr286~ppp) @ 0.1
'Ste11membr_Phosph_Pbs2membrThr' :Membrane Pbs2(Ste11!1,Thr518~u),Ste11(
    Ser281Ser285Thr286~ppp,Pbs2!1) -> :Membrane Pbs2(Ste11!1,Thr518~p),Ste11(
    Ser281Ser285Thr286~ppp,Pbs2!1) @ 10
```

```
'Ste11membr_Phosph_Pbs2membrSer' :Membrane Pbs2(Ste11!1,Ser514~u),Ste11(
    Ser281Ser285Thr286~ppp,Pbs2!1) -> :Membrane Pbs2(Ste11!1,Ser514~p),Ste11(
    Ser281Ser285Thr286~ppp,Pbs2!1) @ 10
'Ste11pppcyto_Unbinds_Pbs2cyto' :Cytosol Pbs2(Ste11!1),Ste11(Pbs2!1) -> :Cytosol
     Pbs2(Ste11),Ste11(Pbs2) @ 0.1
'Ste11pppcyto_Binds_Pbs2cyto Ref1' :Cytosol Pbs2(Ste11,sho91_102),Ste11(
    Ser281Ser285Thr286~ppp,Pbs2) -> :Cytosol Pbs2(Ste11!1,sho91_102),Ste11(
    Ser281Ser285Thr286~ppp,Pbs2!1) @ 0.00000005
'Ste11pppcyto_Unbinds_Pbs2cyto Ref1' :Cytosol Pbs2(Ste11!1,sho91_102),Ste11(
    Ser281Ser285Thr286~ppp,Pbs2!1) -> :Cytosol Pbs2(Ste11,sho91_102),Ste11(
    Ser281Ser285Thr286~ppp,Pbs2) @ 0.1
'Ste11pppcyto_Binds_Pbs2cyto Ref2' :Cytosol Ste11(Pbs2,Ser281Ser285Thr286~ppp),
    Pbs2(Ste11,Thr518~u,Ser514~p,sho91_102) -> :Cytosol Ste11(Pbs2!1,
    Ser281Ser285Thr286~ppp),Pbs2(Ste11!1,Thr518~u,Ser514~p,sho91_102) @ 0.000005
'Ste11pppcyto_Unbinds_Pbs2cyto Ref2' :Cytosol Ste11(Pbs2!1,Ser281Ser285Thr286~
    ppp),Pbs2(Ste11!1,Thr518~u,Ser514~p,sho91_102) -> :Cytosol Ste11(Pbs2,
    Ser281Ser285Thr286~ppp),Pbs2(Ste11,Thr518~u,Ser514~p,sho91_102) @ 0.1
'Ste11pppcyto_Binds_Pbs2cyto Ref3' :Cytosol Ste11(Pbs2,Ser281Ser285Thr286~ppp),
    Pbs2(Ste11,Thr518~p,Ser514~u,sho91_102) -> :Cytosol Ste11(Pbs2!1,
    Ser281Ser285Thr286~ppp),Pbs2(Ste11!1,Thr518~p,Ser514~u,sho91_102) @ 0.000005
'Ste11pppcyto_Unbinds_Pbs2cyto Ref3' :Cytosol Ste11(Pbs2!1,Ser281Ser285Thr286~
    ppp),Pbs2(Ste11!1,Thr518~p,Ser514~u,sho91_102) -> :Cytosol Ste11(Pbs2,
    Ser281Ser285Thr286~ppp),Pbs2(Ste11,Thr518~p,Ser514~u,sho91_102) @ 0.1
'Ste11cyto_Phosph_Pbs2cytoSer' :Cytosol Pbs2(Ste11!1,Ser514~u),Ste11(
    Ser281Ser285Thr286~ppp,Pbs2!1) -> :Cytosol Pbs2(Ste11!1,Ser514~p),Ste11(
    Ser281Ser285Thr286~ppp,Pbs2!1) @ 10
'Ste11cyto_Phosph_Pbs2cytoThr' :Cytosol Pbs2(Ste11!1,Thr518~u),Ste11(Pbs2!1,
    Ser281Ser285Thr286~ppp) -> :Cytosol Pbs2(Ste11!1,Thr518~p),Ste11(Pbs2!1,
    Ser281Ser285Thr286~ppp) @ 10
'Hog1_Binds_Pbs2_1' :Cytosol Pbs2(HBD-I!1,Thr518,Ser514),Hog1(Thr174,Tyr176,CD~
    pbd1!1) -> :Cytosol Pbs2(HBD-I,Thr518,Ser514),Hog1(Thr174,Tyr176,CD~pbd1) @
    0.5
'Hog1_Binds_Pbs2_1 Ref2' :Cytosol Hog1(Thr174~u,Tyr176~u,CD~pbd1),Pbs2(Ser514~p,
    Thr518~p,HBD-I) -> :Cytosol Hog1(Thr174~u,Tyr176~u,CD~pbd1!1),Pbs2(Ser514~p,
    Thr518~p,HBD-I!1) @ 0.000005
'Hog1_Unbinds_Pbs2_1 Ref2' :Cytosol Hog1(Thr174~u,Tyr176~u,CD~pbd1!1),Pbs2(
    Ser514~p,Thr518~p,HBD-I!1) -> :Cytosol Hog1(Thr174~u,Tyr176~u,CD~pbd1),Pbs2(
    Ser514~p,Thr518~p,HBD-I) @ 0.5
'Hog1_Binds_Pbs2_1 Ref3' :Cytosol Hog1(Thr174~p,Tyr176~u,CD~pbd1),Pbs2(HBD-I,
    Ser514~p,Thr518~p) -> :Cytosol Hog1(Thr174~p,Tyr176~u,CD~pbd1!1),Pbs2(HBD-I
    !1,Ser514~p,Thr518~p) @ 0.00005
'Hog1_Unbinds_Pbs2_1 Ref3' :Cytosol Hog1(Thr174~p,Tyr176~u,CD~pbd1!1),Pbs2(HBD-I
```

```
    !1,Ser514~p,Thr518~p) -> :Cytosol Hog1(Thr174~p,Tyr176~u,CD~pbd1),Pbs2(HBD-I
    ,Ser514~p,Thr518~p) @ 0.5
'Hog1_Binds_Pbs2_1 Ref4' :Cytosol Hog1(Thr174~u,Tyr176~p,CD~pbd1),Pbs2(HBD-I,
    Ser514~p,Thr518~p) -> :Cytosol Hog1(Thr174~u,Tyr176~p,CD~pbd1!1),Pbs2(HBD-I
    !1,Ser514~p,Thr518~p) @ 0.00005
'Hog1_Unbinds_Pbs2_1 Ref4' :Cytosol Hog1(Thr174~u,Tyr176~p,CD~pbd1!1),Pbs2(HBD-I
    !1,Ser514~p,Thr518~p) -> :Cytosol Hog1(Thr174~u,Tyr176~p,CD~pbd1),Pbs2(HBD-I
    ,Ser514~p,Thr518~p) @ 0.5
'Hog1_Binds_Pbs2_2' :Cytosol Pbs2(HBD-I!1,Thr518,Ser514),Hog1(Thr174,Tyr176,CD~
    pbd1!1) -> :Cytosol Pbs2(HBD-I!1,Thr518,Ser514),Hog1(Thr174,Tyr176,CD~pbd2
    !1) @ 20
'Hog1_Unbind_Pbs2_pbd2' :Cytosol Pbs2(HBD-I!1,Thr518,Ser514),Hog1(Thr174,Tyr176,
    CD~pbd2!1) -> :Cytosol Pbs2(HBD-I,Thr518,Ser514),Hog1(Thr174,Tyr176,CD~pbd1)
     @ 0.1
'Hog1cyto_Binds_Pbs2_Direct' :Cytosol Pbs2(HBD-I!1),Hog1(CD~pbd2!1) -> :Cytosol
    Pbs2(HBD-I),Hog1(CD~pbd1) @ 0.1
'Hog1cyto_Binds_Pbs2_Direct Ref2' :Cytosol Pbs2(HBD-I,Thr518~p,Ser514~p),Hog1(
    Tyr176~u,Thr174~u,CD~pbd1) -> :Cytosol Pbs2(HBD-I!1,Thr518~p,Ser514~p),Hog1(
    Tyr176~u,Thr174~u,CD~pbd2!1) @ 0.0000005
'Hog1cyto_Unbinds_Pbs2_Direct Ref2' :Cytosol Pbs2(HBD-I!1,Thr518~p,Ser514~p),
    Hog1(Tyr176~u,Thr174~u,CD~pbd2!1) -> :Cytosol Pbs2(HBD-I,Thr518~p,Ser514~p),
    Hog1(Tyr176~u,Thr174~u,CD~pbd1) @ 0.1
'Hog1cyto_Binds_Pbs2_Direct Ref3' :Cytosol Hog1(Thr174~u,Tyr176~p,CD~pbd1),Pbs2(
    HBD-I,Thr518~p,Ser514~p) -> :Cytosol Hog1(Thr174~u,Tyr176~p,CD~pbd2!1),Pbs2(
    HBD-I!1,Thr518~p,Ser514~p) @ 0.00005
'Hog1cyto_Unbinds_Pbs2_Direct Ref3' :Cytosol Hog1(Thr174~u,Tyr176~p,CD~pbd2!1),
    Pbs2(HBD-I!1,Thr518~p,Ser514~p) -> :Cytosol Hog1(Thr174~u,Tyr176~p,CD~pbd1),
    Pbs2(HBD-I,Thr518~p,Ser514~p) @ 0.1
'Hog1cyto_Binds_Pbs2_Direct Ref4' :Cytosol Hog1(Thr174~p,Tyr176~u,CD~pbd1),Pbs2(
    HBD-I,Thr518~p,Ser514~p) -> :Cytosol Hog1(Thr174~p,Tyr176~u,CD~pbd2!1),Pbs2(
    HBD-I!1,Thr518~p,Ser514~p) @ 0.00005
'Hog1cyto_Unbinds_Pbs2_Direct Ref4' :Cytosol Hog1(Thr174~p,Tyr176~u,CD~pbd2!1),
    Pbs2(HBD-I!1,Thr518~p,Ser514~p) -> :Cytosol Hog1(Thr174~p,Tyr176~u,CD~pbd1),
    Pbs2(HBD-I,Thr518~p,Ser514~p) @ 0.1
'Pbs2_Phosph_Hog1Thr' :Cytosol Pbs2(HBD-I!1,Ser514~p,Thr518~p),Hog1(Thr174~u,CD~
    pbd2!1) -> :Cytosol Pbs2(HBD-I!1,Ser514~p,Thr518~p),Hog1(Thr174~p,CD~pbd2!1)
     @ 5
'Pbs2_Phosph_Hog1Tyr' :Cytosol Pbs2(HBD-I!1,Ser514~p,Thr518~p),Hog1(Tyr176~u,CD~
    pbd2!1) -> :Cytosol Pbs2(HBD-I!1,Ser514~p,Thr518~p),Hog1(Tyr176~p,CD~pbd2!1)
     @ 10
'Pbs2_Phosph_Hog1Thr_Pbd1' :Cytosol Pbs2(HBD-I!1,Ser514~p,Thr518~p),Hog1(Thr174~
    u,CD~pbd1!1) -> :Cytosol Pbs2(HBD-I!1,Ser514~p,Thr518~p),Hog1(Thr174~p,CD~
```

```
    pbd1!1) @ 0.05
'Pbs2_Phosph_Hog1Tyr_Pbd1' :Cytosol Pbs2(HBD-I!1,Ser514~p,Thr518~p),Hog1(Tyr176~
    u,CD~pbd1!1) -> :Cytosol Pbs2(HBD-I!1,Ser514~p,Thr518~p),Hog1(Tyr176~p,CD~
    pbd1!1) @ 0.1
'Ptc1_Binds_Npb2' Ptc1(Nbp2,P),Nbp2(Ptc1) -> Ptc1(Nbp2!1,P),Nbp2(Ptc1!1) @
    0.0001
'Ptc1_Unbinds_Npb2' Ptc1(Nbp2!1,P),Nbp2(Ptc1!1) -> Ptc1(Nbp2!1,P),Nbp2(Ptc1!1) @
     0.00001
'Nbp2_Unbinds_Pbs2' :Cytosol Pbs2(Nbp2!1),Nbp2(Pbs2!1) -> :Cytosol Pbs2(Nbp2),
    Nbp2(Pbs2) @ 1.0
'Nbp2_Binds_Pbs2pp Ref1' :Cytosol Nbp2(Pbs2),Pbs2(Nbp2,Ser514~p,Thr518~p) -> :
    Cytosol Nbp2(Pbs2!1),Pbs2(Nbp2!1,Ser514~p,Thr518~p) @ 0.0001
'Nbp2_Unbinds_Pbs2pp Ref1' :Cytosol Nbp2(Pbs2!1),Pbs2(Nbp2!1,Ser514~p,Thr518~p)
    -> :Cytosol Nbp2(Pbs2),Pbs2(Nbp2,Ser514~p,Thr518~p) @ 1.0
'Ptc1p_Dephosph_Pbs2Ser' :Cytosol Ptc1(P~a,Nbp2!1),Nbp2(Ptc1!1,Pbs2!2),Pbs2(Nbp2
    !2,Ser514~p) -> :Cytosol Ptc1(P~a,Nbp2!1),Nbp2(Ptc1!1,Pbs2!2),Pbs2(Nbp2!2,
    Ser514~u) @ 5
'Ptc1p_Dephosph_Pbs2Thr' :Cytosol Ptc1(Nbp2!1,P~a),Nbp2(Pbs2!2,Ptc1!1),Pbs2(
    Thr518~p,Nbp2!2) -> :Cytosol Ptc1(Nbp2!1,P~a),Nbp2(Pbs2!2,Ptc1!1),Pbs2(
    Thr518~u,Nbp2!2) @ 5
'Ptc1_Dephos_Hog1Thr_cyto Ref1' :Cytosol Nbp2(Ptc1!1,Pbs2!2),Pbs2(Nbp2!2,HBD-I
    !3),Hog1(Thr174~p,CD!3),Ptc1(Nbp2!1,P~a) -> :Cytosol Nbp2(Ptc1!1,Pbs2!2),
    Pbs2(Nbp2!2,HBD-I!3),Hog1(Thr174~u,CD!3),Ptc1(Nbp2!1,P~a) @ 10
'Ptc1a_Binds_Hog1_cyto' :Cytosol Ptc1(Hog1!1,P~a),Hog1(Thr174,Ptc23!1) -> :
    Cytosol Ptc1(Hog1,P~a),Hog1(Thr174,Ptc23) @ 0.1
'Ptc1a_Binds_Hog1 Ref2' :Cytosol Hog1(Ptc23,Thr174~p),Ptc1(Hog1,P~a,Nbp2) -> :
    Cytosol Hog1(Ptc23!1,Thr174~p),Ptc1(Hog1!1,P~a,Nbp2) @ 0.000005
'Ptc1a_Unbinds_Hog1 Ref2' :Cytosol Hog1(Ptc23!1,Thr174~p),Ptc1(Hog1!1,P~a,Nbp2)
    -> :Cytosol Hog1(Ptc23,Thr174~p),Ptc1(Hog1,P~a,Nbp2) @ 0.1
'Ptc1a_Unbinds_Hog1_nuc' :Nucleus Ptc1(Hog1!1,P),Hog1(Thr174,Ptc23!1) -> :
    Nucleus Ptc1(Hog1,P),Hog1(Thr174,Ptc23) @ 0.1
'Ptc1a_Binds_Hog1_nuc Ref1' :Nucleus Ptc1(Hog1,P~a,Nbp2),Hog1(Ptc23,Thr174~p,TF)
     -> :Nucleus Ptc1(Hog1!1,P~a,Nbp2),Hog1(Ptc23!1,Thr174~p,TF) @ 0.000005
'Ptc1a_Unbinds_Hog1_nuc Ref1' :Nucleus Ptc1(Hog1!1,P~a,Nbp2),Hog1(Ptc23!1,Thr174
    ~p,TF) -> :Nucleus Ptc1(Hog1,P~a,Nbp2),Hog1(Ptc23,Thr174~p,TF) @ 0.1
'Ptc1a_Binds_Hog1_nuc Ref2' :Nucleus Ptc1(Hog1,Nbp2,P~a),Hog1(Ptc23,Thr174~p,TF!
    _) -> :Nucleus Ptc1(Hog1!1,Nbp2,P~a),Hog1(Ptc23!1,Thr174~p,TF!_) @ 0.0000001
'Ptc1a_Unbinds_Hog1_nuc Ref2' :Nucleus Ptc1(Hog1!1,Nbp2,P~a),Hog1(Ptc23!1,Thr174
    ~p,TF!_) -> :Nucleus Ptc1(Hog1,Nbp2,P~a),Hog1(Ptc23,Thr174~p,TF!_) @ 0.1
'Ptc1_DephosDirect_Hog1' :Cytosol Ptc1(Hog1!1,P~a),Hog1(Thr174~p,Ptc23!1) -> :
    Cytosol Ptc1(Hog1!1,P~a),Hog1(Thr174~u,Ptc23!1) @ 1
'Ptc1_DephosDirect_Hog1_nuc' :Nucleus Ptc1(Hog1!1,P~a),Hog1(Thr174~p,Ptc23!1) ->
```

```
     :Nucleus Ptc1(Hog1!1,P~a),Hog1(Thr174~u,Ptc23!1) @ 1
'Hog1p_Activat_Ptc1' Hog1(Thr174~p,Tyr176~p),Ptc1(P~ia) -> Hog1(Thr174~p,Tyr176~
    p),Ptc1(P~a) @ 0.000005
'Hog1pp_Unbinds_FeedbDummy' :Cytosol Hog1(Thr174,Tyr176,Phosphat!1),
    FeedbackDummy(Hog1pp!1) -> :Cytosol Hog1(Thr174,Tyr176,Phosphat),
    FeedbackDummy(Hog1pp) @ 0.1
'Hog1pp_Binds_FeedbDummy Ref1' :Cytosol Hog1(Thr174~p,Tyr176~p,Phosphat),
    FeedbackDummy(Hog1pp,X~ia) -> :Cytosol Hog1(Thr174~p,Tyr176~p,Phosphat!1),
    FeedbackDummy(Hog1pp!1,X~ia) @ 0.00000005
'Hog1pp_Unbinds_FeedbDummy Ref1' :Cytosol Hog1(Thr174~p,Tyr176~p,Phosphat!1),
    FeedbackDummy(Hog1pp!1,X~ia) -> :Cytosol Hog1(Thr174~p,Tyr176~p,Phosphat),
    FeedbackDummy(Hog1pp,X~ia) @ 0.5
'Hog1pp_Activat_FeddbDummy' :Cytosol Hog1(Thr174~p,Tyr176~p,Phosphat!1),
    FeedbackDummy(Hog1pp!1,X~ia) -> :Cytosol Hog1(Thr174~p,Tyr176~p,Phosphat!1),
    FeedbackDummy(Hog1pp!1,X~a) @ 1
'FeedbDummy_Unbinds_Ptc23' FeedbackDummy(Ptc23!1,X),Ptc23(P,Hog1!1) ->
    FeedbackDummy(Ptc23!1,X),Ptc23(P,Hog1!1) @ 0.1
'FeedbDummy_Binds_Ptc23 Ref1' :Cytosol FeedbackDummy(Ptc23,X~a),Ptc23(P~ia,Hog1)
     -> :Cytosol FeedbackDummy(Ptc23!1,X~a),Ptc23(P~ia,Hog1!1) @ 0.00005
'FeedbDummy_Unbinds_Ptc23 Ref1' :Cytosol FeedbackDummy(Ptc23!1,X~a),Ptc23(P~ia,
    Hog1!1) -> :Cytosol FeedbackDummy(Ptc23,X~a),Ptc23(P~ia,Hog1) @ 0.1
'FeedbDummy_Act_Ptc23' :Cytosol FeedbackDummy(Ptc23!1,X~a),Ptc23(P~ia,Hog1!1) ->
     :Cytosol FeedbackDummy(Ptc23!1,X~a),Ptc23(P~a,Hog1!1) @ 10
'FeedbackDummy_Inactivat' FeedbackDummy(X~a) -> FeedbackDummy(X~ia) @ 0.00000001
'Ptc23a_Unbinds_Hog1Cyto' :Cytosol Ptc23(P,Hog1!1),Hog1(Ptc23!1) -> :Cytosol
    Ptc23(P,Hog1),Hog1(Ptc23) @ 0.1
'Ptc23a_Binds_Hog1 Ref1' :Cytosol Ptc23(P~a,Hog1),Hog1(Ptc23,Thr174) -> :Cytosol
     Ptc23(P~a,Hog1!1),Hog1(Ptc23!1,Thr174) @ 0.0000005
'Ptc23a_Unbinds_Hog1 Ref1' :Cytosol Ptc23(P~a,Hog1!1),Hog1(Ptc23!1,Thr174) -> :
    Cytosol Ptc23(P~a,Hog1),Hog1(Ptc23,Thr174) @ 0.1
'Ptc23a_Unbinds_Hog1Nuc' :Nucleus Hog1(Ptc23!1),Ptc23(P,Hog1!1) -> :Nucleus Hog1
    (Ptc23),Ptc23(P,Hog1) @ 0.1
'Ptc23a_Binds_Hog1Nuc Ref1' :Nucleus Ptc23(P~a,Hog1),Hog1(Ptc23,TF,Thr174~p) ->
    :Nucleus Ptc23(P~a,Hog1!1),Hog1(Ptc23!1,TF,Thr174~p) @ 0.00005
'Ptc23a_Unbinds_Hog1Nuc Ref1' :Nucleus Ptc23(P~a,Hog1!1),Hog1(Ptc23!1,TF,Thr174~
    p) -> :Nucleus Ptc23(P~a,Hog1),Hog1(Ptc23,TF,Thr174~p) @ 0.1
'Ptc23a_Binds_Hog1Nuc Ref2' :Nucleus Ptc23(Hog1,P~a),Hog1(Ptc23,TF!_,Thr174~p)
    -> :Nucleus Ptc23(Hog1!1,P~a),Hog1(Ptc23!1,TF!_,Thr174~p) @ 0.000001
'Ptc23a_Unbinds_Hog1Nuc Ref2' :Nucleus Ptc23(Hog1!1,P~a),Hog1(Ptc23!1,TF!_,
    Thr174~p) -> :Nucleus Ptc23(Hog1,P~a),Hog1(Ptc23,TF!_,Thr174~p) @ 0.1
'Ptc23a_Deph_Hog1ThrCyto' :Cytosol Ptc23(P~a,Hog1!1),Hog1(Ptc23!1,Thr174~p) -> :
    Cytosol Ptc23(P~a,Hog1!1),Hog1(Ptc23!1,Thr174~u) @ 10
```

```
'Ptc23a_Desph_Hog1ThrNuc' :Nucleus Ptc23(P~a,Hog1!1),Hog1(Ptc23!1,Thr174~p) -> :
    Nucleus Ptc23(P~a,Hog1!1),Hog1(Ptc23!1,Thr174~u) @ 10
'Ptcs23_Unbinds_Pbs2p' :Cytosol Ptc23(Hog1!1,P),Pbs2(Ptc23!1) -> :Cytosol Ptc23(
    Hog1,P),Pbs2(Ptc23) @ 1.0
'Ptcs23_Binds_Pbs2p Ref1' :Cytosol Ptc23(Hog1,P~a),Pbs2(Ptc23,Ser514~p,Thr518~p)
     -> :Cytosol Ptc23(Hog1!1,P~a),Pbs2(Ptc23!1,Ser514~p,Thr518~p) @ 0.0000005
'Ptcs23_Unbinds_Pbs2p Ref1' :Cytosol Ptc23(Hog1!1,P~a),Pbs2(Ptc23!1,Ser514~p,
    Thr518~p) -> :Cytosol Ptc23(Hog1,P~a),Pbs2(Ptc23,Ser514~p,Thr518~p) @ 1.0
'Ptc23_Dephosph_Pbs2' :Cytosol Ptc23(P~a,Hog1!1),Pbs2(Ptc23!1,Thr518~p,Ser514~p)
     -> :Cytosol Ptc23(P~a,Hog1!1),Pbs2(Ptc23!1,Thr518~u,Ser514~u) @ 1.0
'Ptc23_Unbinds_Ssk2' :Cytosol Ptc23(P,Hog1!1),Ssk2(phosphatas!1) -> :Cytosol
    Ptc23(P,Hog1),Ssk2(phosphatas) @ 1.0
'Ptc23_Binds_Ssk2 Ref1' :Cytosol Ptc23(P~a,Hog1),Ssk2(phosphatas,Thr1460~p) -> :
    Cytosol Ptc23(P~a,Hog1!1),Ssk2(phosphatas!1,Thr1460~p) @ 0.0000005
'Ptc23_Unbinds_Ssk2 Ref1' :Cytosol Ptc23(P~a,Hog1!1),Ssk2(phosphatas!1,Thr1460~p
    ) -> :Cytosol Ptc23(P~a,Hog1),Ssk2(phosphatas,Thr1460~p) @ 1.0
'Ptc23_Dephosph_Ssk2' :Cytosol Ptc23(P~a,Hog1!1),Ssk2(phosphatas!1,Thr1460~p) ->
     :Cytosol Ptc23(P~a,Hog1!1),Ssk2(phosphatas!1,Thr1460~u) @ 1.0
'Ptc23_Unbinds_Ssk22' :Cytosol Ptc23(P,Hog1!1),Ssk22(phosphatas!1) -> :Cytosol
    Ptc23(P,Hog1),Ssk22(phosphatas) @ 1.0
'Ptc23_Binds_Ssk22 Ref1' :Cytosol Ptc23(P~a,Hog1),Ssk22(phosphatas,Thr1460~p) ->
     :Cytosol Ptc23(P~a,Hog1!1),Ssk22(phosphatas!1,Thr1460~p) @ 0.0000005
'Ptc23_Unbinds_Ssk22 Ref1' :Cytosol Ptc23(P~a,Hog1!1),Ssk22(phosphatas!1,Thr1460
    ~p) -> :Cytosol Ptc23(P~a,Hog1),Ssk22(phosphatas,Thr1460~p) @ 1.0
'Ptc23_Dephosph_Ssk22' :Cytosol Ptc23(P~a,Hog1!1),Ssk22(Thr1460~p,phosphatas!1)
    -> :Cytosol Ptc23(P~a,Hog1!1),Ssk22(Thr1460~u,phosphatas!1) @ 1.0
'Ptc23_Deact' Ptc23(P~a) -> Ptc23(P~ia) @ 0.0005
'Hog1ppcyto_Act_Ptpcyto' :Cytosol Ptp(P~ia,mapk),Hog1(Thr174~p,Tyr176~p) -> :
    Cytosol Ptp(P~a,mapk),Hog1(Thr174~p,Tyr176~p) @ 0.000005
'Hog1ppNuc_Act_PtpNuc' :Nucleus Ptp(P~ia,mapk),Hog1(Thr174~p,Tyr176~p) -> :
    Nucleus Ptp(P~a,mapk),Hog1(Thr174~p,Tyr176~p) @ 0.00005
'Ptp_Unbinds_Hog1_cyto' :Cytosol Ptp(mapk!1,P),Hog1(CD~pbd1!1) -> :Cytosol Ptp(
    mapk,P),Hog1(CD~pbd1) @ 0.1
'Ptp_Binds_Hog1_cyto Ref3' :Cytosol Ptp(mapk,P~a),Hog1(Tyr176~p,CD~pbd1) -> :
    Cytosol Ptp(mapk!1,P~a),Hog1(Tyr176~p,CD~pbd1!1) @ 0.000005
'Ptp_Unbinds_Hog1_cyto Ref3' :Cytosol Ptp(mapk!1,P~a),Hog1(Tyr176~p,CD~pbd1!1)
    -> :Cytosol Ptp(mapk,P~a),Hog1(Tyr176~p,CD~pbd1) @ 0.1
'Ptp_Binds_Hog1_cyto_pbd2' :Cytosol Ptp(mapk!1,P~a),Hog1(CD~pbd1!1) -> :Cytosol
    Ptp(mapk!1,P~a),Hog1(CD~pbd2!1) @ 1.0
'Ptp_Unbinds_Hog1_cyto_pbd2' :Cytosol Ptp(mapk!1,P~a),Hog1(CD~pbd2!1) -> :
    Cytosol Ptp(mapk!1,P~a),Hog1(CD~pbd1!1) @ 0.1
'Ptp_Dephosph_Hog1Tyr_cyto' :Cytosol Ptp(mapk!1,P~a),Hog1(Tyr176~p,CD~pbd2!1) ->
```

```
     :Cytosol Ptp(mapk,P~a),Hog1(Tyr176~u,CD~pbd1) @ 10.0
'Ptp_Unbinds_Hog1_nuc' :Nucleus Ptp(mapk!1),Hog1(Tyr176,CD~pbd1!1) -> :Nucleus
    Ptp(mapk),Hog1(Tyr176,CD~pbd1) @ 0.1
'Ptp_Binds_Hog1_nuc Ref1' :Nucleus Ptp(mapk),Hog1(Tyr176~p,TF,CD~pbd1) -> :
    Nucleus Ptp(mapk!1),Hog1(Tyr176~p,TF,CD~pbd1!1) @ 0.000005
'Ptp_Unbinds_Hog1_nuc Ref1' :Nucleus Ptp(mapk!1),Hog1(Tyr176~p,TF,CD~pbd1!1) ->
    :Nucleus Ptp(mapk),Hog1(Tyr176~p,TF,CD~pbd1) @ 1.0
'Ptp_Binds_Hog1_nuc Ref2' :Nucleus Ptp(mapk),Hog1(TF!_,CD~pbd1,Tyr176~p) -> :
    Nucleus Ptp(mapk!1),Hog1(TF!_,CD~pbd1!1,Tyr176~p) @ 0.0000001
'Ptp_Unbinds_Hog1_nuc Ref2' :Nucleus Ptp(mapk!1),Hog1(TF!_,CD~pbd1!1,Tyr176~p)
    -> :Nucleus Ptp(mapk),Hog1(TF!_,CD~pbd1,Tyr176~p) @ 1.0
'Ptp_Binds_Hog1_nuc_pbd2' :Nucleus Ptp(mapk!1),Hog1(Tyr176,CD~pbd1!1) -> :
    Nucleus Ptp(mapk!1),Hog1(Tyr176,CD~pbd2!1) @ 1.0
'Ptp_Unbinds_Hog1_nuc_pbd2' :Nucleus Ptp(mapk!1),Hog1(Tyr176,CD~pbd2!1) -> :
    Nucleus Ptp(mapk!1),Hog1(Tyr176,CD~pbd1!1) @ 0.1
'Ptp_Dephosph_Hog1Tyr_nuc' :Nucleus Ptp(mapk!1),Hog1(Tyr176~p,CD~pbd2!1) -> :
    Nucleus Ptp(mapk),Hog1(Tyr176~u,CD~pbd1) @ 1.0
'Hog1ppcyto_Unbinds_Ste50cyto' :Cytosol Hog1(Thr174,Ste50!1,Tyr176),Ste50(Hog1
    !1) -> :Cytosol Hog1(Thr174,Ste50,Tyr176),Ste50(Hog1) @ 0.5
'Hog1ppcyto_Binds_Ste50cyto Ref2' :Cytosol Hog1(Thr174,Ste50,Tyr176~p),Ste50(
    Hog1) -> :Cytosol Hog1(Thr174,Ste50!1,Tyr176~p),Ste50(Hog1!1) @ 0.000007
'Hog1ppcyto_Unbinds_Ste50cyto Ref2' :Cytosol Hog1(Thr174,Ste50!1,Tyr176~p),Ste50
    (Hog1!1) -> :Cytosol Hog1(Thr174,Ste50,Tyr176~p),Ste50(Hog1) @ 0.5
'Hog1pp_Phosph_Ste50' Hog1(Thr174~p,Ste50!1,Tyr176~p),Ste50(S155_196_202_248~u,
    Hog1!1) -> Hog1(Thr174~p,Ste50!1,Tyr176~p),Ste50(S155_196_202_248~p,Hog1!1)
    @ 10
'Hog1pp_Phosph_Ste50p' Hog1(Thr174~p,Ste50!1,Tyr176~p),Ste50(S155_196_202_248~p,
    Hog1!1) -> Hog1(Thr174~p,Ste50!1,Tyr176~p),Ste50(S155_196_202_248~pp,Hog1!1)
     @ 10
'Hog1pp_Phosph_Ste50pp' Hog1(Thr174~p,Ste50!1,Tyr176~p),Ste50(S155_196_202_248~
    pp,Hog1!1) -> Hog1(Thr174~p,Ste50!1,Tyr176~p),Ste50(S155_196_202_248~ppp,
    Hog1!1) @ 10
'Hog1pp_Phosph_Ste50ppp' Hog1(Thr174~p,Ste50!1,Tyr176~p),Ste50(S155_196_202_248~
    ppp,Hog1!1) -> Hog1(Thr174~p,Ste50!1,Tyr176~p),Ste50(S155_196_202_248~pppp,
    Hog1!1) @ 10
'Ste50pppp_ppp_Membr' :Membrane Ste50(S155_196_202_248~pppp) -> :Membrane Ste50(
    S155_196_202_248~ppp) @ 0.00001
'Ste50ppp_ppu_Membr' :Membrane Ste50(S155_196_202_248~ppp) -> :Membrane Ste50(
    S155_196_202_248~pp) @ 0.00001
'Ste50ppu_puu_Membr' :Membrane Ste50(S155_196_202_248~pp) -> :Membrane Ste50(
    S155_196_202_248~p) @ 0.00001
'Ste50puu_uuu_Membr' :Membrane Ste50(S155_196_202_248~p) -> :Membrane Ste50(
```

```
    S155_196_202_248~u) @ 0.00001
'Ste50pppp_ppp_Cyto' :Cytosol Ste50(S155_196_202_248~pppp) -> :Cytosol Ste50(
    S155_196_202_248~ppp) @ 0.00001
'Ste50ppp_ppu_Cyto' :Cytosol Ste50(S155_196_202_248~ppp) -> :Cytosol Ste50(
    S155_196_202_248~pp) @ 0.00001
'Ste50ppu_puu_Cyto' :Cytosol Ste50(S155_196_202_248~pp) -> :Cytosol Ste50(
    S155_196_202_248~p) @ 0.00001
'Ste50puu_uuu_Cyto' :Cytosol Ste50(S155_196_202_248~p) -> :Cytosol Ste50(
    S155_196_202_248~u) @ 0.00001

# Osmo System
'TriggerOsm' :Membrane Trigger() -> :Membrane Osm(r,Localiz~e) @ 0.0
'OsmEx_Change_OsmInt' Osm(Localiz~e,r) -> Osm(Localiz~i,r) @ 0.0
'OsmEx_Del' Osm(r,Localiz~e) -> @ 0.0
'Sln1SensD_Act' :Membrane Sln1_SensD(X~ia) -> :Membrane Sln1_SensD(X~a) @ 0.0
'Osm_Binds_Sln1SensD' Osm(Localiz~e,r),Sln1_SensD(Osm) -> Osm(Localiz~e,r!1),
    Sln1_SensD(Osm!1) @ 10.0
'Osm_Unbinds_Sln1SensD' Osm(Localiz~e,r!1),Sln1_SensD(Osm!1) -> Osm(Localiz~e,r)
    ,Sln1_SensD(Osm) @ 1.0
'OsmEx_DEactivates_Sln1SensD' Sln1_SensD(Osm!1,X~a),Osm(Localiz~e,r!1) ->
    Sln1_SensD(Osm!1,X~ia),Osm(Localiz~e,r!1) @ 10.0
'OsmIntern_Binds_Sln1SensDE' Osm(Localiz~i,r),Sln1_SensD(Osm,OsmIntern) -> Osm(
    Localiz~i,r!1),Sln1_SensD(Osm!1,OsmIntern) @ 10.0
'OsmIntern_Unbinds_Sln1SensDE' Osm(Localiz~i,r!1),Sln1_SensD(Osm!1,OsmIntern) ->
     Osm(Localiz~i,r),Sln1_SensD(Osm,OsmIntern) @ 0.8
'OsmInt_Activates_Sln1SensDI' Osm(Localiz~i,r!1),Sln1_SensD(X~ia,Osm!1,OsmIntern
    ) -> Osm(Localiz~i,r!1),Sln1_SensD(X~a,Osm!1,OsmIntern) @ 10.0
'OsmEx_Binds_Hkr1' Hkr1(STR),Osm(r,Localiz~e) -> Hkr1(STR!1),Osm(r!1,Localiz~e)
    @ 10.0
'OsmEx_Unbinds_Hkr1' Hkr1(STR!1),Osm(r!1,Localiz~e) -> Hkr1(STR),Osm(r,Localiz~e
    ) @ 1.0
'OsmEx_Activates_HKR1' Osm(r!1,Localiz~e),Hkr1(STR!1,X~ia) -> Osm(r!1,Localiz~e)
    ,Hkr1(STR!1,X~a) @ 2.0
'OsmInt_Binds_Hkr1E' Osm(r,Localiz~i),Hkr1(OsmIntern,STR) -> Osm(r!1,Localiz~i),
    Hkr1(OsmIntern,STR!1) @ 0.05
'OsmInt_Unbinds_Hkr1E' Osm(r!1,Localiz~i),Hkr1(OsmIntern,STR!1) -> Osm(r,Localiz
    ~i),Hkr1(OsmIntern,STR) @ 0.1
'OsmInt_UnbindsCoo_Hkr1E' Hkr1(OsmIntern!1,STR),Osm(r!1,Localiz~i) -> Hkr1(
    OsmIntern,STR),Osm(r,Localiz~i) @ 0.1
'OsmInt_BindsCoo_Hkr1E Ref1' Osm(r!1,Localiz~i),Hkr1(OsmIntern,STR!1),Osm(r,
    Localiz~i) -> Osm(r!1,Localiz~i),Hkr1(OsmIntern!2,STR!1),Osm(r!2,Localiz~i)
    @ 3.0
```

```
'OsmInt_UnbindsCoo_Hkr1E Ref1' Osm(r!1,Localiz~i),Hkr1(OsmIntern!2,STR!1),Osm(r
    !2,Localiz~i) -> Osm(r!1,Localiz~i),Hkr1(OsmIntern,STR!1),Osm(r,Localiz~i) @
     0.1
'OsmIntCoo_Deactivates _HKR1E' Osm(r!1,Localiz~i),Hkr1(X~a,OsmIntern!2,STR!1),
    Osm(r!2,Localiz~i) -> Osm(r!1,Localiz~i),Hkr1(X~ia,OsmIntern!2,STR!1),Osm(r
    !2,Localiz~i) @ 10.0
'OsmEx_Binds_Msb2' Msb2(STR),Osm(r,Localiz~e) -> Msb2(STR!1),Osm(r!1,Localiz~e)
    @ 10.0
'OsmEx_Unbinds_Msb2' Msb2(STR!1),Osm(r!1,Localiz~e) -> Msb2(STR),Osm(r,Localiz~e
    ) @ 1.0
'OsmEx_Activates_Msb2' Msb2(STR!1,X~ia),Osm(r!1,Localiz~e) -> Msb2(STR!1,X~a),
    Osm(r!1,Localiz~e) @ 2.0
'OsmInt_Binds_Msb2E' Msb2(STR,OsmIntern),Osm(r,Localiz~i) -> Msb2(STR!1,
    OsmIntern),Osm(r!1,Localiz~i) @ 0.05
'OsmInt_Unbinds_Msb2E' Msb2(STR!1,OsmIntern),Osm(r!1,Localiz~i) -> Msb2(STR,
    OsmIntern),Osm(r,Localiz~i) @ 0.1
'OsmInt_UnbindsCoo_Msb2E' Msb2(STR,OsmIntern!1),Osm(r!1,Localiz~i) -> Msb2(STR,
    OsmIntern),Osm(r,Localiz~i) @ 0.1
'OsmInt_BindsCoo_Msb2E Ref1' Msb2(STR!1,OsmIntern),Osm(r!1,Localiz~i),Osm(r,
    Localiz~i) -> Msb2(STR!1,OsmIntern!2),Osm(r!1,Localiz~i),Osm(r!2,Localiz~i)
    @ 3.0
'OsmInt_UnbindsCoo_Msb2E Ref1' Msb2(STR!1,OsmIntern!2),Osm(r!1,Localiz~i),Osm(r
    !2,Localiz~i) -> Msb2(STR!1,OsmIntern),Osm(r!1,Localiz~i),Osm(r,Localiz~i) @
     0.1
'OsmIntCoo_Deactivates_Msb2E' Msb2(X~a,OsmIntern!2,STR!1),Osm(r!1,Localiz~i),Osm
    (r!2,Localiz~i) -> Msb2(X~ia,OsmIntern!2,STR!1),Osm(r!1,Localiz~i),Osm(r!2,
    Localiz~i) @ 10.0
'Fps1_Transp_Osm1' :Membrane Fps1(Osm,A~a),Osm(Localiz~i,r) -> :Membrane Fps1(
    Osm!1,A~a),Osm(Localiz~i,r!1) @ 100.0
'Fps1_Transp_Osm2 Ref1' :Membrane Fps1(Osm!1,A~a),Osm(Localiz~i,r!1) -> :
    Membrane Fps1(Osm,A~a) @ 10.0
'Sho1Act_Deactiv_Fps1_WOBinding' :Membrane Fps1(A~a,Hog1),Sho1(X~a) -> :Membrane
     Fps1(A~ia,Hog1),Sho1(X~a) @ 0.5
'Fps1_Activation' :Membrane Fps1(Hog1,A~ia) -> :Membrane Fps1(Hog1,A~a) @ 0.0
'Sln1Deact_Deactivates_Fps1' :Membrane Fps1(Hog1,A~a),Sln1_SensD(X~ia) -> :
    Membrane Fps1(Hog1,A~ia),Sln1_SensD(X~ia) @ 0.5

# General Rules
'Hog1pp_Unbinds_GlycFeedback' :Cytosol Hog1(Kinase!1),GlycFeedback(Hog1!1,X) ->
    :Cytosol Hog1(Kinase),GlycFeedback(Hog1,X) @ 0.1
'Hog1pp_Binds_GlycFeedback Ref2' :Cytosol GlycFeedback(Hog1,GlycProd,X~ia),Hog1(
    Kinase,TF,Thr174,CD,Ste50,Tyr176,Phosphat,Ptc23,Sho1) -> :Cytosol
```

```
    GlycFeedback(Hog1!1,GlycProd,X~ia),Hog1(Kinase!1,TF,Thr174,CD,Ste50,Tyr176,
    Phosphat,Ptc23,Sho1) @ 0.005
'Hog1pp_Unbinds_GlycFeedback Ref2' :Cytosol GlycFeedback(Hog1!1,GlycProd,X~ia),
    Hog1(Kinase!1,TF,Thr174,CD,Ste50,Tyr176,Phosphat,Ptc23,Sho1) -> :Cytosol
    GlycFeedback(Hog1,GlycProd,X~ia),Hog1(Kinase,TF,Thr174,CD,Ste50,Tyr176,
    Phosphat,Ptc23,Sho1) @ 0.1
'Hog1pp_Activ_GlycFeedb' :Cytosol GlycFeedback(Hog1!1,X~ia),Hog1(Thr174~p,Tyr176
    ~p,Kinase!1) -> :Cytosol GlycFeedback(Hog1!1,X~a),Hog1(Thr174~p,Tyr176~p,
    Kinase!1) @ 1.0
'GlycFeedB_Unbinds_GlycProd' GlycFeedback(X,GlycProd!1),GlycProd(Hog1!1,X) ->
    GlycFeedback(X,GlycProd),GlycProd(Hog1,X) @ 0.1
'GlycFeedB_Binds_GlycProd Ref2' GlycFeedback(X,GlycProd,Hog1),GlycProd(Hog1,X~ia
    ) -> GlycFeedback(X,GlycProd!1,Hog1),GlycProd(Hog1!1,X~ia) @ 0.005
'GlycFeedB_Unbinds_GlycProd Ref2' GlycFeedback(X,GlycProd!1,Hog1),GlycProd(Hog1
    !1,X~ia) -> GlycFeedback(X,GlycProd,Hog1),GlycProd(Hog1,X~ia) @ 0.1
'GlycFeedB_Act_GlycProd' GlycFeedback(X~a,GlycProd!1),GlycProd(Hog1!1,X~ia) ->
    GlycFeedback(X~a,GlycProd!1),GlycProd(Hog1!1,X~a) @ 1.0
'GlycProd_Synt_OsmInt' :Cytosol GlycProd(Hog1,X~a) -> :Cytosol GlycProd(Hog1,X~a
    ),Osm(r,Localiz~i) @ 0.08
'GlycProd_BasalSynt_OsmInt' :Cytosol GlycProd(Hog1,X~ia) -> :Cytosol GlycProd(
    Hog1,X~ia),Osm(r,Localiz~i) @ 0.0005
'GlycFeedB_Deact' GlycFeedback(X~a) -> GlycFeedback(X~ia) @ 0.01
'GlycProd_Deact' GlycProd(Hog1,X~a) -> GlycProd(Hog1,X~ia) @ 0.004
'Hog1ppnuc_Unbinds_TFGen' :Nucleus TF(Hog1pp!1),Hog1(TF!1) -> :Nucleus TF(Hog1pp
    ),Hog1(TF) @ 0.01
'Hog1ppnuc_Binds_TFGen Ref2' :Nucleus Hog1(TF,Ste50,Tyr176~p,Thr174~p),TF(Hog1pp
    ,Gene!_) -> :Nucleus Hog1(TF!1,Ste50,Tyr176~p,Thr174~p),TF(Hog1pp!1,Gene!_)
    @ 0.00005
'Hog1ppnuc_Unbinds_TFGen Ref2' :Nucleus Hog1(TF!1,Ste50,Tyr176~p,Thr174~p),TF(
    Hog1pp!1,Gene!_) -> :Nucleus Hog1(TF,Ste50,Tyr176~p,Thr174~p),TF(Hog1pp,Gene
    !_) @ 0.01
'Hog1ppnuc_Unbinds_TFGen Ref1' :Nucleus TF(Hog1pp!1),Hog1(TF!1,Tyr176~u) -> :
    Nucleus TF(Hog1pp),Hog1(TF,Tyr176~u) @ 1.0
'Hog1ppnuc_Unbinds_TFGen Ref3' :Nucleus TF(Hog1pp!1),Hog1(TF!1,Thr174~u) -> :
    Nucleus TF(Hog1pp),Hog1(TF,Thr174~u) @ 1.0
'Hog1ppnuc_Unbinds_TFSko1' :Nucleus Sko1(Hog1pp!1),Hog1(TF!1) -> :Nucleus Sko1(
    Hog1pp),Hog1(TF) @ 0.1
'Hog1ppnuc_Binds_TFSko1 Ref1' :Nucleus Sko1(Hog1pp),Hog1(TF,Ste50,Tyr176~p,
    Thr174~p) -> :Nucleus Sko1(Hog1pp!1),Hog1(TF!1,Ste50,Tyr176~p,Thr174~p) @
    0.00005
'Hog1ppnuc_Unbinds_TFSko1 Ref1' :Nucleus Sko1(Hog1pp!1),Hog1(TF!1,Ste50,Tyr176~p
    ,Thr174~p) -> :Nucleus Sko1(Hog1pp),Hog1(TF,Ste50,Tyr176~p,Thr174~p) @ 0.1
```

```
'Hog1ppnuc_Unbinds_TFSko1 Ref2' :Nucleus Sko1(Hog1pp!1),Hog1(TF!1,Thr174~u) -> :
    Nucleus Sko1(Hog1pp),Hog1(TF,Thr174~u) @ 1.0
'Hog1ppnuc_Unbinds_TFSko1 Ref3' :Nucleus Sko1(Hog1pp!1),Hog1(TF!1,Tyr176~u) -> :
    Nucleus Sko1(Hog1pp),Hog1(TF,Tyr176~u) @ 1.0
'TF_Unbinds_GeneVenus' Venus(TF!1),TF(Gene!1) -> Venus(TF),TF(Gene) @ 0.1
'TF_Binds_GeneVenus Ref1' TF(Hog1pp,Gene),Venus(TF,PolII) -> TF(Hog1pp,Gene!1),
    Venus(TF!1,PolII) @ 0.001
'TF_Unbinds_GeneVenus Ref1' TF(Hog1pp,Gene!1),Venus(TF!1,PolII) -> TF(Hog1pp,
    Gene),Venus(TF,PolII) @ 0.1
'TF_Unbinds_GeneCherry' TF(Gene!1),mCherry(TF!1) -> TF(Gene),mCherry(TF) @ 0.1
'TF_Binds_GeneCherry Ref1' TF(Hog1pp,Gene),mCherry(TF,PolII) -> TF(Hog1pp,Gene
    !1),mCherry(TF!1,PolII) @ 0.001
'TF_Unbinds_GeneCherry Ref1' TF(Hog1pp,Gene!1),mCherry(TF!1,PolII) -> TF(Hog1pp,
    Gene),mCherry(TF,PolII) @ 0.1
'TF_Unbinds_Gene' gene(TF!1),TF(Gene!1) -> gene(TF),TF(Gene) @ 0.1
'TF_Binds_Gene Ref1' gene(TF,PolII),TF(Hog1pp,Gene) -> gene(TF!1,PolII),TF(
    Hog1pp,Gene!1) @ 0.001
'TF_Unbinds_Gene Ref1' gene(TF!1,PolII),TF(Hog1pp,Gene!1) -> gene(TF,PolII),TF(
    Hog1pp,Gene) @ 0.1
'PolII_Unbinds_VenTFHog' Venus(PolII!1),PolII(Gene!1) -> Venus(PolII),PolII(Gene
    ) @ 0.005
'PolII_Binds_InitComplex Ref1' :Nucleus Hog1(Thr174~p,Tyr176~p,TF!1),TF(Hog1pp
    !1,Gene!2),Venus(TF!2,PolII),PolII(Remodel,Gene) -> :Nucleus Hog1(Thr174~p,
    Tyr176~p,TF!1),TF(Hog1pp!1,Gene!2),Venus(TF!2,PolII!3),PolII(Remodel,Gene!3)
     @ 0.005
'PolII_Unbinds_InitComplex Ref1' :Nucleus Hog1(Thr174~p,Tyr176~p,TF!1),TF(Hog1pp
    !1,Gene!2),Venus(TF!2,PolII!3),PolII(Remodel,Gene!3) -> :Nucleus Hog1(Thr174
    ~p,Tyr176~p,TF!1),TF(Hog1pp!1,Gene!2),Venus(TF!2,PolII),PolII(Remodel,Gene)
    @ 0.01
'PolII_Unbinds_InitComplex Ref2' PolII(Gene!2),TF(Hog1pp,Gene!1),Venus(PolII!2,
    TF!1) -> PolII(Gene),TF(Hog1pp,Gene!1),Venus(PolII,TF!1) @ 1.0
'PolII_Unbinds_InitComplex Ref3' PolII(Gene!1),Venus(PolII!1,TF) -> PolII(Gene),
    Venus(PolII,TF) @ 1.0
'PolII_Unbinds_CherTFHog' PolII(Gene!1),mCherry(PolII!1) -> PolII(Gene),mCherry(
    PolII) @ 0.005
'PolII_Binds_InitComplexCherry Ref1' :Nucleus Hog1(Thr174~p,Tyr176~p,TF!1),TF(
    Hog1pp!1,Gene!2),PolII(Remodel,Gene),mCherry(PolII,TF!2) -> :Nucleus Hog1(
    Thr174~p,Tyr176~p,TF!1),TF(Hog1pp!1,Gene!2),PolII(Remodel,Gene!3),mCherry(
    PolII!3,TF!2) @ 0.005
'PolII_Unbinds_InitComplexCherry Ref1' :Nucleus Hog1(Thr174~p,Tyr176~p,TF!1),TF(
    Hog1pp!1,Gene!2),PolII(Remodel,Gene!3),mCherry(PolII!3,TF!2) -> :Nucleus
    Hog1(Thr174~p,Tyr176~p,TF!1),TF(Hog1pp!1,Gene!2),PolII(Remodel,Gene),mCherry
```

```
    (PolII,TF!2) @ 0.01
'PolII_Unbinds_InitComplexCherry Ref2' PolII(Gene!2),TF(Hog1pp,Gene!1),mCherry(
    TF!1,PolII!2) -> PolII(Gene),TF(Hog1pp,Gene!1),mCherry(TF!1,PolII) @ 1.0
'PolII_Unbinds_InitComplexCherry Ref3' PolII(Gene!1),mCherry(TF,PolII!1) ->
    PolII(Gene),mCherry(TF,PolII) @ 1.0
'PolII_Unbinds_GeneTFHog' gene(PolII!1),PolII(Gene!1) -> gene(PolII),PolII(Gene)
     @ 0.005
'PolII_Binds_InitComplexGene Ref1' :Nucleus Hog1(Thr174~p,Tyr176~p,TF!1),TF(
    Hog1pp!1,Gene!2),gene(TF!2,PolII),PolII(Remodel,Gene) -> :Nucleus Hog1(
    Thr174~p,Tyr176~p,TF!1),TF(Hog1pp!1,Gene!2),gene(TF!2,PolII!3),PolII(Remodel
    ,Gene!3) @ 0.005
'PolII_Unbinds_InitComplexGene Ref1' :Nucleus Hog1(Thr174~p,Tyr176~p,TF!1),TF(
    Hog1pp!1,Gene!2),gene(TF!2,PolII!3),PolII(Remodel,Gene!3) -> :Nucleus Hog1(
    Thr174~p,Tyr176~p,TF!1),TF(Hog1pp!1,Gene!2),gene(TF!2,PolII),PolII(Remodel,
    Gene) @ 0.02
'PolII_Unbinds_InitComplexGene Ref2' PolII(Gene!2),TF(Hog1pp,Gene!1),gene(PolII
    !2,TF!1) -> PolII(Gene),TF(Hog1pp,Gene!1),gene(PolII,TF!1) @ 1.0
'PolII_Unbinds_InitComplexGene Ref3' PolII(Gene!1),gene(PolII!1,TF) -> PolII(
    Gene),gene(PolII,TF) @ 1.0
'Remod_Unbinds_InitComplVen' Venus(Remodel!1),Remodeler(PolII!1) -> Venus(
    Remodel),Remodeler(PolII) @ 0.0001
'Remod_Binds_InitCompl_2 Ref1' :Nucleus Hog1(Thr174~p,Tyr176~p,TF!1),TF(Hog1pp
    !1,Gene!2),Venus(TF!2,PolII!3,Remodel),PolII(Remodel,Gene!3),Remodeler(PolII
    ) -> :Nucleus Hog1(Thr174~p,Tyr176~p,TF!1),TF(Hog1pp!1,Gene!2),Venus(TF!2,
    PolII!3,Remodel!4),PolII(Remodel,Gene!3),Remodeler(PolII!4) @ 0.001
'Remod_Unbinds_InitCompl_2 Ref1' :Nucleus Hog1(Thr174~p,Tyr176~p,TF!1),TF(Hog1pp
    !1,Gene!2),Venus(TF!2,PolII!3,Remodel!4),PolII(Remodel,Gene!3),Remodeler(
    PolII!4) -> :Nucleus Hog1(Thr174~p,Tyr176~p,TF!1),TF(Hog1pp!1,Gene!2),Venus(
    TF!2,PolII!3,Remodel),PolII(Remodel,Gene!3),Remodeler(PolII) @ 0.0001
'Remod_Unbinds_InitComplCherry' Remodeler(PolII!1),mCherry(Remodel!1) ->
    Remodeler(PolII),mCherry(Remodel) @ 0.0001
'Remod_Binds_InitComplCherry_2 Ref1' :Nucleus Hog1(Thr174~p,Tyr176~p,TF!1),TF(
    Hog1pp!1,Gene!2),PolII(Remodel,Gene!3),Remodeler(PolII),mCherry(TF!2,PolII
    !3,Remodel) -> :Nucleus Hog1(Thr174~p,Tyr176~p,TF!1),TF(Hog1pp!1,Gene!2),
    PolII(Remodel,Gene!3),Remodeler(PolII!4),mCherry(TF!2,PolII!3,Remodel!4) @
    0.001
'Remod_Unbinds_InitComplCherry_2 Ref1' :Nucleus Hog1(Thr174~p,Tyr176~p,TF!1),TF(
    Hog1pp!1,Gene!2),PolII(Remodel,Gene!3),Remodeler(PolII!4),mCherry(TF!2,PolII
    !3,Remodel!4) -> :Nucleus Hog1(Thr174~p,Tyr176~p,TF!1),TF(Hog1pp!1,Gene!2),
    PolII(Remodel,Gene!3),Remodeler(PolII),mCherry(TF!2,PolII!3,Remodel) @
    0.0001
'Remod_Unbinds_InitComplGene' Remodeler(PolII!1),gene(Remodel!1) -> Remodeler(
```

```
    PolII),gene(Remodel) @ 0.0001
'Remod_Binds_InitComplGene2 Ref1' :Nucleus TF(Hog1pp!1,Gene!2),Hog1(Thr174~p,
    Tyr176~p,TF!1),PolII(Remodel,Gene!3),Remodeler(PolII),gene(TF!2,PolII!3,
    Remodel) -> :Nucleus TF(Hog1pp!1,Gene!2),Hog1(Thr174~p,Tyr176~p,TF!1),PolII(
    Remodel,Gene!3),Remodeler(PolII!4),gene(TF!2,PolII!3,Remodel!4) @ 0.001
'Remod_Unbinds_InitComplGene2 Ref1' :Nucleus TF(Hog1pp!1,Gene!2),Hog1(Thr174~p,
    Tyr176~p,TF!1),PolII(Remodel,Gene!3),Remodeler(PolII!4),gene(TF!2,PolII!3,
    Remodel!4) -> :Nucleus TF(Hog1pp!1,Gene!2),Hog1(Thr174~p,Tyr176~p,TF!1),
    PolII(Remodel,Gene!3),Remodeler(PolII),gene(TF!2,PolII!3,Remodel) @ 0.0001
'TrscrptCmplx_cr_VenusmRNA2' :Nucleus Hog1(Thr174~p,Tyr176~p,TF!1),TF(Hog1pp!1,
    Gene!2),Venus(TF!2,PolII!3,Remodel!4),PolII(Remodel,Gene!3),Remodeler(PolII
    !4) -> :Nucleus Hog1(Thr174~p,Tyr176~p,TF!1),TF(Hog1pp!1,Gene!2),Venus(TF!2,
    PolII!3,Remodel!4),PolII(Remodel,Gene!3),Remodeler(PolII!4),mRNA_Venus() @
    0.5
'TrscrptCmplx_cr_GeneRNA2' :Nucleus Hog1(Thr174~p,Tyr176~p,TF!1),TF(Hog1pp!1,
    Gene!2),PolII(Remodel,Gene!3),Remodeler(PolII!4),gene(PolII!3,TF!2,Remodel
    !4) -> :Nucleus Hog1(Thr174~p,Tyr176~p,TF!1),TF(Hog1pp!1,Gene!2),PolII(
    Remodel,Gene!3),Remodeler(PolII!4),gene(PolII!3,TF!2,Remodel!4),mRNA_Gene()
    @ 0.5
'TrscrptCmplx_cr_mCherryRNA2' :Nucleus Hog1(Thr174~p,Tyr176~p,TF!1),TF(Hog1pp!1,
    Gene!2),PolII(Remodel,Gene!3),Remodeler(PolII!4),mCherry(PolII!3,TF!2,
    Remodel!4) -> :Nucleus Hog1(Thr174~p,Tyr176~p,TF!1),TF(Hog1pp!1,Gene!2),
    PolII(Remodel,Gene!3),Remodeler(PolII!4),mCherry(PolII!3,TF!2,Remodel!4),
    mRNA_mCherry() @ 0.5

# Sln1 Branch
'Sln1_Dimerization Ref1' Sln1_HkD(DimR,s!1),Sln1_SensD(k!1,X~ia),Sln1_HkD(DimR,s
    !2),Sln1_SensD(k!2,X~ia) -> Sln1_HkD(DimR!3,s!2),Sln1_SensD(k!2,X~ia),
    Sln1_HkD(DimR!3,s!1),Sln1_SensD(k!1,X~ia) @ 0.00016
'Sln1_Undimerization Ref1' Sln1_HkD(DimR!3,s!1),Sln1_SensD(k!1,X~ia),Sln1_HkD(
    DimR!3,s!2),Sln1_SensD(k!2,X~ia) -> Sln1_HkD(DimR,s!2),Sln1_SensD(k!2,X~ia),
    Sln1_HkD(DimR,s!1),Sln1_SensD(k!1,X~ia) @ 0.1
'Sln1_Dimerization Ref2' Sln1_HkD(DimR,s!1),Sln1_SensD(k!1,X~a),Sln1_HkD(DimR,s
    !2),Sln1_SensD(k!2,X~a) -> Sln1_HkD(DimR!3,s!2),Sln1_SensD(k!2,X~a),Sln1_HkD
    (DimR!3,s!1),Sln1_SensD(k!1,X~a) @ 0.00016
'Sln1_Undimerization Ref2' Sln1_HkD(DimR!3,s!1),Sln1_SensD(k!1,X~a),Sln1_HkD(
    DimR!3,s!2),Sln1_SensD(k!2,X~a) -> Sln1_HkD(DimR,s!2),Sln1_SensD(k!2,X~a),
    Sln1_HkD(DimR,s!1),Sln1_SensD(k!1,X~a) @ 0.1
'Sln1SensAct_AutPhosph_Sln1His' Sln1_SensD(k!1,X~a),Sln1_HkD(His576~u,s!1) ->
    Sln1_SensD(k!1,X~a),Sln1_HkD(His576~p,s!1) @ 1.0
'Sln1_CrossPh' Sln1_HkD(s!4,r!2,DimR!5),Sln1_RecD(h!2,Asp1144~u),Sln1_RecD(h!1),
    Sln1_HkD(r!1,DimR!5,His576~p,s!3),Sln1_SensD(k!3),Sln1_SensD(k!4) ->
```

```
    Sln1_HkD(s!4,r!1,DimR!5),Sln1_RecD(h!1,Asp1144~p),Sln1_RecD(h!3),Sln1_HkD(r
    !3,DimR!5,His576~u,s!2),Sln1_SensD(k!2),Sln1_SensD(k!4) @ 40.0
'Ypd1_Unbinds_Sln1RecD' :Membrane Sln1_RecD(Ypd1!1),Ypd1(Sln1!1) -> :Membrane
    Sln1_RecD(Ypd1),Ypd1(Sln1) @ 0.0001
'Ypd1_Binds_Sln1RecD Ref1' :Membrane Sln1_RecD(Ypd1),Ypd1(Sln1,His64~u) -> :
    Membrane Sln1_RecD(Ypd1!1),Ypd1(Sln1!1,His64~u) @ 0.0005
'Ypd1_Unbinds_Sln1RecD Ref1' :Membrane Sln1_RecD(Ypd1!1),Ypd1(Sln1!1,His64~u) ->
     :Membrane Sln1_RecD(Ypd1),Ypd1(Sln1,His64~u) @ 0.0001
'Sln1RecD_P_YPD1' :Membrane Ypd1(Sln1!1,His64~u),Sln1_RecD(Asp1144~p,Ypd1!1) ->
    :Membrane Ypd1(Sln1!1,His64~p),Sln1_RecD(Asp1144~u,Ypd1!1) @ 50.0
'Sln1RecD_CrossPh_Ypd1' :Membrane Sln1_HkD(r!2,DimR!5),Sln1_HkD(r!1,DimR!5),
    Sln1_RecD(h!1,Ypd1!3),Sln1_RecD(h!2,Asp1144~p),Ypd1(His64~u,Sln1!3) -> :
    Membrane Sln1_HkD(r!1,DimR!5),Sln1_HkD(r!3,DimR!5),Sln1_RecD(h!3,Ypd1!2),
    Sln1_RecD(h!1,Asp1144~u),Ypd1(His64~p,Sln1!2) @ 50.0
'Ssk1_Desphos' Ssk1(Asp554~p) -> Ssk1(Asp554~u) @ 0.008
'Ssk1Dimerpu_Dephosph' Ssk1(Dimer!1,Asp554~u),Ssk1(Dimer!1,Asp554~p) -> Ssk1(
    Dimer!1,Asp554~u),Ssk1(Dimer!1,Asp554~u) @ 20.0
'Ypd1_Binds_Ssk1Membr Ref1' :Membrane Ypd1(Ssk1,His64~p),Ssk1(Ypd1,Asp554~u) ->
    :Membrane Ypd1(Ssk1!1,His64~p),Ssk1(Ypd1!1,Asp554~u) @ 0.05
'Ypd1_UnBind_Ssk1_membr' :Membrane Ypd1(Ssk1!1),Ssk1(Asp554,Ypd1!1) -> :Membrane
     Ypd1(Ssk1),Ssk1(Asp554,Ypd1) @ 0.01
'Ypd1_UnBind_Ssk1_membr Ref2' :Membrane Ypd1(Ssk1!1),Ssk1(Asp554~u,Ypd1!1) -> :
    Membrane Ypd1(Ssk1),Ssk1(Asp554~u,Ypd1) @ 0.01
'Ypd1_UnBind_Ssk1_membr Ref3' :Membrane Ypd1(Ssk1!1),Ssk1(Asp554~p,Ypd1!1) -> :
    Membrane Ypd1(Ssk1),Ssk1(Asp554~p,Ypd1) @ 0.1
'Ypd1_Phos_Ssk1_membr' :Membrane Ypd1(His64~p,Ssk1!1),Ssk1(Asp554~u,Ypd1!1) -> :
    Membrane Ypd1(His64~u,Ssk1!1),Ssk1(Asp554~p,Ypd1!1) @ 50.0
'Ypd1_Phos_Ssk1Dimer_membr' :Membrane Ypd1(His64~p,Ssk1!1),Ssk1(Asp554~p,Ypd1!1,
    Dimer!2),Ssk1(Dimer!2,Asp554~u) -> :Membrane Ypd1(His64~u,Ssk1!1),Ssk1(
    Asp554~p,Ypd1!1,Dimer!2),Ssk1(Dimer!2,Asp554~p) @ 30.0
'Ssk1_Dimerization Ref1' :Membrane Ssk1(Dimer),Ssk1(Dimer) -> :Membrane Ssk1(
    Dimer!1),Ssk1(Dimer!1) @ 0.005
'Ssk1_Undimerization Ref1' :Membrane Ssk1(Dimer!1),Ssk1(Dimer!1) -> :Membrane
    Ssk1(Dimer),Ssk1(Dimer) @ 0.01
'Ssk1_Dimerization Ref2' :Cytosol Ssk1(Dimer),Ssk1(Dimer) -> :Cytosol Ssk1(Dimer
    !1),Ssk1(Dimer!1) @ 0.0005
'Ssk1_Undimerization Ref2' :Cytosol Ssk1(Dimer!1),Ssk1(Dimer!1) -> :Cytosol Ssk1
    (Dimer),Ssk1(Dimer) @ 0.01
'SSk1Mono_Unbinds_Ssk22_Cyto' :Cytosol Ssk1(Ssk2_22!1),Ssk22(s98_179!1) -> :
    Cytosol Ssk1(Ssk2_22),Ssk22(s98_179) @ 0.1
'Ssk1cyto_Binds_Ssk22_cyto Ref3' :Cytosol Ssk1(Dimer!1,Asp554~u),Ssk1(Ssk2_22,
    Dimer!1,Asp554~u),Ssk22(s98_179,Thr1460~u) -> :Cytosol Ssk1(Dimer!1,Asp554~u
```

```
    ),Ssk1(Ssk2_22!2,Dimer!1,Asp554~u),Ssk22(s98_179!2,Thr1460~u) @ 0.000005
'Ssk1cyto_Unbinds_Ssk22_cyto Ref3' :Cytosol Ssk1(Dimer!1,Asp554~u),Ssk1(Ssk2_22
    !2,Dimer!1,Asp554~u),Ssk22(s98_179!2,Thr1460~u) -> :Cytosol Ssk1(Dimer!1,
    Asp554~u),Ssk1(Ssk2_22,Dimer!1,Asp554~u),Ssk22(s98_179,Thr1460~u) @ 0.1
'Ssk1Mono_Unbinds_Ssk2_Cyto' :Cytosol Ssk1(Ssk2_22!1),Ssk2(s294_413!1) -> :
    Cytosol Ssk1(Ssk2_22),Ssk2(s294_413) @ 0.1
'Ssk1_Binds_Ssk2_cyto Ref2' :Cytosol Ssk1(Dimer!1,Asp554~u),Ssk1(Ssk2_22,Dimer
    !1,Asp554~u),Ssk2(s294_413,Thr1460~u) -> :Cytosol Ssk1(Dimer!1,Asp554~u),
    Ssk1(Ssk2_22!2,Dimer!1,Asp554~u),Ssk2(s294_413!2,Thr1460~u) @ 0.000005
'Ssk1_Unbinds_Ssk2_cyto Ref2' :Cytosol Ssk1(Dimer!1,Asp554~u),Ssk1(Ssk2_22!2,
    Dimer!1,Asp554~u),Ssk2(s294_413!2,Thr1460~u) -> :Cytosol Ssk1(Dimer!1,Asp554
    ~u),Ssk1(Ssk2_22,Dimer!1,Asp554~u),Ssk2(s294_413,Thr1460~u) @ 0.1

# Sho1 Branch
'Hkr1_Unbinds_Sho1' Hkr1(Sho1!1),Sho1(Mucin!1) -> Hkr1(Sho1),Sho1(Mucin) @ 0.1
'Hkr1_Binds_Sho1 Ref1' Hkr1(Sho1,X~a),Sho1(Mucin) -> Hkr1(Sho1!1,X~a),Sho1(Mucin
    !1) @ 0.0001
'Hkr1_Unbinds_Sho1 Ref1' Hkr1(Sho1!1,X~a),Sho1(Mucin!1) -> Hkr1(Sho1,X~a),Sho1(
    Mucin) @ 0.1
'Hkr1_Binds_Sho1 Ref2' Sho1(Mucin),Hkr1(Sho1,X~ia) -> Sho1(Mucin!1),Hkr1(Sho1!1,
    X~ia) @ 0.0001
'Hkr1_Unbinds_Sho1 Ref2' Sho1(Mucin!1),Hkr1(Sho1!1,X~ia) -> Sho1(Mucin),Hkr1(
    Sho1,X~ia) @ 0.1
'Hkr1_Activates_Sho1 Ref1' Hkr1(Sho1!1,X~a),Sho1(Mucin!1,Hog1~u,X~ia) -> Hkr1(
    Sho1!1,X~a),Sho1(Mucin!1,Hog1~u,X~a) @ 1.0
'Hkr1_DEActivates_Sho1' Sho1(Mucin!1,X~a),Hkr1(Sho1!1,X~ia) -> Sho1(Mucin!1,X~ia
    ),Hkr1(Sho1!1,X~ia) @ 10.0
'Msb2_Unbinds_Sho1' Sho1(Mucin!1),Msb2(Sho1!1) -> Sho1(Mucin),Msb2(Sho1) @ 0.1
'Msb2_Binds_Sho1 Ref1' Msb2(Sho1,X~a),Sho1(Mucin) -> Msb2(Sho1!1,X~a),Sho1(Mucin
    !1) @ 0.0001
'Msb2_Unbinds_Sho1 Ref1' Msb2(Sho1!1,X~a),Sho1(Mucin!1) -> Msb2(Sho1,X~a),Sho1(
    Mucin) @ 0.1
'Msb2_Binds_Sho1 Ref2' Sho1(Mucin),Msb2(Sho1,X~ia) -> Sho1(Mucin!1),Msb2(Sho1!1,
    X~ia) @ 0.0001
'Msb2_Unbinds_Sho1 Ref2' Sho1(Mucin!1),Msb2(Sho1!1,X~ia) -> Sho1(Mucin),Msb2(
    Sho1,X~ia) @ 0.1
'Msb2_Activates_Sho1 Ref1' Msb2(Sho1!1,X~a),Sho1(Mucin!1,Hog1~u,X~ia) -> Msb2(
    Sho1!1,X~a),Sho1(Mucin!1,Hog1~u,X~a) @ 1.0
'Msb2_DEActivates_Sho1' Sho1(Mucin!1,X~a),Msb2(Sho1!1,X~ia) -> Sho1(Mucin!1,X~ia
    ),Msb2(Sho1!1,X~ia) @ 10.0
'Sho1_Activates_Cdc24Unbound' :Membrane Sho1(Cdc24,X~a),Cdc24(Sho1,X~ia) -> :
    Membrane Sho1(Cdc24,X~a),Cdc24(Sho1,X~a) @ 0.0001
```

```
'Cdc24_Deactivation' Cdc24(X~a) -> Cdc24(X~ia) @ 0.005
'Cla4membr_Unbinds_Ste11membr' :Membrane Cla4(Ste11!1),Ste11(Cla4!1) -> :
    Membrane Cla4(Ste11),Ste11(Cla4) @ 0.1
'Cla4membr_Binds_Ste11membr Ref2' :Membrane Cla4(Ste11,X~a),Ste11(Cla4,Ste20) ->
     :Membrane Cla4(Ste11!1,X~a),Ste11(Cla4!1,Ste20) @ 0.0000005
'Cla4membr_Unbinds_Ste11membr Ref2' :Membrane Cla4(Ste11!1,X~a),Ste11(Cla4!1,
    Ste20) -> :Membrane Cla4(Ste11,X~a),Ste11(Cla4,Ste20) @ 0.1
'Cla4cyto_Unbinds_Ste11cyto' :Cytosol Cla4(Ste11!1),Ste11(Cla4!1) -> :Cytosol
    Cla4(Ste11),Ste11(Cla4) @ 0.1
'Cla4cyto_Binds_Ste11cyto Ref2' :Cytosol Cla4(Ste11,X~a),Ste11(Cla4,Ste20) -> :
    Cytosol Cla4(Ste11!1,X~a),Ste11(Cla4!1,Ste20) @ 0.00000005
'Cla4cyto_Unbinds_Ste11cyto Ref2' :Cytosol Cla4(Ste11!1,X~a),Ste11(Cla4!1,Ste20)
     -> :Cytosol Cla4(Ste11,X~a),Ste11(Cla4,Ste20) @ 0.1
'Cla4_PhosphDirect_Ste11u_Membr' :Membrane Cla4(X~a,Ste11!1),Ste11(
    Ser281Ser285Thr286~u,Cla4!1) -> :Membrane Cla4(X~a,Ste11!1),Ste11(
    Ser281Ser285Thr286~p,Cla4!1) @ 0.1
'Cla4_PhosphDirect_Ste11u_Cyto' :Cytosol Cla4(X~a,Ste11!1),Ste11(
    Ser281Ser285Thr286~u,Cla4!1) -> :Cytosol Cla4(X~a,Ste11!1),Ste11(
    Ser281Ser285Thr286~p,Cla4!1) @ 0.1
'Cla4_PhosphDirect_Ste11p_Membr' :Membrane Cla4(X~a,Ste11!1),Ste11(
    Ser281Ser285Thr286~p,Cla4!1) -> :Membrane Cla4(X~a,Ste11!1),Ste11(
    Ser281Ser285Thr286~pp,Cla4!1) @ 0.1
'Cla4_PhosphDirect_Ste11p_Cyto' :Cytosol Cla4(X~a,Ste11!1),Ste11(
    Ser281Ser285Thr286~p,Cla4!1) -> :Cytosol Cla4(X~a,Ste11!1),Ste11(
    Ser281Ser285Thr286~pp,Cla4!1) @ 0.1
'Cla4_PhosphDirect_Ste11pp_Membr' :Membrane Cla4(X~a,Ste11!1),Ste11(
    Ser281Ser285Thr286~pp,Cla4!1) -> :Membrane Cla4(X~a,Ste11!1),Ste11(
    Ser281Ser285Thr286~ppp,Cla4!1) @ 0.1
'Cla4_PhosphDirect_Ste11pp_Cyto' :Cytosol Cla4(X~a,Ste11!1),Ste11(
    Ser281Ser285Thr286~pp,Cla4!1) -> :Cytosol Cla4(X~a,Ste11!1),Ste11(
    Ser281Ser285Thr286~ppp,Cla4!1) @ 0.1
'Cla4_Deactivation_cyto Ref1' :Cytosol Cla4(X~a,CRIB) -> Cla4(X~ia,CRIB) @ 0.008
'Cla4_Deactivation_membr Ref1' :Membrane Cla4(X~a,CRIB) -> :Membrane Cla4(X~ia,
    CRIB) @ 0.008
'Ste20membr_Unbinds_Ste11membr' :Membrane Ste20(Ste11!1),Ste11(Ste20!1) -> :
    Membrane Ste20(Ste11),Ste11(Ste20) @ 0.1
'Ste20membr_Binds_Ste11membr Ref2' :Membrane Ste20(Ste11,X~a),Ste11(Ste20,Cla4)
    -> :Membrane Ste20(Ste11!1,X~a),Ste11(Ste20!1,Cla4) @ 0.000005
'Ste20membr_Unbinds_Ste11membr Ref2' :Membrane Ste20(Ste11!1,X~a),Ste11(Ste20!1,
    Cla4) -> :Membrane Ste20(Ste11,X~a),Ste11(Ste20,Cla4) @ 0.1
'Ste20cyto_Unbinds_Ste11cyto' :Cytosol Ste20(Ste11!1),Ste11(Ste20!1) -> :Cytosol
     Ste20(Ste11),Ste11(Ste20) @ 0.1
```

```
'Ste20cyto_Binds_Ste11cyto Ref2' :Cytosol Ste20(Ste11,X~a),Ste11(Ste20,Cla4) ->
    :Cytosol Ste20(Ste11!1,X~a),Ste11(Ste20!1,Cla4) @ 0.0000005
'Ste20cyto_Unbinds_Ste11cyto Ref2' :Cytosol Ste20(Ste11!1,X~a),Ste11(Ste20!1,
    Cla4) -> :Cytosol Ste20(Ste11,X~a),Ste11(Ste20,Cla4) @ 0.1
'Ste20_PhosphDirect_Ste11u_Membr' :Membrane Ste20(Ste11!1,X~a),Ste11(Ste20!1,
    Ser281Ser285Thr286~u) -> :Membrane Ste20(Ste11!1,X~a),Ste11(Ste20!1,
    Ser281Ser285Thr286~p) @ 0.1
'Ste20_PhosphDirect_Ste11u_Cyto' :Cytosol Ste20(Ste11!1,X~a),Ste11(Ste20!1,
    Ser281Ser285Thr286~u) -> :Cytosol Ste20(Ste11!1,X~a),Ste11(Ste20!1,
    Ser281Ser285Thr286~p) @ 0.1
'Ste20_PhosphDirect_Ste11p_Membr' :Membrane Ste20(Ste11!1,X~a),Ste11(Ste20!1,
    Ser281Ser285Thr286~p) -> :Membrane Ste20(Ste11!1,X~a),Ste11(Ste20!1,
    Ser281Ser285Thr286~pp) @ 0.1
'Ste20_PhosphDirect_Ste11p_Cyto' :Cytosol Ste20(Ste11!1,X~a),Ste11(Ste20!1,
    Ser281Ser285Thr286~p) -> :Cytosol Ste20(Ste11!1,X~a),Ste11(Ste20!1,
    Ser281Ser285Thr286~pp) @ 0.1
'Ste20_PhosphDirect_Ste11pp_Membr' :Membrane Ste20(Ste11!1,X~a),Ste11(Ste20!1,
    Ser281Ser285Thr286~pp) -> :Membrane Ste20(Ste11!1,X~a),Ste11(Ste20!1,
    Ser281Ser285Thr286~ppp) @ 0.1
'Ste20_PhosphDirect_Ste11pp_Cyto' :Cytosol Ste20(Ste11!1,X~a),Ste11(Ste20!1,
    Ser281Ser285Thr286~pp) -> :Cytosol Ste20(Ste11!1,X~a),Ste11(Ste20!1,
    Ser281Ser285Thr286~ppp) @ 0.1
'Ste20_Deactivation_Membr Ref1' :Membrane Ste20(CRIB,X~a) -> :Membrane Ste20(
    CRIB,X~ia) @ 0.0008
'Ste20_Deactivation_Cyto Ref1' :Cytosol Ste20(CRIB,X~a) -> :Cytosol Ste20(CRIB,X
    ~ia) @ 0.0008
'Bem1membr_Binds_Cla4membr' :Membrane Cla4(PRR),Bem1(SH3b) -> :Membrane Cla4(PRR
    !1),Bem1(SH3b!1) @ 0.0005
'Bem1membr_Unbinds_Cla4membr' :Membrane Cla4(PRR!1),Bem1(SH3b!1) -> :Membrane
    Cla4(PRR),Bem1(SH3b) @ 0.1
'Bem1cyto_Binds_Cla4cyto' :Cytosol Bem1(SH3b),Cla4(PRR) -> :Cytosol Bem1(SH3b!1)
    ,Cla4(PRR!1) @ 0.00005
'Bem1cyto_Unbinds_Cla4cyto' :Cytosol Bem1(SH3b!1),Cla4(PRR!1) -> :Cytosol Bem1(
    SH3b),Cla4(PRR) @ 0.1
'Bem1membr_Unbinds_Ste20membr' :Membrane Bem1(SH3b!1),Ste20(PRR!1) -> :Membrane
    Bem1(SH3b!1),Ste20(PRR!1) @ 0.01
'Bem1membr_Binds_Ste20membr Ref1' :Membrane Bem1(SH3b),Ste20(PRR,X~ia) -> :
    Membrane Bem1(SH3b!1),Ste20(PRR!1,X~ia) @ 0.0005
'Bem1membr_Unbinds_Ste20membr Ref1' :Membrane Bem1(SH3b!1),Ste20(PRR!1,X~ia) ->
    :Membrane Bem1(SH3b),Ste20(PRR,X~ia) @ 0.01
'Bem1membr_Binds_Ste20membr Ref2' :Membrane Bem1(SH3b),Ste20(PRR,X~a) -> :
    Membrane Bem1(SH3b!1),Ste20(PRR!1,X~a) @ 0.00005
```

```
'Bem1membr_Unbinds_Ste20membr Ref2' :Membrane Bem1(SH3b!1),Ste20(PRR!1,X~a) -> :
    Membrane Bem1(SH3b),Ste20(PRR,X~a) @ 0.1
'Bem1cyto_Unbinds_Ste20cyto' :Cytosol Bem1(SH3b!1),Ste20(PRR!1) -> :Cytosol Bem1
    (SH3b),Ste20(PRR) @ 0.01
'Bem1cyto_Binds_Ste20cyto Ref1' :Cytosol Bem1(SH3b),Ste20(PRR,X~ia) -> :Cytosol
    Bem1(SH3b!1),Ste20(PRR!1,X~ia) @ 0.00005
'Bem1cyto_Unbinds_Ste20cyto Ref1' :Cytosol Bem1(SH3b!1),Ste20(PRR!1,X~ia) -> :
    Cytosol Bem1(SH3b),Ste20(PRR,X~ia) @ 0.01
'Bem1cyto_Binds_Ste20cyto Ref2' :Cytosol Bem1(SH3b),Ste20(PRR) -> :Cytosol Bem1(
    SH3b!1),Ste20(PRR!1) @ 0.000005
'Bem1cyto_Unbinds_Ste20cyto Ref2' :Cytosol Bem1(SH3b!1),Ste20(PRR!1) -> :Cytosol
     Bem1(SH3b),Ste20(PRR) @ 0.1
'Bem1membr_Unbinds_Cdc24membr' :Membrane Bem1(PB1!1),Cdc24(PB1!1) -> :Membrane
    Bem1(PB1),Cdc24(PB1) @ 0.1
'Bem1membr_Binds_Cdc24membr Ref1' :Membrane Bem1(PB1),Cdc24(PB1,PhosphSites~u)
    -> :Membrane Bem1(PB1!1),Cdc24(PB1!1,PhosphSites~u) @ 0.00005
'Bem1membr_Unbinds_Cdc24membr Ref1' :Membrane Bem1(PB1!1),Cdc24(PB1!1,
    PhosphSites~u) -> :Membrane Bem1(PB1),Cdc24(PB1,PhosphSites~u) @ 1.0
'Bem1cyto_Unbinds_Cdc24cyto' :Cytosol Bem1(PB1!1),Cdc24(PB1!1) -> :Cytosol Bem1(
    PB1),Cdc24(PB1) @ 0.1
'Bem1cyto_Binds_Cdc24cyto Ref1' :Cytosol Bem1(PB1),Cdc24(PB1,PhosphSites~u) -> :
    Cytosol Bem1(PB1!1),Cdc24(PB1!1,PhosphSites~u) @ 0.000005
'Bem1cyto_Unbinds_Cdc24cyto Ref1' :Cytosol Bem1(PB1!1),Cdc24(PB1!1,PhosphSites~u
    ) -> :Cytosol Bem1(PB1),Cdc24(PB1,PhosphSites~u) @ 0.1
'Cla4_Phosph_Cdc24_membr' :Membrane Cla4(X~a,PRR!1),Bem1(PB1!2,SH3b!1),Cdc24(PB1
    !2,PhosphSites~u) -> :Membrane Cla4(X~a,PRR!1),Bem1(PB1!2,SH3b!1),Cdc24(PB1
    !2,PhosphSites~p) @ 1.0
'Cla4_Phosph_Cdc24_cyto' :Cytosol Bem1(SH3b!1,PB1!2),Cla4(X~a,PRR!1),Cdc24(PB1
    !2,PhosphSites~u) -> :Cytosol Bem1(SH3b!1,PB1!2),Cla4(X~a,PRR!1),Cdc24(PB1
    !2,PhosphSites~p) @ 1.0
'Cdc24Phomembr_Diss_Bem1membr' :Membrane Bem1(PB1!1),Cdc24(PB1!1,PhosphSites~p)
    -> :Membrane Bem1(PB1),Cdc24(PB1,PhosphSites~p) @ 10.0
'Cdc24Phocyto_Diss_Bem1cyto' :Cytosol Bem1(PB1!1),Cdc24(PB1!1,PhosphSites~p) ->
    :Cytosol Bem1(PB1),Cdc24(PB1,PhosphSites~p) @ 10.0
'Ste11ppp_U_Membr' :Membrane Ste11(Ser281Ser285Thr286~ppp) -> :Membrane Ste11(
    Ser281Ser285Thr286~pp) @ 0.001
'Ste11ppp_U_Cyto' :Cytosol Ste11(Ser281Ser285Thr286~ppp) -> :Cytosol Ste11(
    Ser281Ser285Thr286~pp) @ 0.001
'Ste11pp_U_Membr' :Membrane Ste11(Ser281Ser285Thr286~pp) -> :Membrane Ste11(
    Ser281Ser285Thr286~p) @ 0.0005
'Ste11pp_U_Cyto' :Cytosol Ste11(Ser281Ser285Thr286~pp) -> :Cytosol Ste11(
    Ser281Ser285Thr286~p) @ 0.0005
```

```
'Ste11p_U_Membr' :Membrane Ste11(Ser281Ser285Thr286~p) -> :Membrane Ste11(
    Ser281Ser285Thr286~u) @ 0.0005
'Ste11p_U_Cyto' :Cytosol Ste11(Ser281Ser285Thr286~p) -> :Cytosol Ste11(
    Ser281Ser285Thr286~u) @ 0.0005
'Cdc24pmembr_u' :Membrane Cdc24(PhosphSites~p) -> :Membrane Cdc24(PhosphSites~u)
     @ 0.0005
'Cdc24pcyto_u' :Cytosol Cdc24(PhosphSites~p) -> :Cytosol Cdc24(PhosphSites~u) @
    0.0005
'Hog1_Binds_Sho1 Ref1' Sho1(Hog1,X~a),Hog1(Tyr176~p,Phosphat,Kinase,Sho1,Ptc23,
    Ste50,Thr174~p,TF,CD) -> Sho1(Hog1!1,X~a),Hog1(Tyr176~p,Phosphat,Kinase,Sho1
    !1:Memb_Cyto,Ptc23,Ste50,Thr174~p,TF,CD) @ 0.000005
'Hog1_Unbinds_Sho1 Ref1' Sho1(Hog1!1,X~a),Hog1(Tyr176~p,Phosphat,Kinase,Sho1!1:
    Memb_Cyto,Ptc23,Ste50,Thr174~p,TF,CD) -> Sho1(Hog1,X~a),Hog1(Tyr176~p,
    Phosphat,Kinase,Sho1,Ptc23,Ste50,Thr174~p,TF,CD) @ 1.0
'Hog1_Phosph_Sho1' Sho1(Hog1~u!1,X~a),Hog1(Tyr176~p,Phosphat,Kinase,Sho1!1:
    Memb_Cyto,Ptc23,Ste50,Thr174~p,TF,CD) -> Sho1(Hog1~p!1,X~a),Hog1(Tyr176~p,
    Phosphat,Kinase,Sho1!1:Memb_Cyto,Ptc23,Ste50,Thr174~p,TF,CD) @ 1.0
'Sho1_Dephosph' :Membrane Sho1(Hog1~p) -> :Membrane Sho1(Hog1~u) @ 0.0005

# Cdc42 Interaction
'Msb2_Unbinds_Cdc42' :Membrane Msb2(Cyt_Cdc42!1),Cdc42(Msb2!1) -> :Membrane Msb2
    (Cyt_Cdc42),Cdc42(Msb2) @ 0.1
'Msb2_Binds_Cdc42 Ref1' :Membrane Msb2(Cyt_Cdc42,X~a),Cdc42(Msb2) -> :Membrane
    Msb2(Cyt_Cdc42!1,X~a),Cdc42(Msb2!1) @ 0.00005
'Msb2_Unbinds_Cdc42 Ref1' :Membrane Msb2(Cyt_Cdc42!1,X~a),Cdc42(Msb2!1) -> :
    Membrane Msb2(Cyt_Cdc42,X~a),Cdc42(Msb2) @ 0.1
'Cdc24membr_Unbinds_Cdc42' :Membrane Cdc42(GEF!1),Cdc24(DH!1) -> :Membrane Cdc42
    (GEF),Cdc24(DH) @ 0.1
'Cdc24membr_Binds_Cdc42 Ref1' :Membrane Cdc42(GEF,X~GDP),Cdc24(X~a,DH) -> :
    Membrane Cdc42(GEF!1,X~GDP),Cdc24(X~a,DH!1) @ 0.000005
'Cdc24membr_Unbinds_Cdc42 Ref1' :Membrane Cdc42(GEF!1,X~GDP),Cdc24(X~a,DH!1) ->
    :Membrane Cdc42(GEF,X~GDP),Cdc24(X~a,DH) @ 0.1
'Cdc24membr_Binds_Cdc42 Ref2' :Membrane Cdc42(GEF,X~GDP),Bem1(PB1!1),Cdc24(X~a,
    PB1!1,DH) -> :Membrane Cdc42(GEF!2,X~GDP),Bem1(PB1!1),Cdc24(X~a,PB1!1,DH!2)
    @ 0.0005
'Cdc24membr_Unbinds_Cdc42 Ref2' :Membrane Cdc42(GEF!2,X~GDP),Bem1(PB1!1),Cdc24(X
    ~a,PB1!1,DH!2) -> :Membrane Cdc42(GEF,X~GDP),Bem1(PB1!1),Cdc24(X~a,PB1!1,DH)
     @ 0.1
'Cdc24membr_Unbinds_Cdc42 Ref3' :Membrane Cdc42(GEF!1,X~GTP),Cdc24(DH!1) -> :
    Membrane Cdc42(GEF,X~GTP),Cdc24(DH) @ 10.0
'Cdc24membr_Activates_Cdc42' :Membrane Cdc42(GEF!1,X~GDP),Cdc24(X~a,DH!1) -> :
    Membrane Cdc42(GEF!1,X~GTP),Cdc24(X~a,DH!1) @ 10.0
```

```
'Bem1membr_Unbinds_Cdc42GTP' :Membrane Cdc42(Bem1!1),Bem1(CI!1) -> :Membrane
    Cdc42(Bem1),Bem1(CI) @ 0.1
'Bem1membr_Binds_Cdc42GTP Ref1' :Membrane Bem1(CI),Cdc42(Bem1,ToCRIB,X~GTP) -> :
    Membrane Bem1(CI!1),Cdc42(Bem1!1,ToCRIB,X~GTP) @ 0.000005
'Bem1membr_Unbinds_Cdc42GTP Ref1' :Membrane Bem1(CI!1),Cdc42(Bem1!1,ToCRIB,X~GTP
    ) -> :Membrane Bem1(CI),Cdc42(Bem1,ToCRIB,X~GTP) @ 0.1
'Bem1_Scaffolds_Cdc24Cdc42' :Membrane Bem1(CI,PB1!1),Cdc42(GEF!2,Bem1,X~GTP),
    Cdc24(PB1!1,DH!2) -> :Membrane Bem1(CI!1,PB1!2),Cdc42(GEF,Bem1!1,X~GTP),
    Cdc24(PB1!2,DH) @ 5.0
'Cdc42_Activates_Ste20membrViaBem1_I' :Membrane Cdc42(ToCRIB,Bem1!1,X~GTP),Bem1(
    CI!1,SH3b!2),Ste20(CRIB,PRR!2,X~ia) -> :Membrane Cdc42(ToCRIB!3,Bem1!1,X~GTP
    ),Bem1(CI!1,SH3b!2),Ste20(CRIB!3,PRR!2,X~ia) @ 50.0
'Cdc42_Activates_Ste20membrViaBem1_II' :Membrane Bem1(SH3b!1,CI!2),Ste20(CRIB!3,
    PRR!1,X~ia),Cdc42(Bem1!2,ToCRIB!3,X~GTP) -> :Membrane Bem1(SH3b!3,CI),Ste20(
    CRIB!2,PRR!3,X~a),Cdc42(Bem1,ToCRIB!2,X~GTP) @ 100.0
'Cdc42_Unbinds_Ste20membr' :Membrane Cdc42(ToCRIB!1),Ste20(CRIB!1) -> :Membrane
    Cdc42(ToCRIB),Ste20(CRIB) @ 0.01
'Cdc42_Binds_Ste20membr Ref2' :Membrane Cdc42(ToCRIB,X~GTP),Ste20(CRIB) -> :
    Membrane Cdc42(ToCRIB!1,X~GTP),Ste20(CRIB!1) @ 0.0005
'Cdc42_Unbinds_Ste20membr Ref2' :Membrane Cdc42(ToCRIB!1,X~GTP),Ste20(CRIB!1) ->
     :Membrane Cdc42(ToCRIB,X~GTP),Ste20(CRIB) @ 0.01
'Cdc42_Activates_Ste20membr' :Membrane Cdc42(ToCRIB!1,X~GTP),Ste20(X~ia,CRIB!1)
    -> :Membrane Cdc42(ToCRIB!1,X~GTP),Ste20(X~a,CRIB!1) @ 10.0
'Ste50membr_Unbinds_Cdc42' :Membrane Ste50(RA!1),Cdc42(Ste50!1) -> :Membrane
    Ste50(RA),Cdc42(Ste50) @ 0.1
'Ste50membr_Binds_Cdc42 Ref4' :Membrane Ste50(RA),Cdc42(Ste50) -> :Membrane
    Ste50(RA!1),Cdc42(Ste50!1) @ 0.0005
'Ste50membr_Unbinds_Cdc42 Ref4' :Membrane Ste50(RA!1),Cdc42(Ste50!1) -> :
    Membrane Ste50(RA),Cdc42(Ste50) @ 0.1
'Ste50membr_Binds_Cdc42 Ref5' :Membrane Ste50(RA),Cdc42(Ste50,Msb2!1),Msb2(
    Cyt_Cdc42!1) -> :Membrane Ste50(RA!2),Cdc42(Ste50!2,Msb2!1),Msb2(Cyt_Cdc42
    !1) @ 0.0005
'Ste50membr_Unbinds_Cdc42 Ref5' :Membrane Ste50(RA!2),Cdc42(Ste50!2,Msb2!1),Msb2
    (Cyt_Cdc42!1) -> :Membrane Ste50(RA),Cdc42(Ste50,Msb2!1),Msb2(Cyt_Cdc42!1) @
     1.0
'Ste20_Phosph_Ste11' Ste20(CRIB!1,X~a),Cdc42(ToCRIB!1,Ste50!2,X~GTP),Ste50(SAM
    !3,RA!2),Ste11(SAM!3,Ser281Ser285Thr286~u) -> Ste20(CRIB!1,X~a),Cdc42(ToCRIB
    !1,Ste50!2,X~GTP),Ste50(SAM!3,RA!2),Ste11(SAM!3,Ser281Ser285Thr286~p) @ 10.0
'Ste20_Phosph_Ste11p' Ste20(CRIB!1,X~a),Cdc42(ToCRIB!1,Ste50!2,X~GTP),Ste50(SAM
    !3,RA!2),Ste11(SAM!3,Ser281Ser285Thr286~p) -> Ste20(CRIB!1,X~a),Cdc42(ToCRIB
    !1,Ste50!2,X~GTP),Ste50(SAM!3,RA!2),Ste11(SAM!3,Ser281Ser285Thr286~pp) @
    10.0
```

```
'Ste20_Phosph_Ste11pp' Ste20(CRIB!1,X~a),Cdc42(ToCRIB!1,Ste50!2,X~GTP),Ste50(SAM
    !3,RA!2),Ste11(SAM!3,Ser281Ser285Thr286~pp) -> Ste20(CRIB!1,X~a),Cdc42(
    ToCRIB!1,Ste50!2,X~GTP),Ste50(SAM!3,RA!2),Ste11(SAM!3,Ser281Ser285Thr286~ppp
    ) @ 10.0
'Ste20_Phosph_Ste11/Oligo' Ste20(CRIB!1,X~a),Cdc42(ToCRIB!1,Ste50!2,X~GTP),Ste50
    (SAMoligo!3,RA!2),Ste11(SAM!3,Ser281Ser285Thr286~u) -> Ste20(CRIB!1,X~a),
    Cdc42(ToCRIB!1,Ste50!2,X~GTP),Ste50(SAMoligo!3,RA!2),Ste11(SAM!3,
    Ser281Ser285Thr286~p) @ 10.0
'Ste20_Phosph_Ste11p/Oligo' Ste20(CRIB!1,X~a),Cdc42(ToCRIB!1,Ste50!2,X~GTP),
    Ste50(SAMoligo!3,RA!2),Ste11(SAM!3,Ser281Ser285Thr286~p) -> Ste20(CRIB!1,X~a
    ),Cdc42(ToCRIB!1,Ste50!2,X~GTP),Ste50(SAMoligo!3,RA!2),Ste11(SAM!3,
    Ser281Ser285Thr286~pp) @ 10.0
'Ste20_Phosph_Ste11pp/Oligo' Ste20(CRIB!1,X~a),Cdc42(ToCRIB!1,Ste50!2,X~GTP),
    Ste50(SAMoligo!3,RA!2),Ste11(SAM!3,Ser281Ser285Thr286~pp) -> Ste20(CRIB!1,X~
    a),Cdc42(ToCRIB!1,Ste50!2,X~GTP),Ste50(SAMoligo!3,RA!2),Ste11(SAM!3,
    Ser281Ser285Thr286~ppp) @ 10.0
'Cdc42_Binds_Cla4membr' :Membrane Cdc42(ToCRIB),Cla4(CRIB) -> :Membrane Cdc42(
    ToCRIB!1),Cla4(CRIB!1) @ 0.0005
'Cdc42_Binds_Cla4membr Ref1' :Membrane Cla4(CRIB),Cdc42(ToCRIB,X~GTP) -> :
    Membrane Cla4(CRIB!1),Cdc42(ToCRIB!1,X~GTP) @ 0.0005
'Cdc42_Unbinds_Cla4membr Ref1' :Membrane Cla4(CRIB!1),Cdc42(ToCRIB!1,X~GTP) -> :
    Membrane Cla4(CRIB),Cdc42(ToCRIB,X~GTP) @ 0.01
'Cdc42_Activates_Cla4membr' :Membrane Cdc42(ToCRIB!1,X~GTP),Cla4(CRIB!1,X~ia) ->
     :Membrane Cdc42(ToCRIB!1,X~GTP),Cla4(CRIB!1,X~a) @ 10.0
'Cla4_Phosph_Ste11' Cla4(CRIB!1,X~a),Cdc42(ToCRIB!1,Ste50!2,X~GTP),Ste50(SAM!3,
    RA!2),Ste11(SAM!3,Ser281Ser285Thr286~u) -> Cla4(CRIB!1,X~a),Cdc42(ToCRIB!1,
    Ste50!2,X~GTP),Ste50(SAM!3,RA!2),Ste11(SAM!3,Ser281Ser285Thr286~p) @ 10.0
'Cla4_Phosph_Ste11p' Cla4(CRIB!1,X~a),Cdc42(ToCRIB!1,Ste50!2,X~GTP),Ste50(SAM!3,
    RA!2),Ste11(SAM!3,Ser281Ser285Thr286~p) -> Cla4(CRIB!1,X~a),Cdc42(ToCRIB!1,
    Ste50!2,X~GTP),Ste50(SAM!3,RA!2),Ste11(SAM!3,Ser281Ser285Thr286~pp) @ 10.0
'Cla4_Phosph_Ste11pp' Cla4(CRIB!1,X~a),Cdc42(ToCRIB!1,Ste50!2,X~GTP),Ste50(SAM
    !3,RA!2),Ste11(SAM!3,Ser281Ser285Thr286~pp) -> Cla4(CRIB!1,X~a),Cdc42(ToCRIB
    !1,Ste50!2,X~GTP),Ste50(SAM!3,RA!2),Ste11(SAM!3,Ser281Ser285Thr286~ppp) @
    10.0
'Cla4_Phosph_Ste11/Oligo' Cla4(CRIB!1,X~a),Cdc42(ToCRIB!1,Ste50!2,X~GTP),Ste50(
    SAMoligo!3,RA!2),Ste11(SAM!3,Ser281Ser285Thr286~u) -> Cla4(CRIB!1,X~a),Cdc42
    (ToCRIB!1,Ste50!2,X~GTP),Ste50(SAMoligo!3,RA!2),Ste11(SAM!3,
    Ser281Ser285Thr286~p) @ 10.0
'Cla4_Phosph_Ste11p/Oligo' Cla4(CRIB!1,X~a),Cdc42(ToCRIB!1,Ste50!2,X~GTP),Ste50(
    SAMoligo!3,RA!2),Ste11(SAM!3,Ser281Ser285Thr286~p) -> Cla4(CRIB!1,X~a),Cdc42
    (ToCRIB!1,Ste50!2,X~GTP),Ste50(SAMoligo!3,RA!2),Ste11(SAM!3,
    Ser281Ser285Thr286~pp) @ 10.0
```

```
'Cla4_Phosph_Ste11pp/Oligo' Cla4(CRIB!1,X~a),Cdc42(ToCRIB!1,Ste50!2,X~GTP),Ste50
    (SAMoligo!3,RA!2),Ste11(SAM!3,Ser281Ser285Thr286~pp) -> Cla4(CRIB!1,X~a),
    Cdc42(ToCRIB!1,Ste50!2,X~GTP),Ste50(SAMoligo!3,RA!2),Ste11(SAM!3,
    Ser281Ser285Thr286~ppp) @ 10.0
'Cdc42_Deactivation' Cdc42(X~GTP) -> Cdc42(X~GDP) @ 0.005

# Sho1 Interaction
'Sho1IA_Unbinds_Pbs2' :Membrane Pbs2(sho91_102!1),Sho1(SH3_338!1,X~ia) -> :
    Membrane Pbs2(sho91_102),Sho1(SH3_338,X~ia) @ 0.1
'Sho1_UnbindsPbs2uu' :Membrane Pbs2(Thr518,Ser514,sho91_102!1),Sho1(SH3_338!1,X)
     -> :Membrane Pbs2(Thr518,Ser514,sho91_102),Sho1(SH3_338,X) @ 0.0001
'Sho1A_BindsPbs2uu Ref1' :Membrane Sho1(SH3_338,X~a),Pbs2(Thr518~u,Ser514~u,
    Ste11,sho91_102) -> :Membrane Sho1(SH3_338!1,X~a),Pbs2(Thr518~u,Ser514~u,
    Ste11,sho91_102!1) @ 0.0005
'Sho1A_UnbindsPbs2uu Ref1' :Membrane Sho1(SH3_338!1,X~a),Pbs2(Thr518~u,Ser514~u,
    Ste11,sho91_102!1) -> :Membrane Sho1(SH3_338,X~a),Pbs2(Thr518~u,Ser514~u,
    Ste11,sho91_102) @ 0.1
'Sho1A_UnbindsPbs2uu Ref2' :Membrane Pbs2(Thr518~p,Ser514~p,Ste11,sho91_102!1),
    Sho1(SH3_338!1,X~a) -> :Membrane Pbs2(Thr518~p,Ser514~p,Ste11,sho91_102),
    Sho1(SH3_338,X~a) @ 1.0
'Sho1A_BindsPbs2uu Ref3' :Membrane Pbs2(Ste11,sho91_102,Ser514~u,Thr518~u),Sho1(
    SH3_338,X~ia) -> :Membrane Pbs2(Ste11,sho91_102!1,Ser514~u,Thr518~u),Sho1(
    SH3_338!1,X~ia) @ 0.0005
'Sho1A_UnbindsPbs2uu Ref3' :Membrane Pbs2(Ste11,sho91_102!1,Ser514~u,Thr518~u),
    Sho1(SH3_338!1,X~ia) -> :Membrane Pbs2(Ste11,sho91_102,Ser514~u,Thr518~u),
    Sho1(SH3_338,X~ia) @ 0.1
'Sho1A_BindsPbs2uu Ref4' :Membrane Pbs2(Ste11,sho91_102,Ser514~u,Thr518~p),Sho1(
    SH3_338,X) -> :Membrane Pbs2(Ste11,sho91_102!1,Ser514~u,Thr518~p),Sho1(
    SH3_338!1,X) @ 0.0005
'Sho1A_UnbindsPbs2uu Ref4' :Membrane Pbs2(Ste11,sho91_102!1,Ser514~u,Thr518~p),
    Sho1(SH3_338!1,X) -> :Membrane Pbs2(Ste11,sho91_102,Ser514~u,Thr518~p),Sho1(
    SH3_338,X) @ 0.1
'Sho1A_BindsPbs2uu Ref5' :Membrane Pbs2(Ste11,sho91_102,Ser514~p,Thr518~u),Sho1(
    SH3_338,X) -> :Membrane Pbs2(Ste11,sho91_102!1,Ser514~p,Thr518~u),Sho1(
    SH3_338!1,X) @ 0.0005
'Sho1A_UnbindsPbs2uu Ref5' :Membrane Pbs2(Ste11,sho91_102!1,Ser514~p,Thr518~u),
    Sho1(SH3_338!1,X) -> :Membrane Pbs2(Ste11,sho91_102,Ser514~p,Thr518~u),Sho1(
    SH3_338,X) @ 0.1
'T_Ste11membr_Unbinds_Sho1' :Membrane Sho1(SH3_342_346!1),Ste11(Sho1!1) -> :
    Membrane Sho1(SH3_342_346),Ste11(Sho1) @ 0.1
'T_Ste11membr_Binds_Sho1 Ref2' :Membrane Sho1(SH3_342_346,X~a),Ste11(Sho1,
    Ser281Ser285Thr286~ppp,SAM!_) -> :Membrane Sho1(SH3_342_346!1,X~a),Ste11(
```

```
    Sho1!1,Ser281Ser285Thr286~ppp,SAM!_) @ 0.0005
'T_Ste11membr_Unbinds_Sho1 Ref2' :Membrane Sho1(SH3_342_346!1,X~a),Ste11(Sho1!1,
    Ser281Ser285Thr286~ppp,SAM!_) -> :Membrane Sho1(SH3_342_346,X~a),Ste11(Sho1,
    Ser281Ser285Thr286~ppp,SAM!_) @ 0.1
'T2_Ste50_Unbinds_Sho1' :Membrane Ste50(Sho1!1),Sho1(Ste50!1) -> :Membrane Ste50
    (Sho1),Sho1(Ste50) @ 0.1
'T2_Ste50_Unbinds_Sho1 Ref2' :Membrane Sho1(Ste50!1,X),Ste50(Sho1!1,SAM) -> :
    Membrane Sho1(Ste50,X),Ste50(Sho1,SAM) @ 1.0
'T2_Ste50_Unbinds_Sho1 Ref3' :Membrane Sho1(Ste50!1,X),Ste50(Sho1!1,SAMoligo) ->
     :Membrane Sho1(Ste50,X),Ste50(Sho1,SAMoligo) @ 1.0
'T2_Ste50_Unbinds_Sho1 Ref4' :Membrane Ste50(Sho1!1),Sho1(Ste50!1,X~ia) -> :
    Membrane Ste50(Sho1),Sho1(Ste50,X~ia) @ 1.0
'Ste11Sam_Attach_Ste50_Sho1' :Membrane Sho1(Ste50,SH3_342_346!2,X~a),Ste11(Sho1
    !2,SAM!1),Ste50(Sho1,SAM!1) -> :Membrane Sho1(Ste50!1,SH3_342_346!2,X~a),
    Ste11(Sho1!2,SAM!3),Ste50(Sho1!1,SAM!3) @ 20.0
'Ste11OligSam_Attach_Ste50_Sho1' :Membrane Sho1(Ste50,SH3_342_346!2,X~a),Ste11(
    Sho1!2,SAM!3),Ste50(Sho1,SAMoligo!3) -> :Membrane Sho1(Ste50!1,SH3_342_346
    !2,X~a),Ste11(Sho1!2,SAM!3),Ste50(Sho1!1,SAMoligo!3) @ 20.0
'Sho1_Brks_Ste50Ste11Sam' :Membrane Sho1(Ste50!1,SH3_342_346!2,X),Ste11(Sho1!2,
    SAM!3),Ste50(Sho1!1,SAM!3) -> :Membrane Sho1(Ste50!1,SH3_342_346!2,X),Ste11(
    Sho1!2,SAM),Ste50(Sho1!1,SAM) @ 50.0
'Sho1_Brks_Ste50Ste11OligSam' :Membrane Sho1(Ste50!1,SH3_342_346!2,X),Ste11(Sho1
    !2,SAM!3),Ste50(Sho1!1,SAMoligo!3) -> :Membrane Sho1(Ste50!1,SH3_342_346!2,X
    ),Ste11(Sho1!2,SAM),Ste50(Sho1!1,SAMoligo) @ 50.0
'Sho1_QarternComplxSam' :Membrane Sho1(SH3_342_346!2,Ste50,SH3_338!4,X),Ste11(
    Sho1!2,Pbs2),Pbs2(sho91_102!4,Ste11) -> :Membrane Sho1(SH3_342_346!2,Ste50,
    SH3_338!4,X),Ste11(Sho1!2,Pbs2!5),Pbs2(sho91_102!4,Ste11!5) @ 1.0
'Ste11Pbs2_Unbind_Sho1' :Membrane Sho1(SH3_342_346!2,SH3_338!4,X),Pbs2(sho91_102
    !4,Ste11!5),Ste11(Sho1!2,Pbs2!5) -> :Membrane Sho1(SH3_342_346,SH3_338!4,X),
    Pbs2(sho91_102!4,Ste11!1),Ste11(Sho1,Pbs2!1) @ 50.0
'Sho1_Unbind_Pbs2Ste11' Sho1(SH3_338!1),Pbs2(sho91_102!1,Ste11!2),Ste11(Pbs2!2)
    -> Sho1(SH3_338),Pbs2(sho91_102,Ste11!1),Ste11(Pbs2!1) @ 1.0

# Opy2 Interaction
'Ste50membr_Unbinds_Opy2CRA' :Membrane Opy2(CR-A!1),Ste50(RA!1,S155_196_202_248)
     -> :Membrane Opy2(CR-A),Ste50(RA,S155_196_202_248) @ 0.01
'Ste50membr_Binds_Opy2CRA Ref1' :Membrane Opy2(CR-A),Ste50(RA,S155_196_202_248~u
    ) -> :Membrane Opy2(CR-A!1),Ste50(RA!1,S155_196_202_248~u) @ 0.00005
'Ste50membr_Unbinds_Opy2CRA Ref1' :Membrane Opy2(CR-A!1),Ste50(RA!1,
    S155_196_202_248~u) -> :Membrane Opy2(CR-A),Ste50(RA,S155_196_202_248~u) @
    0.01
'Ste50pmembr_Binds_Opy2CRA' :Membrane Opy2(CR-A),Ste50(RA,S155_196_202_248~p) ->
```

```
     :Membrane Opy2(CR-A!1),Ste50(RA!1,S155_196_202_248~p) @ 0.0000005
'Ste50pmembr_Unbinds_Opy2CRA' :Membrane Opy2(CR-A!1),Ste50(RA!1,S155_196_202_248
    ~p) -> :Membrane Opy2(CR-A),Ste50(RA,S155_196_202_248~p) @ 0.1
'Ste50ppmembr_Binds_Opy2CRA' :Membrane Opy2(CR-A),Ste50(RA,S155_196_202_248~pp)
    -> :Membrane Opy2(CR-A!1),Ste50(RA!1,S155_196_202_248~pp) @ 0.00000005
'Ste50ppmembr_Unbinds_Opy2CRA' :Membrane Opy2(CR-A!1),Ste50(RA!1,
    S155_196_202_248~pp) -> :Membrane Opy2(CR-A),Ste50(RA,S155_196_202_248~pp) @
     0.1
'Ste50pppmembr_Binds_Opy2CRA' :Membrane Opy2(CR-A),Ste50(RA,S155_196_202_248~ppp
    ) -> :Membrane Opy2(CR-A!1),Ste50(RA!1,S155_196_202_248~ppp) @ 0.000000005
'Ste50pppmembr_Unbinds_Opy2CRA' :Membrane Opy2(CR-A!1),Ste50(RA!1,
    S155_196_202_248~ppp) -> :Membrane Opy2(CR-A),Ste50(RA,S155_196_202_248~ppp)
     @ 0.1
'Ste50ppppmembr_Unbinds_Opy2CRA' :Membrane Opy2(CR-A!1),Ste50(RA!1,
    S155_196_202_248~pppp) -> :Membrane Opy2(CR-A),Ste50(RA,S155_196_202_248~
    pppp) @ 1.0
'Ste50membr_Unbinds_Opy2CRB' :Membrane Opy2(CR-B!1),Ste50(RA!1,S155_196_202_248)
     -> :Membrane Opy2(CR-B),Ste50(RA,S155_196_202_248) @ 0.01
'Ste50membr_Binds_Opy2CRB Ref1' :Membrane Opy2(CR-B~p),Ste50(RA,S155_196_202_248
    ~u) -> :Membrane Opy2(CR-B~p!1),Ste50(RA!1,S155_196_202_248~u) @ 0.00005
'Ste50membr_Unbinds_Opy2CRB Ref1' :Membrane Opy2(CR-B~p!1),Ste50(RA!1,
    S155_196_202_248~u) -> :Membrane Opy2(CR-B~p),Ste50(RA,S155_196_202_248~u) @
     0.01
'Ste50pmembr_Binds_Opy2CRB' :Membrane Opy2(CR-B~p),Ste50(RA,S155_196_202_248~p)
    -> :Membrane Opy2(CR-B~p!1),Ste50(RA!1,S155_196_202_248~p) @ 0.0000005
'Ste50pmembr_Unbinds_Opy2CRB' :Membrane Opy2(CR-B~p!1),Ste50(RA!1,
    S155_196_202_248~p) -> :Membrane Opy2(CR-B~p),Ste50(RA,S155_196_202_248~p) @
     0.1
'Ste50ppmembr_Binds_Opy2CRB' :Membrane Opy2(CR-B~p),Ste50(RA,S155_196_202_248~pp
    ) -> :Membrane Opy2(CR-B~p!1),Ste50(RA!1,S155_196_202_248~pp) @ 0.00000005
'Ste50ppmembr_Unbinds_Opy2CRB' :Membrane Opy2(CR-B~p!1),Ste50(RA!1,
    S155_196_202_248~pp) -> :Membrane Opy2(CR-B~p),Ste50(RA,S155_196_202_248~pp)
     @ 0.1
'Ste50pppmembr_Binds_Opy2CRB' :Membrane Opy2(CR-B~p),Ste50(RA,S155_196_202_248~
    ppp) -> :Membrane Opy2(CR-B~p!1),Ste50(RA!1,S155_196_202_248~ppp) @
    0.000000005
'Ste50pppmembr_Unbinds_Opy2CRB' :Membrane Opy2(CR-B~p!1),Ste50(RA!1,
    S155_196_202_248~ppp) -> :Membrane Opy2(CR-B~p),Ste50(RA,S155_196_202_248~
    ppp) @ 0.1
'Ste50ppppmembr_Unbinds_Opy2CRB' :Membrane Opy2(CR-B~p!1),Ste50(RA!1,
    S155_196_202_248~pppp) -> :Membrane Opy2(CR-B~p),Ste50(RA,S155_196_202_248~
    pppp) @ 1.0
```

```
'Ste50membr_Unbinds_Opy2CRD' :Membrane Opy2(CR-D!1),Ste50(RA!1,S155_196_202_248)
     -> :Membrane Opy2(CR-D),Ste50(RA,S155_196_202_248) @ 0.01
'Ste50membr_Binds_Opy2CRD Ref1' :Membrane Opy2(CR-D),Ste50(RA,S155_196_202_248~u
    ) -> :Membrane Opy2(CR-D!1),Ste50(RA!1,S155_196_202_248~u) @ 0.00005
'Ste50membr_Unbinds_Opy2CRD Ref1' :Membrane Opy2(CR-D!1),Ste50(RA!1,
    S155_196_202_248~u) -> :Membrane Opy2(CR-D),Ste50(RA,S155_196_202_248~u) @
    0.01
'Ste50pmembr_Binds_Opy2CRD' :Membrane Opy2(CR-D),Ste50(RA,S155_196_202_248~p) ->
     :Membrane Opy2(CR-D!1),Ste50(RA!1,S155_196_202_248~p) @ 0.0000005
'Ste50pmembr_Unbinds_Opy2CRD' :Membrane Opy2(CR-D!1),Ste50(RA!1,S155_196_202_248
    ~p) -> :Membrane Opy2(CR-D),Ste50(RA,S155_196_202_248~p) @ 0.1
'Ste50ppmembr_Binds_Opy2CRD' :Membrane Opy2(CR-D),Ste50(RA,S155_196_202_248~pp)
    -> :Membrane Opy2(CR-D!1),Ste50(RA!1,S155_196_202_248~pp) @ 0.00000005
'Ste50ppmembr_Unbinds_Opy2CRD' :Membrane Opy2(CR-D!1),Ste50(RA!1,
    S155_196_202_248~pp) -> :Membrane Opy2(CR-D),Ste50(RA,S155_196_202_248~pp) @
     0.1
'Ste50pppmembr_Binds_Opy2CRD' :Membrane Opy2(CR-D),Ste50(RA,S155_196_202_248~ppp
    ) -> :Membrane Opy2(CR-D!1),Ste50(RA!1,S155_196_202_248~ppp) @ 0.000000005
'Ste50pppmembr_Unbinds_Opy2CRD' :Membrane Opy2(CR-D!1),Ste50(RA!1,
    S155_196_202_248~ppp) -> :Membrane Opy2(CR-D),Ste50(RA,S155_196_202_248~ppp)
     @ 0.1
'Ste50ppppmembr_Unbinds_Opy2CRD' :Membrane Opy2(CR-D!1),Ste50(RA!1,
    S155_196_202_248~pppp) -> :Membrane Opy2(CR-D),Ste50(RA,S155_196_202_248~
    pppp) @ 1.0
'Yck1_2_Binds_Opy2' :Membrane Yck1_2(Opy2),Opy2(Yck) -> :Membrane Yck1_2(Opy2!1)
    ,Opy2(Yck!1) @ 0.000005
'Yck1_2_Unbinds_Opy2' :Membrane Yck1_2(Opy2!1),Opy2(Yck!1) -> :Membrane Yck1_2(
    Opy2),Opy2(Yck) @ 0.1
'Yck1_2_Phosph_Opy2CRB' :Membrane Opy2(CR-B~u,Yck!1),Yck1_2(Opy2!1,A~a) -> :
    Membrane Opy2(CR-B~p,Yck!1),Yck1_2(Opy2!1,A~a) @ 10.0

# Ste50Ste11OligDim
'2Ste50Dim2_Oligomerization1' :Cytosol Ste50(SAMoligo,SAM!1),Ste50(SAM,SAMoligo
    !2),Ste50(SAM!1),Ste50(SAMoligo!2) -> :Cytosol Ste50(SAMoligo!3,SAM!1),Ste50
    (SAM!3,SAMoligo!2),Ste50(SAM!1),Ste50(SAMoligo!2) @ 0.0000001
'2Ste50Dim2_Unoligomerization1' :Cytosol Ste50(SAMoligo!3,SAM!1),Ste50(SAM!3,
    SAMoligo!2),Ste50(SAM!1),Ste50(SAMoligo!2) -> :Cytosol Ste50(SAMoligo,SAM!1)
    ,Ste50(SAM,SAMoligo!2),Ste50(SAM!1),Ste50(SAMoligo!2) @ 0.0001
'2Ste50OligoDim_Oligomerization2' :Cytosol Ste50(SAM,SAMoligo!1),Ste50(SAM!1),
    Ste50(SAMoligo,SAM!2),Ste50(SAMoligo!2) -> :Cytosol Ste50(SAM!3,SAMoligo!1),
    Ste50(SAM!1),Ste50(SAMoligo!3,SAM!2),Ste50(SAMoligo!2) @ 0.0000001
'2Ste50OligoDim_Unoligomerization2' :Cytosol Ste50(SAM!3,SAMoligo!1),Ste50(SAM
```

```
    !1),Ste50(SAMoligo!3,SAM!2),Ste50(SAMoligo!2) -> :Cytosol Ste50(SAM,SAMoligo
    !1),Ste50(SAM!1),Ste50(SAMoligo,SAM!2),Ste50(SAMoligo!2) @ 0.0001
'2Ste50OligoDim_Oligomerization1' :Cytosol Ste50(SAM,SAMoligo!1),Ste50(SAM!1),
    Ste50(SAMoligo!2,SAM),Ste50(SAM!2) -> :Cytosol Ste50(SAM!3,SAMoligo!1),Ste50
    (SAM!1),Ste50(SAMoligo!2,SAM!3),Ste50(SAM!2) @ 0.0000001
'2Ste50OligoDim_Unoligomerization1' :Cytosol Ste50(SAM!3,SAMoligo!1),Ste50(SAM
    !1),Ste50(SAMoligo!2,SAM!3),Ste50(SAM!2) -> :Cytosol Ste50(SAM,SAMoligo!1),
    Ste50(SAM!1),Ste50(SAMoligo!2,SAM),Ste50(SAM!2) @ 0.0001
'2Ste50OligoDim2_Oligomerization2' :Cytosol Ste50(SAM,SAMoligo!1),Ste50(SAM!1),
    Ste50(SAMoligo,SAM!2),Ste50(SAM!2) -> :Cytosol Ste50(SAM!3,SAMoligo!1),Ste50
    (SAM!1),Ste50(SAMoligo!3,SAM!2),Ste50(SAM!2) @ 0.0000001
'2Ste50OligoDim2_Unoligomerization2' :Cytosol Ste50(SAM!3,SAMoligo!1),Ste50(SAM
    !1),Ste50(SAMoligo!3,SAM!2),Ste50(SAM!2) -> :Cytosol Ste50(SAM,SAMoligo!1),
    Ste50(SAM!1),Ste50(SAMoligo,SAM!2),Ste50(SAM!2) @ 0.0001
'2Ste50OligoDim2_Oligomerization1' :Cytosol Ste50(SAM,SAMoligo!1),Ste50(SAM!1),
    Ste50(SAMoligo!2,SAM),Ste50(SAMoligo!2) -> :Cytosol Ste50(SAM!3,SAMoligo!1),
    Ste50(SAM!1),Ste50(SAMoligo!2,SAM!3),Ste50(SAMoligo!2) @ 0.0000001
'2Ste50OligoDim2_Unoligomerization1' :Cytosol Ste50(SAM!3,SAMoligo!1),Ste50(SAM
    !1),Ste50(SAMoligo!2,SAM!3),Ste50(SAMoligo!2) -> :Cytosol Ste50(SAM,SAMoligo
    !1),Ste50(SAM!1),Ste50(SAMoligo!2,SAM),Ste50(SAMoligo!2) @ 0.0001
'2Ste50Dim_Oligomerization2' :Cytosol Ste50(SAM!1),Ste50(SAMoligo,SAM!2),Ste50(
    SAM!1,SAMoligo),Ste50(SAM!2) -> :Cytosol Ste50(SAM!1),Ste50(SAMoligo!3,SAM
    !2),Ste50(SAM!1,SAMoligo!3),Ste50(SAM!2) @ 0.0000001
'2Ste50Dim_Unoligomerization2' :Cytosol Ste50(SAM!1),Ste50(SAMoligo!3,SAM!2),
    Ste50(SAM!1,SAMoligo!3),Ste50(SAM!2) -> :Cytosol Ste50(SAM!1),Ste50(SAMoligo
    ,SAM!2),Ste50(SAM!1,SAMoligo),Ste50(SAM!2) @ 0.0001
'Ste50_Oligomerization2_new' :Cytosol Ste50(SAM!1,SAMoligo),Ste50(SAM!1),Ste50(
    SAMoligo,SAM) -> :Cytosol Ste50(SAM!1,SAMoligo!2),Ste50(SAM!1),Ste50(
    SAMoligo!2,SAM) @ 0.0000001
'Ste50_Unoligomerization2_new' :Cytosol Ste50(SAM!1,SAMoligo!2),Ste50(SAM!1),
    Ste50(SAMoligo!2,SAM) -> :Cytosol Ste50(SAM!1,SAMoligo),Ste50(SAM!1),Ste50(
    SAMoligo,SAM) @ 0.0001
'Ste50Oligo_Oligomerization2' :Cytosol Ste50(SAM,SAMoligo!1),Ste50(SAM!1),Ste50(
    SAMoligo,SAM) -> :Cytosol Ste50(SAM!2,SAMoligo!1),Ste50(SAM!1),Ste50(
    SAMoligo!2,SAM) @ 0.0000001
'Ste50Oligo_Unoligomerization2' :Cytosol Ste50(SAM!2,SAMoligo!1),Ste50(SAM!1),
    Ste50(SAMoligo!2,SAM) -> :Cytosol Ste50(SAM,SAMoligo!1),Ste50(SAM!1),Ste50(
    SAMoligo,SAM) @ 0.0001
'2Ste50Dim_Oligomerization1' :Cytosol Ste50(SAM!1,SAMoligo),Ste50(SAM,SAMoligo
    !2),Ste50(SAM!1),Ste50(SAM!2) -> :Cytosol Ste50(SAM!1,SAMoligo!3),Ste50(SAM
    !3,SAMoligo!2),Ste50(SAM!1),Ste50(SAM!2) @ 0.0000001
'2Ste50Dim_Unoligomerization1' :Cytosol Ste50(SAM!1,SAMoligo!3),Ste50(SAM!3,
```

```
    SAMoligo!2),Ste50(SAM!1),Ste50(SAM!2) -> :Cytosol Ste50(SAM!1,SAMoligo),
    Ste50(SAM,SAMoligo!2),Ste50(SAM!1),Ste50(SAM!2) @ 0.0001
'2Ste50Dim2_Oligomerization2' :Cytosol Ste50(SAMoligo,SAM!2),Ste50(SAMoligo,SAM
    !1),Ste50(SAM!1),Ste50(SAMoligo!2) -> :Cytosol Ste50(SAMoligo!3,SAM!2),Ste50
    (SAMoligo!3,SAM!1),Ste50(SAM!1),Ste50(SAMoligo!2) @ 0.0000001
'2Ste50Dim2_Unoligomerization2' :Cytosol Ste50(SAMoligo!3,SAM!2),Ste50(SAMoligo
    !3,SAM!1),Ste50(SAM!1),Ste50(SAMoligo!2) -> :Cytosol Ste50(SAMoligo,SAM!2),
    Ste50(SAMoligo,SAM!1),Ste50(SAM!1),Ste50(SAMoligo!2) @ 0.0001
'Ste50Oligo_Oligomerization1' :Cytosol Ste50(SAM,SAMoligo!1),Ste50(SAM!1),Ste50(
    SAMoligo,SAM) -> :Cytosol Ste50(SAM!2,SAMoligo!1),Ste50(SAM!1),Ste50(
    SAMoligo,SAM!2) @ 0.0000001
'Ste50Oligo_Unoligomerization1' :Cytosol Ste50(SAM!2,SAMoligo!1),Ste50(SAM!1),
    Ste50(SAMoligo,SAM!2) -> :Cytosol Ste50(SAM,SAMoligo!1),Ste50(SAM!1),Ste50(
    SAMoligo,SAM) @ 0.0001
'Ste50_Oligomerization1_new' :Cytosol Ste50(SAM!1),Ste50(SAM!1,SAMoligo),Ste50(
    SAM,SAMoligo) -> :Cytosol Ste50(SAM!1),Ste50(SAM!1,SAMoligo!2),Ste50(SAM!2,
    SAMoligo) @ 0.0000001
'Ste50_Unoligomerization1_new' :Cytosol Ste50(SAM!1),Ste50(SAM!1,SAMoligo!2),
    Ste50(SAM!2,SAMoligo) -> :Cytosol Ste50(SAM!1),Ste50(SAM!1,SAMoligo),Ste50(
    SAM,SAMoligo) @ 0.0001
'Ste50_Dimerization_new' :Cytosol Ste50(SAM,SAMoligo),Ste50(SAM,SAMoligo) -> :
    Cytosol Ste50(SAM!1,SAMoligo),Ste50(SAM!1,SAMoligo) @ 0.0000001
'Ste50_Undimerization_new' :Cytosol Ste50(SAM!1,SAMoligo),Ste50(SAM!1,SAMoligo)
    -> :Cytosol Ste50(SAM,SAMoligo),Ste50(SAM,SAMoligo) @ 0.0001
'Ste50_Dimerization_Oligo' :Cytosol Ste50(SAM,SAMoligo),Ste50(SAM,SAMoligo) -> :
    Cytosol Ste50(SAM!1,SAMoligo),Ste50(SAM,SAMoligo!1) @ 0.0000001
'Ste50_Undimerization_Oligo' :Cytosol Ste50(SAM!1,SAMoligo),Ste50(SAM,SAMoligo
    !1) -> :Cytosol Ste50(SAM,SAMoligo),Ste50(SAM,SAMoligo) @ 0.0001
'Ste50_Binds_Ste11_SAM' :Cytosol Ste11(SAM),Ste50(SAM,SAMoligo) -> :Cytosol
    Ste11(SAM!1),Ste50(SAM!1,SAMoligo) @ 0.000005
'Ste50_Unbinds_Ste11_SAM' :Cytosol Ste11(SAM!1),Ste50(SAM!1,SAMoligo) -> :
    Cytosol Ste11(SAM),Ste50(SAM,SAMoligo) @ 0.0001
'Ste50_Binds_Ste11_OligoSAM' :Cytosol Ste50(SAMoligo,SAM),Ste11(SAM) -> :Cytosol
     Ste50(SAMoligo!1,SAM),Ste11(SAM!1) @ 0.000005
'Ste50_Unbinds_Ste11_OligoSAM' :Cytosol Ste50(SAMoligo!1,SAM),Ste11(SAM!1) -> :
    Cytosol Ste50(SAMoligo,SAM),Ste11(SAM) @ 0.0001
'Ste11_Binds_Ste50Ste11' :Cytosol Ste11(SAM),Ste50(SAM,SAMoligo!1),Ste11(SAM!1)
    -> :Cytosol Ste11(SAM!2),Ste50(SAM!2,SAMoligo!1),Ste11(SAM!1) @ 0.000005
'Ste11_Unbinds_Ste50Ste11' :Cytosol Ste11(SAM!2),Ste50(SAM!2,SAMoligo!1),Ste11(
    SAM!1) -> :Cytosol Ste11(SAM),Ste50(SAM,SAMoligo!1),Ste11(SAM!1) @ 0.00001
'Ste11_Binds_Ste50Ste11_Oligo' :Cytosol Ste11(SAM!1),Ste50(SAMoligo,SAM!1),Ste11
    (SAM) -> :Cytosol Ste11(SAM!1),Ste50(SAMoligo!2,SAM!1),Ste11(SAM!2) @
```

```
    0.000005
'Ste11_Unbinds_Ste50Ste11_Oligo' :Cytosol Ste11(SAM!1),Ste50(SAMoligo!2,SAM!1),
    Ste11(SAM!2) -> :Cytosol Ste11(SAM!1),Ste50(SAMoligo,SAM!1),Ste11(SAM) @
    0.00001
'Ste50Oligmembr_Binds_Ste11membr' :Membrane Ste11(SAM),Ste50(SAMoligo,SAM) -> :
    Membrane Ste11(SAM!1),Ste50(SAMoligo!1,SAM) @ 0.00005
'Ste50Oligmembr_Unbinds_Ste11membr' :Membrane Ste11(SAM!1),Ste50(SAMoligo!1,SAM)
     -> :Membrane Ste11(SAM),Ste50(SAMoligo,SAM) @ 0.0001
'Ste50OligmembrOpyCRA_Unbinds_Ste11membr' :Membrane Opy2(CR-A!1),Ste11(SAM!2),
    Ste50(SAMoligo!2,RA!1) -> :Membrane Opy2(CR-A!1),Ste11(SAM),Ste50(SAMoligo,
    RA!1) @ 0.0001
'Ste50OligmembrOpyCRA_Binds_Ste11membr Ref1' :Membrane Opy2(CR-A!1),Ste50(
    SAMoligo,RA!1),Ste11(SAM,Pbs2,Sho1) -> :Membrane Opy2(CR-A!1),Ste50(SAMoligo
    !2,RA!1),Ste11(SAM!2,Pbs2,Sho1) @ 0.005
'Ste50OligmembrOpyCRA_Unbinds_Ste11membr Ref1' :Membrane Opy2(CR-A!1),Ste50(
    SAMoligo!2,RA!1),Ste11(SAM!2,Pbs2,Sho1) -> :Membrane Opy2(CR-A!1),Ste50(
    SAMoligo,RA!1),Ste11(SAM,Pbs2,Sho1) @ 0.0001
'Ste50OligmembrOpyCRB_Unbinds_Ste11membr' :Membrane Ste11(SAM!2),Ste50(SAMoligo
    !2,RA!1),Opy2(CR-B~p!1) -> :Membrane Ste11(SAM),Ste50(SAMoligo,RA!1),Opy2(CR
    -B~p!1) @ 0.0001
'Ste50OligmembrOpyCRB_Binds_Ste11membr Ref1' :Membrane Ste50(SAMoligo,RA!1),Opy2
    (CR-B~p!1),Ste11(SAM,Sho1,Pbs2) -> :Membrane Ste50(SAMoligo!2,RA!1),Opy2(CR-
    B~p!1),Ste11(SAM!2,Sho1,Pbs2) @ 0.005
'Ste50OligmembrOpyCRB_Unbinds_Ste11membr Ref1' :Membrane Ste50(SAMoligo!2,RA!1),
    Opy2(CR-B~p!1),Ste11(SAM!2,Sho1,Pbs2) -> :Membrane Ste50(SAMoligo,RA!1),Opy2
    (CR-B~p!1),Ste11(SAM,Sho1,Pbs2) @ 0.0001
'Ste50OligmembrOpyCRD_Unbinds_Ste11membr' :Membrane Ste11(SAM!2),Ste50(SAMoligo
    !2,RA!1),Opy2(CR-D!1) -> :Membrane Ste11(SAM),Ste50(SAMoligo,RA!1),Opy2(CR-D
    !1) @ 0.0001
'Ste50OligmembrOpyCRD_Binds_Ste11membr Ref1' :Membrane Ste50(SAMoligo,RA!1),Opy2
    (CR-D!1),Ste11(SAM,Sho1,Pbs2) -> :Membrane Ste50(SAMoligo!2,RA!1),Opy2(CR-D
    !1),Ste11(SAM!2,Sho1,Pbs2) @ 0.005
'Ste50OligmembrOpyCRD_Unbinds_Ste11membr Ref1' :Membrane Ste50(SAMoligo!2,RA!1),
    Opy2(CR-D!1),Ste11(SAM!2,Sho1,Pbs2) -> :Membrane Ste50(SAMoligo,RA!1),Opy2(
    CR-D!1),Ste11(SAM,Sho1,Pbs2) @ 0.0001
'Ste50membr_Binds_Ste11membr' :Membrane Ste11(SAM),Ste50(SAM,SAMoligo) -> :
    Membrane Ste11(SAM!1),Ste50(SAM!1,SAMoligo) @ 0.00005
'Ste50membr_Unbinds_Ste11membr' :Membrane Ste11(SAM!1),Ste50(SAM!1,SAMoligo) ->
    :Membrane Ste11(SAM),Ste50(SAM,SAMoligo) @ 0.0001
'Ste50membrOpy2A_Unbinds_Ste11membr' :Membrane Ste11(SAM!2),Opy2(CR-A!1),Ste50(
    SAM!2,RA!1) -> :Membrane Ste11(SAM),Opy2(CR-A!1),Ste50(SAM,RA!1) @ 0.0001
'Ste50membrOpy2A_Binds_Ste11membr Ref1' :Membrane Opy2(CR-A!1),Ste50(SAM,RA!1),
```

```
    Ste11(SAM,Sho1,Pbs2) -> :Membrane Opy2(CR-A!1),Ste50(SAM!2,RA!1),Ste11(SAM
    !2,Sho1,Pbs2) @ 0.005
'Ste50membrOpy2A_Unbinds_Ste11membr Ref1' :Membrane Opy2(CR-A!1),Ste50(SAM!2,RA
    !1),Ste11(SAM!2,Sho1,Pbs2) -> :Membrane Opy2(CR-A!1),Ste50(SAM,RA!1),Ste11(
    SAM,Sho1,Pbs2) @ 0.0001
'Ste50membrOpy2B_Unbinds_Ste11membr' :Membrane Ste11(SAM!2),Opy2(CR-B~p!1),Ste50
    (SAM!2,RA!1) -> :Membrane Ste11(SAM),Opy2(CR-B~p!1),Ste50(SAM,RA!1) @ 0.0001
'Ste50membrOpy2B_Binds_Ste11membr Ref1' :Membrane Opy2(CR-B~p!1),Ste50(SAM,RA!1)
    ,Ste11(SAM,Sho1,Pbs2) -> :Membrane Opy2(CR-B~p!1),Ste50(SAM!2,RA!1),Ste11(
    SAM!2,Sho1,Pbs2) @ 0.005
'Ste50membrOpy2B_Unbinds_Ste11membr Ref1' :Membrane Opy2(CR-B~p!1),Ste50(SAM!2,
    RA!1),Ste11(SAM!2,Sho1,Pbs2) -> :Membrane Opy2(CR-B~p!1),Ste50(SAM,RA!1),
    Ste11(SAM,Sho1,Pbs2) @ 0.0001
'Ste50membrOpy2D_Unbinds_Ste11membr' :Membrane Ste11(SAM!2),Ste50(SAM!2,RA!1),
    Opy2(CR-D!1) -> :Membrane Ste11(SAM),Ste50(SAM,RA!1),Opy2(CR-D!1) @ 0.0001
'Ste50membrOpy2D_Binds_Ste11membr Ref1' :Membrane Ste50(SAM,RA!1),Opy2(CR-D!1),
    Ste11(SAM,Sho1,Pbs2) -> :Membrane Ste50(SAM!2,RA!1),Opy2(CR-D!1),Ste11(SAM
    !2,Sho1,Pbs2) @ 0.005
'Ste50membrOpy2D_Unbinds_Ste11membr Ref1' :Membrane Ste50(SAM!2,RA!1),Opy2(CR-D
    !1),Ste11(SAM!2,Sho1,Pbs2) -> :Membrane Ste50(SAM,RA!1),Opy2(CR-D!1),Ste11(
    SAM,Sho1,Pbs2) @ 0.0001
'Ste11membr_Binds_Ste50Ste11membr' :Membrane Ste11(SAM),Ste50(SAM,SAMoligo!1),
    Ste11(SAM!1) -> :Membrane Ste11(SAM!2),Ste50(SAM!2,SAMoligo!1),Ste11(SAM!1)
    @ 0.00005
'Ste11membr_Unbinds_Ste50Ste11membr' :Membrane Ste11(SAM!2),Ste50(SAM!2,SAMoligo
    !1),Ste11(SAM!1) -> :Membrane Ste11(SAM),Ste50(SAM,SAMoligo!1),Ste11(SAM!1)
    @ 0.00001
'Ste11membr_Binds_Ste50Ste11membr_Oligo' :Membrane Ste11(SAM!1),Ste50(SAMoligo,
    SAM!1),Ste11(SAM) -> :Membrane Ste11(SAM!1),Ste50(SAMoligo!2,SAM!1),Ste11(
    SAM!2) @ 0.00005
'Ste11membr_Unbinds_Ste50Ste11membr_Oligo' :Membrane Ste11(SAM!1),Ste50(SAMoligo
    !2,SAM!1),Ste11(SAM!2) -> :Membrane Ste11(SAM!1),Ste50(SAMoligo,SAM!1),Ste11
    (SAM) @ 0.00001

# Transport
'Ptc23_Transp_CytoNuc' Ptc23(P,Hog1) ->:Cyto_Nuc Ptc23(P,Hog1) @ 1
'Ptc23_Transp_NucCyto' Ptc23(P,Hog1) ->:Nuc_Cyto Ptc23(P,Hog1) @ 10
'Ptc1_Transp_CytNuc' Ptc1(Nbp2,KIM,Hog1,P) ->:Cyto_Nuc Ptc1(Nbp2,KIM,Hog1,P) @ 1
'Ptc1_Transp_NucCyt' Ptc1(Nbp2,KIM,Hog1,P) ->:Nuc_Cyto Ptc1(Nbp2,KIM,Hog1,P) @
    10
'Hog1pp_Tr_Nuc Ref1' Hog1(Thr174~p,Tyr176~p,Ste50,CD,TF,Phosphat,Ptc23,Kinase,
    Sho1) ->:Cyto_Nuc Hog1(Thr174~p,Tyr176~p,Ste50,CD,TF,Phosphat,Ptc23,Kinase,
```

```
    Sho1) @ 30
'Hog1nuc_Tr_Cyto' Hog1(TF,CD,Ste50,Thr174,Tyr176,Phosphat,Ptc23,Kinase,Sho1) ->:
    Nuc_Cyto Hog1(TF,CD,Ste50,Thr174,Tyr176,Phosphat,Ptc23,Kinase,Sho1) @ 10
'Bem1membr_cyto' Bem1(Ste5,SH3b,PB1,CI) ->:Memb_Cyto Bem1(Ste5,SH3b,PB1,CI) @ 10
'Bem1cyto_membr' Bem1(Ste5,SH3b,PB1,CI) ->:Cyto_Memb Bem1(Ste5,SH3b,PB1,CI) @ 1
'Ypd1membr_Cyto' Ypd1(Sln1,Ssk1) ->:Memb_Cyto Ypd1(Sln1,Ssk1) @ 10
'Ypd1cyto_membr' Ypd1(Sln1,Ssk1) ->:Cyto_Memb Ypd1(Sln1,Ssk1) @ 1
'Ssk1DimerMembr_Cyto' Ssk1(Dimer!1,Ssk2_22,Ypd1),Ssk1(Ypd1,Dimer!1,Ssk2_22) ->:
    Memb_Cyto Ssk1(Dimer!1,Ssk2_22,Ypd1),Ssk1(Ypd1,Dimer!1,Ssk2_22) @ 10
'Ssk1DimerCyto_Membr' Ssk1(Dimer!1,Ssk2_22,Ypd1),Ssk1(Ypd1,Dimer!1,Ssk2_22) ->:
    Cyto_Memb Ssk1(Dimer!1,Ssk2_22,Ypd1),Ssk1(Ypd1,Dimer!1,Ssk2_22) @ 1
'Ssk1Membr_Cyto' Ssk1(Ypd1,Ssk2_22,Dimer) ->:Memb_Cyto Ssk1(Ypd1,Ssk2_22,Dimer)
    @ 10
'Ssk1Cyto_Membr' Ssk1(Ypd1,Ssk2_22,Dimer) ->:Cyto_Memb Ssk1(Ypd1,Ssk2_22,Dimer)
    @ 1
'Pbs2membr_Transp_cyto' Pbs2(sho91_102,Ste11,HBD-I,ssk44_57,Nbp2) ->:Memb_Cyto
    Pbs2(sho91_102,Ste11,HBD-I,ssk44_57,Nbp2) @ 20
'Pbs2cyto_Transp_membr' Pbs2(sho91_102,Ste11,HBD-I,ssk44_57,Nbp2) ->:Cyto_Memb
    Pbs2(sho91_102,Ste11,HBD-I,ssk44_57,Nbp2) @ 2
'Pbs2Ste11membr_Transp_cyto' Pbs2(Nbp2,HBD-I,sho91_102,Ste11!1,ssk44_57,Ptc23),
    Ste11(Sho1,Cla4,SAM,Ste20,Pbs2!1) ->:Memb_Cyto Pbs2(Nbp2,HBD-I,sho91_102,
    Ste11!1,ssk44_57,Ptc23),Ste11(Sho1,Cla4,SAM,Ste20,Pbs2!1) @ 10
'Pbs2Ste11cyto_Transp_membr' Pbs2(Nbp2,HBD-I,sho91_102,Ste11!1,ssk44_57,Ptc23),
    Ste11(Sho1,Cla4,SAM,Ste20,Pbs2!1) ->:Cyto_Memb Pbs2(Nbp2,HBD-I,sho91_102,
    Ste11!1,ssk44_57,Ptc23),Ste11(Sho1,Cla4,SAM,Ste20,Pbs2!1) @ 1
'Pbs2Ste11Ste50Ste11membr_Transp_cyto' Pbs2(Nbp2,HBD-I,sho91_102,Ste11!1),Ste11(
    Sho1,Cla4,SAM!2,Ste20,Pbs2!1),Ste50(Sho1,RA,SAM!2,SAMoligo!3),Ste11(Sho1,
    Pbs2,Cla4,Ste20,SAM!3) ->:Memb_Cyto Pbs2(Nbp2,HBD-I,sho91_102,Ste11!1),Ste11
    (Sho1,Cla4,SAM!2,Ste20,Pbs2!1),Ste50(Sho1,RA,SAM!2,SAMoligo!3),Ste11(Sho1,
    Pbs2,Cla4,Ste20,SAM!3) @ 10
'Pbs2Ste11Ste50Ste11cyto_Transp_membr' Pbs2(Nbp2,HBD-I,sho91_102,Ste11!1),Ste11(
    Sho1,Cla4,SAM!2,Ste20,Pbs2!1),Ste50(Sho1,RA,SAM!2,SAMoligo!3),Ste11(Sho1,
    Pbs2,Cla4,Ste20,SAM!3) ->:Cyto_Memb Pbs2(Nbp2,HBD-I,sho91_102,Ste11!1),Ste11
    (Sho1,Cla4,SAM!2,Ste20,Pbs2!1),Ste50(Sho1,RA,SAM!2,SAMoligo!3),Ste11(Sho1,
    Pbs2,Cla4,Ste20,SAM!3) @ 1
'Pbs2Ste11oligSte50Ste11membr_Transp_cyto' Pbs2(HBD-I,Nbp2,ssk44_57,sho91_102,
    Ste11!1),Ste11(Sho1,Cla4,SAM!2,Ste20,Pbs2!1),Ste50(Sho1,SAM!3,RA,SAMoligo!2)
    ,Ste11(Sho1,Pbs2,Cla4,Ste20,SAM!3) ->:Memb_Cyto Pbs2(HBD-I,Nbp2,ssk44_57,
    sho91_102,Ste11!1),Ste11(Sho1,Cla4,SAM!2,Ste20,Pbs2!1),Ste50(Sho1,SAM!3,RA,
    SAMoligo!2),Ste11(Sho1,Pbs2,Cla4,Ste20,SAM!3) @ 10
'Pbs2Ste11oligSte50Ste11cyto_Transp_membr' Pbs2(HBD-I,Nbp2,ssk44_57,sho91_102,
    Ste11!1),Ste11(Sho1,Cla4,SAM!2,Ste20,Pbs2!1),Ste50(Sho1,SAM!3,RA,SAMoligo!2)
```

```
    ,Ste11(Sho1,Pbs2,Cla4,Ste20,SAM!3) ->:Cyto_Memb Pbs2(HBD-I,Nbp2,ssk44_57,
    sho91_102,Ste11!1),Ste11(Sho1,Cla4,SAM!2,Ste20,Pbs2!1),Ste50(Sho1,SAM!3,RA,
    SAMoligo!2),Ste11(Sho1,Pbs2,Cla4,Ste20,SAM!3) @ 1
'Pbs2Nbp2Ptc1membr_Transp_cyto' Pbs2(HBD-I,ssk44_57,Ste11,sho91_102,Nbp2!2),Ptc1
    (KIM,Nbp2!1),Nbp2(Ptc1!1,Pbs2!2) ->:Memb_Cyto Pbs2(HBD-I,ssk44_57,Ste11,
    sho91_102,Nbp2!2),Ptc1(KIM,Nbp2!1),Nbp2(Ptc1!1,Pbs2!2) @ 10
'Pbs2Nbp2Ptc1cyto_Transp_membr' Pbs2(HBD-I,ssk44_57,Ste11,sho91_102,Nbp2!2),Ptc1
    (KIM,Nbp2!1),Nbp2(Ptc1!1,Pbs2!2) ->:Cyto_Memb Pbs2(HBD-I,ssk44_57,Ste11,
    sho91_102,Nbp2!2),Ptc1(KIM,Nbp2!1),Nbp2(Ptc1!1,Pbs2!2) @ 1
'Pbs2Ste11Ste50oligTransp_cyto' Pbs2(HBD-I,ssk44_57,sho91_102,Ste11!1,Nbp2,Ptc23
    ),Ste11(Sho1,Cla4,SAM!2,Ste20,Pbs2!1),Ste50(Sho1,SAM,RA,SAMoligo!2) ->:
    Memb_Cyto Pbs2(HBD-I,ssk44_57,sho91_102,Ste11!1,Nbp2,Ptc23),Ste11(Sho1,Cla4,
    SAM!2,Ste20,Pbs2!1),Ste50(Sho1,SAM,RA,SAMoligo!2) @ 10
'Pbs2Ste11Ste50oligTransp_membr' Pbs2(HBD-I,ssk44_57,sho91_102,Ste11!1,Nbp2,
    Ptc23),Ste11(Sho1,Cla4,SAM!2,Ste20,Pbs2!1),Ste50(Sho1,SAM,RA,SAMoligo!2) ->:
    Cyto_Memb Pbs2(HBD-I,ssk44_57,sho91_102,Ste11!1,Nbp2,Ptc23),Ste11(Sho1,Cla4,
    SAM!2,Ste20,Pbs2!1),Ste50(Sho1,SAM,RA,SAMoligo!2) @ 1
'Pbs2Ste11Ste50_Transp_cyto' Ste11(Sho1,Cla4,SAM!2,Ste20,Pbs2!1),Ste50(Sho1,RA,
    SAM!2,SAMoligo),Pbs2(HBD-I,sho91_102,Ste11!1,Nbp2,Ptc23) ->:Memb_Cyto Ste11(
    Sho1,Cla4,SAM!2,Ste20,Pbs2!1),Ste50(Sho1,RA,SAM!2,SAMoligo),Pbs2(HBD-I,
    sho91_102,Ste11!1,Nbp2,Ptc23) @ 10
'Pbs2Ste11Ste50_Transp_membr' Ste11(Sho1,Cla4,SAM!2,Ste20,Pbs2!1),Ste50(Sho1,RA,
    SAM!2,SAMoligo),Pbs2(HBD-I,sho91_102,Ste11!1,Nbp2,Ptc23) ->:Cyto_Memb Ste11(
    Sho1,Cla4,SAM!2,Ste20,Pbs2!1),Ste50(Sho1,RA,SAM!2,SAMoligo),Pbs2(HBD-I,
    sho91_102,Ste11!1,Nbp2,Ptc23) @ 1
'Cdc24membr_cyto' Cdc24(Sho1,DH,PB1,PhosphSites,PH) ->:Memb_Cyto Cdc24(Sho1,DH,
    PB1,PhosphSites,PH) @ 10
'Cdc24cyto_membr' Cdc24(Sho1,DH,PB1,PhosphSites,PH) ->:Cyto_Memb Cdc24(Sho1,DH,
    PB1,PhosphSites,PH) @ 1
'Ste20membr_cyto' Ste20(Ste11,CRIB,PRR) ->:Memb_Cyto Ste20(Ste11,CRIB,PRR) @ 10
'Ste20cyto_membr' Ste20(Ste11,CRIB,PRR) ->:Cyto_Memb Ste20(Ste11,CRIB,PRR) @ 1
'Cla4membr_cyto' Cla4(CRIB,Ste11,PRR) ->:Memb_Cyto Cla4(CRIB,Ste11,PRR) @ 10
'Cla4cyto_membr' Cla4(CRIB,Ste11,PRR) ->:Cyto_Memb Cla4(CRIB,Ste11,PRR) @ 1
'Ste11membr_cyto' Ste11(SAM,Sho1,Pbs2,Cla4,Ste20) ->:Memb_Cyto Ste11(SAM,Sho1,
    Pbs2,Cla4,Ste20) @ 10
'Ste11cyto_membr' Ste11(SAM,Sho1,Pbs2,Cla4,Ste20) ->:Cyto_Memb Ste11(SAM,Sho1,
    Pbs2,Cla4,Ste20) @ 1
'Ste50Ste11Ste11membr_cyto' Ste50(RA,SAM!3,Sho1,SAMoligo!2),Ste11(SAM!2,Sho1,
    Cla4,Pbs2,Ste20),Ste11(SAM!3,Sho1,Pbs2,Cla4,Ste20) ->:Memb_Cyto Ste50(RA,SAM
    !3,Sho1,SAMoligo!2),Ste11(SAM!2,Sho1,Cla4,Pbs2,Ste20),Ste11(SAM!3,Sho1,Pbs2,
    Cla4,Ste20) @ 10
'Ste50Ste11Ste11cyto_membr' Ste50(RA,SAM!3,Sho1,SAMoligo!2),Ste11(SAM!2,Sho1,
```

```
    Cla4,Pbs2,Ste20),Ste11(SAM!3,Sho1,Pbs2,Cla4,Ste20) ->:Cyto_Memb Ste50(RA,SAM
    !3,Sho1,SAMoligo!2),Ste11(SAM!2,Sho1,Cla4,Pbs2,Ste20),Ste11(SAM!3,Sho1,Pbs2,
    Cla4,Ste20) @ 1
'Ste50Ste11membr_cyto' Ste11(SAM!2,Sho1,Pbs2,Cla4,Ste20),Ste50(RA,SAMoligo,SAM
    !2,Sho1) ->:Memb_Cyto Ste11(SAM!2,Sho1,Pbs2,Cla4,Ste20),Ste50(RA,SAMoligo,
    SAM!2,Sho1) @ 10
'Ste50Ste11cyto_membr' Ste11(SAM!2,Sho1,Pbs2,Cla4,Ste20),Ste50(RA,SAMoligo,SAM
    !2,Sho1) ->:Cyto_Memb Ste11(SAM!2,Sho1,Pbs2,Cla4,Ste20),Ste50(RA,SAMoligo,
    SAM!2,Sho1) @ 1
'Ste50Ste11OligMembr_cyto' Ste11(SAM!2,Sho1,Pbs2,Cla4,Ste20),Ste50(RA,SAMoligo
    !2,SAM,Sho1) ->:Memb_Cyto Ste11(SAM!2,Sho1,Pbs2,Cla4,Ste20),Ste50(RA,
    SAMoligo!2,SAM,Sho1) @ 10
'Ste50Ste11OligCyto_membr' Ste11(SAM!2,Sho1,Pbs2,Cla4,Ste20),Ste50(RA,SAMoligo
    !2,SAM,Sho1) ->:Cyto_Memb Ste11(SAM!2,Sho1,Pbs2,Cla4,Ste20),Ste50(RA,
    SAMoligo!2,SAM,Sho1) @ 1
'Ste50membr_cyto' Ste50(RA,SAMoligo,SAM,Sho1) ->:Memb_Cyto Ste50(RA,SAMoligo,SAM
    ,Sho1) @ 10
'Ste50cyto_membr' Ste50(RA,SAMoligo,SAM,Sho1) ->:Cyto_Memb Ste50(RA,SAMoligo,SAM
    ,Sho1) @ 1
'Osmcyto_membr' Osm(Localiz~i) ->:Cyto_Memb Osm(Localiz~i) @ [inf]


### Initial Conditions:
%init:  20000 :Membrane     Trigger()
%init:   2000 :Membrane     Hkr1(STR,HMH,Sho1,OsmIntern,X~ia)
%init:   1160 :Membrane     Opy2(CR-A,CR-D,CR-B~p,CR-C,Yck)
%init:   2330 :Membrane     Sho1(Cdc24,Ste50,SH3_338,SH3_342_346,Mucin,Mucin2,
    Hog1~u,X~ia)
%init:    656 :Membrane     Sln1_RecD(Ypd1,h!1,Asp1144~p),Sln1_SensD(Osm,
    OsmIntern,k!2,X~a),Sln1_HkD(DimR,r!1,s!2,His576~p)
%init:   1200 :Membrane     Ssk1(Dimer,Ypd1,Ssk2_22,Asp554~p)
%init:    336 :Membrane     Ste11(Sho1,Pbs2,Cla4,Ste20,SAM,Ser281Ser285Thr286~u)
%init:   1360 :Membrane     Msb2(STR,HMH,Sho1,Cyt_Cdc42,OsmIntern,X~ia)
%init:    900 :Membrane     Cdc42(GEF,ToCRIB,Ste50,Msb2,Bem1,X~GDP)
%init:    907 :Membrane     Fps1(Osm,Hog1,A~a)
%init:   1630 :Membrane     Yck1_2(Opy2,A~a)
%init:   5330 :Membrane     Ypd1(Sln1,Ssk1,His64~u)
%init:   1000 :Membrane     Ypd1(Sln1,Ssk1,His64~p)
%init:   6780 :Cytosol      Hog1(Ste50,Sho1,Ptc23,CD~pbd1,TF,Phosphat,Kinase,
    Thr174~u,Tyr176~u)
%init:   2160 :Cytosol      Pbs2(ssk44_57,sho91_102,Ptc23,Nbp2,Ste11,HBD-I,
    Ser514~u,Thr518~u)
```

```
%init:    200 :Cytosol      Ste50(S155_196_202_248~u,Sho1,SAMoligo!2,SAM!1,RA,
    Hog1),Ste11(Sho1,Pbs2,Cla4,SAM!1,Ste20,Ser281Ser285Thr286~u),Ste11(Sho1,Pbs2
    ,Cla4,SAM!2,Ste20,Ser281Ser285Thr286~u)
%init:    590 :Cytosol      Ste50(S155_196_202_248~u,Sho1,SAMoligo,SAM,RA,Hog1)
%init:    110 :Cytosol      Ste50(RA,S155_196_202_248~u,SAM!1,SAMoligo!0,Sho1,
    Hog1),Ste50(RA,S155_196_202_248~u,SAM!1,SAMoligo!2,Sho1,Hog1),Ste50(RA,
    S155_196_202_248~u,SAM,SAMoligo!2,Sho1,Hog1),Ste50(RA,S155_196_202_248~u,SAM
    !0,SAMoligo,Sho1,Hog1)
%init:    110 :Cytosol      Ste50(RA,S155_196_202_248~u,SAM!0,SAMoligo,Sho1,Hog1
    ),Ste50(RA,S155_196_202_248~u,SAM!1,SAMoligo!0,Sho1,Hog1),Ste50(RA,
    S155_196_202_248~u,SAM!1,SAMoligo!2,Sho1,Hog1),Ste50(RA,S155_196_202_248~u,
    SAM!2,SAMoligo,Sho1,Hog1)
%init:   1520 :Cytosol      Ptc1(Hog1,Nbp2,KIM,P~a)
%init:  18000 :Cytosol      Ptc23(Hog1,P~ia)
%init:    768 :Cytosol      Ptp(Hog1,mapk,P~a)
%init:    521 :Cytosol      Nbp2(Ptc1,Pbs2)
%init:   6490 :Cytosol      Bem1(CI,PB1,Ste5,SH3b)
%init:   1010 :Cytosol      Cdc24(Sho1,DH,PB1,PH,X~ia,PhosphSites~u)
%init:    259 :Cytosol      Cla4(PRR,Ste11,CRIB,X~ia)
%init:    217 :Cytosol      Ssk2(Pbs2,phosphatas,s294_413,Thr1460~u,Ssk)
%init:   1000 :Cytosol      Ssk22(Pbs2,phosphatas,s98_179,Thr1460~u)
%init:    259 :Cytosol      Ste20(PRR,Ste11,CRIB,X~ia)
%init:    100 :Cytosol      FeedbackDummy(Hog1pp,Ptc23,X~ia)
%init:    150 :Cytosol      GlycFeedback(Hog1,GlycProd,X~ia)
%init:   5000 :Cytosol      GlycProd(Hog1,X~ia)
%init:    500 :Nucleus      Sko1(Hog1pp)
%init:    149 :Nucleus      Ptp(Hog1,mapk,P~a)
%init:     10 :Nucleus      Remodeler(PolII)
%init:    200 :Nucleus      PolII(Gene,Remodel)
%init:    150 :Nucleus      TF(Gene,Hog1pp)
%init:     10 :Nucleus      gene(TF,PolII,Remodel)
%init:      1 :Nucleus      mCherry(TF,PolII,Remodel)
%init:      1 :Nucleus      Venus(TF,PolII,Remodel)

### Simulation:
#%obs: 'Membrane Internal Osm' :Membrane Osm(r,Localiz~i)
#%obs: 'Cytosol Internal Osm' :Cytosol Osm(r,Localiz~i)
#%obs: 'Internal Osm' Osm(r,Localiz~i)
#%obs: 'Membrane External Osm' :Membrane Osm(Localiz~e)
#%obs: 'External Osm' Osm(Localiz~e)
#%obs: 'Sln1' :Membrane Sln1_SensD(X~ia)
#%obs: 'Sho1' :Membrane Sho1(X~a)
```

```
%obs: 'Nuc Hog1pp' :Nucleus Hog1(Thr174~p,Tyr176~p)
%obs: 'Cyto Hog1pp' :Cytosol Hog1(Tyr176~p,Thr174~p)
#%obs: 'Cyto Pbs2pp' :Cytosol Pbs2(Thr518~p,Ser514~p)

### Perturbations:
%mod: [T]>800 do $UPDATE 'TriggerOsm' [inf]
```

A.9 A Synthetic Switch

```
# John Wilson-Kanamori and John Moore
#
# Nitrosylase Induction in Yeast - last update 30-06-2013
#
# This model displays unidirectional switch behaviour upon perturbation at T
    =100.
# NPR1 monomerises due to induced TRXH5 and de-activated (due to induced GSNOR)
    GSNO,
# thus activating a reporter pathway. When induction is turned off at T=300,
# the model returns to its pre-induction state of oligomeric NPR1.
#
# Observations are levels of NPR1, TRXH5, GSNOR, and LUC reporter.


### Agents:
%agent: NPR1(C156~u~s,S1,S2,tga3,loc~nuc~cyt)
%agent: GSNO(x~active~inactive)
%agent: GSNOR(x)
%agent: TRXH5(x)
%agent: TGA3(npr1,dna)
%agent: DNA(type~luc,tga3)
%agent: mRNA(type~luc,loc~nuc~cyt)
%agent: LUC()


### Rules:

# GSNO action
'NPR1-GSNO binding' NPR1(C156~u), GSNO(x~active) -> NPR1(C156~u!1), GSNO(x~
    active!1) @ 1.0
'NPR1-GSNO unbinding' NPR1(C156!1), GSNO(x!1) -> NPR1(C156), GSNO(x) @ 1.0
'NPR1-GSNO S-nitrosylation' NPR1(C156~u!1), GSNO(x~active!1) -> NPR1(C156~s!1),
    GSNO(x~active!1) @ 10.0

# GSNOR action
'GSNOR creation' -> GSNOR(x) @ 0.0
'GSNO-GSNOR binding' GSNO(x~active), GSNOR(x) -> GSNO(x~active!1), GSNOR(x!1) @
    1.0
'GSNO-GSNOR unbinding' GSNO(x!1), GSNOR(x!1) -> GSNO(x), GSNOR(x) @ 10.0
'GSNO inactivation' GSNO(x~active) -> GSNO(x~inactive) @ 0.001
```

```
'GSNO-GSNOR inactivation' GSNO(x~active!1), GSNOR(x!1) -> GSNO(x~inactive!1),
    GSNOR(x!1) @ 10.0
'GSNO activation' GSNO(x~inactive) -> GSNO(x~active) @ 0.05
'GSNOR degradation' GSNOR(x) -> @ 0.05

# TRXH5 action
'TRXH5 creation' -> TRXH5(x) @ 0.0
'NPR1-TRXH5 binding' NPR1(C156~s), TRXH5(x) -> NPR1(C156~s!1), TRXH5(x!1) @ 1.0
'NPR1-TRXH5 unbinding' NPR1(C156!1), TRXH5(x!1) -> NPR1(C156), TRXH5(x) @ 10.0
'NPR1 de-nitrosylation' NPR1(C156~s) -> NPR1(C156~u) @ 0.1
'NPR1-TRXH5 de-nitrosylation' NPR1(C156~s!1), TRXH5(x!1) -> NPR1(C156~u!1),
    TRXH5(x!1) @ 10.0
'TRXH5 degradation' TRXH5(x) -> @ 0.05

# NPR1 action
'NPR1 multimer formation' NPR1(C156~s,S1,S2,loc~cyt), NPR1(C156~s,S1,S2,loc~cyt)
    , NPR1(C156~s,S1,S2,loc~cyt), NPR1(C156~s,S1,S2,loc~cyt) -> NPR1(C156~s,S1
    !1,S2!2,loc~cyt), NPR1(C156~s,S1!1,S2!3,loc~cyt), NPR1(C156~s,S1!2,S2!4,loc~
    cyt), NPR1(C156~s,S1!3,S2!4,loc~cyt) @ 100.0
'NPR1 multimer dissolution' NPR1(C156~u,S1!1,S2!2,loc~cyt), NPR1(C156~u,S1!1,S2
    !3,loc~cyt), NPR1(C156~u,S1!2,S2!4,loc~cyt), NPR1(C156~u,S1!3,S2!4,loc~cyt)
    -> NPR1(C156~u,S1,S2,loc~cyt), NPR1(C156~u,S1,S2,loc~cyt), NPR1(C156~u,S1,S2
    ,loc~cyt), NPR1(C156~u,S1,S2,loc~cyt) @ 5.0

# Transport
'NPR1 nuclear transport' NPR1(C156~u,S1,S2,loc~cyt) -> NPR1(C156~u,S1,S2,loc~nuc
    ) @ 10.0
'NPR1 cytosolic transport' NPR1(tga3,loc~nuc) -> NPR1(tga3,loc~cyt) @ 10.0
'mRNA cytosolic transport' mRNA(loc~nuc) -> mRNA(loc~cyt) @ 10.0

# Reporter system
'NPR1-TGA3 binding' NPR1(C156~u,tga3,loc~nuc), TGA3(npr1,dna) -> NPR1(C156~u,
    tga3!1,loc~nuc), TGA3(npr1!1,dna) @ 1.0
'NPR1-TGA3 unbinding' NPR1(tga3!1,loc~nuc), TGA3(npr1!1,dna) -> NPR1(tga3,loc~
    nuc), TGA3(npr1,dna) @ 10.0
'Transcription factor binding' NPR1(C156~u,tga3!1,loc~nuc), TGA3(npr1!1,dna),
    DNA(type~luc,tga3) -> NPR1(C156~u,tga3!1,loc~nuc), TGA3(npr1!1,dna!2), DNA(
    type~luc,tga3!2) @ 1.0
'Transcription factor unbinding' NPR1(tga3!1,loc~nuc), TGA3(npr1!1,dna!2), DNA(
    type~luc,tga3!2) -> NPR1(tga3!1,loc~nuc), TGA3(npr1!1,dna), DNA(type~luc,
    tga3) @ 100.0
'Transcription' DNA(type~luc,tga3!_) -> DNA(type~luc,tga3!_), mRNA(type~luc,loc~
```

```
    nuc) @ 1.0
'Translation' mRNA(type~luc,loc~cyt) -> mRNA(type~luc,loc~cyt), LUC() @ 5.0
'mRNA degradation' mRNA() -> @ 1.0
'LUC degradation' LUC() -> @ 0.05


### Initial Conditions:
%init: 100 (NPR1(C156~u,S1!1,S2!2,tga3,loc~cyt), NPR1(C156~u,S1!1,S2!3,tga3,loc~
    cyt), NPR1(C156~u,S1!2,S2!4,tga3,loc~cyt), NPR1(C156~u,S1!3,S2!4,tga3,loc~
    cyt))
%init: 400 (TGA3(npr1,dna))
%init: 300 (GSNO(x~active))
%init: 1 (DNA(type~luc,tga3))

### Simulation:
%obs: 'TRXH' TRXH5()
%obs: 'LUX' LUC()
%obs: 'Nuclear NPR1' NPR1(loc~nuc)
%obs: 'GSNOR' GSNOR()
%obs: 'Active GSNO' GSNO(x~active)

### Stories:

### Perturbations:

# Let's turn on the activity of GSNOR at t=100 and off again at t=300.
%mod: [T]>100 do $UPDATE 'GSNOR creation' 10.0
%mod: [T]>300 do $UPDATE 'GSNOR creation' 0.0

# Let's turn on the activity of TRXH5 at t=100 and off again at t=300.
%mod: [T]>100 do $UPDATE 'TRXH5 creation' 10.0
%mod: [T]>300 do $UPDATE 'TRXH5 creation' 0.0
```

A.10 The Repressilator

```
# Ty Thomson / Edinburgh 2010 iGEM Team
#
# Core repressilator model - last update 29-03-2011
#
# This model has active core repressilator only.
#
# Observations are levels of LacI, TetR and cI.
#
# The core repressilator model was developed initially by Ty Thomson in 2009.
# The original version is available at:
# http://www.rulebase.org/showcase_books/182350-Rule-Based-Modeling-of-BioBrick-
    Parts
#
# DNA types: Within the model, the following DNA() types represent the various
    BioBrick sequences used
#
# BBaB0011 transcription terminator
# BBaB0034 ribosome binding site
# BBaC0012 lacI coding sequence
# BBaC0040 tetR coding sequence
# BBaC0051 lambda-cI coding sequence
# BBaR0010 lacI promoter
# BBaR0040 tetR promoter
# BBaR0051 lambda-cI promoter

%agent: DNA(binding,downstream,type~BBaB0011~BBaB0034~BBaC0012~BBaC0040~BBaC0051
    ~BBaR0010p1~BBaR0010p2~BBaR0010p3~BBaR0010p4~BBaR0040p1~BBaR0040p2~
    BBaR0040p3~BBaR0040p4~BBaR0051p1~BBaR0051p2~BBaR0051p3~BBaR0051p4,upstream)
%agent: RNA(downstream,upstream,type~BBaB0034~BBaC0012~BBaC0040~BBaC0051~
    BBaR0010~BBaR0040~BBaR0051,binding)
%agent: RNAP(rna,dna)
%agent: Ribosome(rna)
%agent: TetR(atc,dna)
%agent: cI(dna)
%agent: LacI(lactose,dna)


# Transcription factor binding to promoter region.

%var: 'transcription factor binding rate' 0.01
```

```
'LacI binding to R0010p2 (no LacI)' \
       DNA(binding,type~BBaR0010p3,upstream!2), LacI(dna,lactose), DNA(
           downstream!2,binding,type~BBaR0010p2) -> \
       DNA(binding,type~BBaR0010p3,upstream!3), LacI(dna!1,lactose), DNA(
           downstream!3,binding!1,type~BBaR0010p2) @ 'transcription factor
           binding rate'
'LacI binding to R0010p2 (LacI bound)' \
       DNA(binding!1,type~BBaR0010p3,upstream!2), LacI(dna!1), DNA(downstream!2,
           binding,type~BBaR0010p2), LacI(dna,lactose) -> \
       DNA(binding!2,type~BBaR0010p3,upstream!3), LacI(dna!2), DNA(downstream!3,
           binding!1,type~BBaR0010p2), LacI(dna!1,lactose) @ 'transcription
           factor binding rate'
'LacI binding to R0010p3 (no LacI)' \
       DNA(binding,type~BBaR0010p3,upstream!2), LacI(dna,lactose), DNA(
           downstream!2,binding,type~BBaR0010p2) -> \
       DNA(binding!1,type~BBaR0010p3,upstream!3), LacI(dna!1,lactose), DNA(
           downstream!3,binding,type~BBaR0010p2) @ 'transcription factor binding
            rate'
'LacI binding to R0010p3 (LacI bound)' \
       DNA(binding,type~BBaR0010p3,upstream!2), LacI(dna!1), DNA(downstream!2,
           binding!1,type~BBaR0010p2), LacI(dna,lactose) -> \
       DNA(binding!1,type~BBaR0010p3,upstream!3), LacI(dna!2), DNA(downstream!3,
           binding!2,type~BBaR0010p2), LacI(dna!1,lactose) @ 'transcription
           factor binding rate'
'TetR binding to R0040p2 (no TetR)' \
       DNA(binding,type~BBaR0040p3,upstream!2), TetR(dna,atc), DNA(downstream!2,
           binding,type~BBaR0040p2) -> \
       DNA(binding,type~BBaR0040p3,upstream!3), TetR(dna!1,atc), DNA(downstream
           !3,binding!1,type~BBaR0040p2) @ 'transcription factor binding rate'
'TetR binding to R0040p2 (TetR bound)' \
       DNA(binding!1,type~BBaR0040p3,upstream!2), TetR(dna!1), TetR(dna,atc),
           DNA(downstream!2,binding,type~BBaR0040p2) -> \
       DNA(binding!2,type~BBaR0040p3,upstream!3), TetR(dna!2), TetR(dna!1,atc),
           DNA(downstream!3,binding!1,type~BBaR0040p2) @ 'transcription factor
           binding rate'
'TetR binding to R0040p3 (no TetR)' \
       DNA(binding,type~BBaR0040p3,upstream!2), TetR(dna,atc), DNA(downstream!2,
           binding,type~BBaR0040p2) -> \
       DNA(binding!1,type~BBaR0040p3,upstream!3), TetR(dna!1,atc), DNA(
           downstream!3,binding,type~BBaR0040p2) @ 'transcription factor binding
            rate'
```

```
'TetR binding to R0040p3 (TetR bound)' \
       DNA(binding,type~BBaR0040p3,upstream!2), TetR(dna,atc), TetR(dna!1), DNA(
           downstream!2,binding!1,type~BBaR0040p2) -> \
       DNA(binding!2,type~BBaR0040p3,upstream!3), TetR(dna!2,atc), TetR(dna!1),
           DNA(downstream!3,binding!1,type~BBaR0040p2) @ 'transcription factor
           binding rate'
'cI binding to R0051p2 (no cI)' \
       DNA(binding,type~BBaR0051p3,upstream!2), cI(dna), DNA(downstream!2,
           binding,type~BBaR0051p2) -> \
       DNA(binding,type~BBaR0051p3,upstream!3), cI(dna!1), DNA(downstream!3,
           binding!1,type~BBaR0051p2) @ 'transcription factor binding rate'
'cI binding to R0051p2 (cI bound)' \
       DNA(binding!1,type~BBaR0051p3,upstream!2), cI(dna), cI(dna!1), DNA(
           downstream!2,binding,type~BBaR0051p2) -> \
       DNA(binding!2,type~BBaR0051p3,upstream!3), cI(dna!1), cI(dna!2), DNA(
           downstream!3,binding!1,type~BBaR0051p2) @ 'transcription factor
           binding rate'
'cI binding to R0051p3 (no cI)' \
       DNA(binding,type~BBaR0051p3,upstream!2), cI(dna), DNA(downstream!2,
           binding,type~BBaR0051p2) -> \
       DNA(binding!1,type~BBaR0051p3,upstream!3), cI(dna!1), DNA(downstream!3,
           binding,type~BBaR0051p2) @ 'transcription factor binding rate'
'cI binding to R0051p3 (cI bound)' \
       DNA(binding,type~BBaR0051p3,upstream!2), cI(dna!1), cI(dna), DNA(
           downstream!2,binding!1,type~BBaR0051p2) -> \
       DNA(binding!1,type~BBaR0051p3,upstream!3), cI(dna!2), cI(dna!1), DNA(
           downstream!3,binding!2,type~BBaR0051p2) @ 'transcription factor
           binding rate'

%var: 'sole LacI transcription factor unbinding rate' 2.24
%var: 'non-sole LacI transcription factor unbinding rate' 0.09
%var: 'sole TetR transcription factor unbinding rate' 2.24
%var: 'non-sole TetR transcription factor unbinding rate' 0.09
%var: 'sole cI transcription factor unbinding rate' 2.24
%var: 'non-sole cI transcription factor unbinding rate' 0.09

'Reverse LacI binding to R0010p2 (no LacI)' \
       DNA(binding,type~BBaR0010p3,upstream!3), LacI(dna!1,lactose), DNA(
           downstream!3,binding!1,type~BBaR0010p2) -> \
       DNA(binding,type~BBaR0010p3,upstream!2), LacI(dna,lactose), DNA(
           downstream!2,binding,type~BBaR0010p2) @ 'sole LacI transcription
           factor unbinding rate'
```

```
'Reverse LacI binding to R0010p2 (LacI bound)' \
       DNA(binding!2,type~BBaR0010p3,upstream!3), LacI(dna!2), DNA(downstream!3,
           binding!1,type~BBaR0010p2), LacI(dna!1,lactose) -> \
       DNA(binding!1,type~BBaR0010p3,upstream!2), LacI(dna!1), DNA(downstream!2,
           binding,type~BBaR0010p2), LacI(dna,lactose) @ 'non-sole LacI
           transcription factor unbinding rate'
'Reverse LacI binding to R0010p3 (no LacI)' \
       DNA(binding!1,type~BBaR0010p3,upstream!3), LacI(dna!1,lactose), DNA(
           downstream!3,binding,type~BBaR0010p2) -> \
       DNA(binding,type~BBaR0010p3,upstream!2), LacI(dna,lactose), DNA(
           downstream!2,binding,type~BBaR0010p2) @ 'sole LacI transcription
           factor unbinding rate'
'Reverse LacI binding to R0010p3 (LacI bound)' \
       DNA(binding!1,type~BBaR0010p3,upstream!3), LacI(dna!2), DNA(downstream!3,
           binding!2,type~BBaR0010p2), LacI(dna!1,lactose) -> \
       DNA(binding,type~BBaR0010p3,upstream!2), LacI(dna!1), DNA(downstream!2,
           binding!1,type~BBaR0010p2), LacI(dna,lactose) @ 'non-sole LacI
           transcription factor unbinding rate'
'Reverse TetR binding to R0040p2 (no TetR)' \
       DNA(binding,type~BBaR0040p3,upstream!3), TetR(dna!1,atc), DNA(downstream
           !3,binding!1,type~BBaR0040p2) -> \
       DNA(binding,type~BBaR0040p3,upstream!2), TetR(dna,atc), DNA(downstream!2,
           binding,type~BBaR0040p2) @ 'sole TetR transcription factor unbinding
           rate'
'Reverse TetR binding to R0040p2 (TetR bound)' \
       DNA(binding!2,type~BBaR0040p3,upstream!3), TetR(dna!2), TetR(dna!1,atc),
           DNA(downstream!3,binding!1,type~BBaR0040p2) -> \
       DNA(binding!1,type~BBaR0040p3,upstream!2), TetR(dna!1), TetR(dna,atc),
           DNA(downstream!2,binding,type~BBaR0040p2) @ 'non-sole TetR
           transcription factor unbinding rate'
'Reverse TetR binding to R0040p3 (no TetR)' \
       DNA(binding!1,type~BBaR0040p3,upstream!3), TetR(dna!1,atc), DNA(
           downstream!3,binding,type~BBaR0040p2) -> \
       DNA(binding,type~BBaR0040p3,upstream!2), TetR(dna,atc), DNA(downstream!2,
           binding,type~BBaR0040p2) @ 'sole TetR transcription factor unbinding
           rate'
'Reverse TetR binding to R0040p3 (TetR bound)' \
       DNA(binding!2,type~BBaR0040p3,upstream!3), TetR(dna!2,atc), TetR(dna!1),
           DNA(downstream!3,binding!1,type~BBaR0040p2) -> \
       DNA(binding,type~BBaR0040p3,upstream!2), TetR(dna,atc), TetR(dna!1), DNA(
           downstream!2,binding!1,type~BBaR0040p2) @ 'non-sole TetR
           transcription factor unbinding rate'
```

```
'Reverse cI binding to R0051p2 (no cI)' \
       DNA(binding,type~BBaR0051p3,upstream!3), cI(dna!1), DNA(downstream!3,
           binding!1,type~BBaR0051p2) -> \
       DNA(binding,type~BBaR0051p3,upstream!2), cI(dna), DNA(downstream!2,
           binding,type~BBaR0051p2) @ 'sole cI transcription factor unbinding
           rate'
'Reverse cI binding to R0051p2 (cI bound)' \
       DNA(binding!2,type~BBaR0051p3,upstream!3), cI(dna!1), cI(dna!2), DNA(
           downstream!3,binding!1,type~BBaR0051p2) -> \
       DNA(binding!1,type~BBaR0051p3,upstream!2), cI(dna), cI(dna!1), DNA(
           downstream!2,binding,type~BBaR0051p2) @ 'non-sole cI transcription
           factor unbinding rate'
'Reverse cI binding to R0051p3 (no cI)' \
       DNA(binding!1,type~BBaR0051p3,upstream!3), cI(dna!1), DNA(downstream!3,
           binding,type~BBaR0051p2) -> \
       DNA(binding,type~BBaR0051p3,upstream!2), cI(dna), DNA(downstream!2,
           binding,type~BBaR0051p2) @ 'sole cI transcription factor unbinding
           rate'
'Reverse cI binding to R0051p3 (cI bound)' \
       DNA(binding!1,type~BBaR0051p3,upstream!3), cI(dna!2), cI(dna!1), DNA(
           downstream!3,binding!2,type~BBaR0051p2) -> \
       DNA(binding,type~BBaR0051p3,upstream!2), cI(dna!1), cI(dna), DNA(
           downstream!2,binding!1,type~BBaR0051p2) @ 'non-sole cI transcription
           factor unbinding rate'

# RNAP binding to promoter regions.

%var: 'high RNAP binding rate' 0.0007
%var: 'low RNAP binding rate' 7e-07

'RNAP binding to R0010 (no LacI)' \
       DNA(binding,type~BBaR0010p3,upstream!2,downstream!1), DNA(upstream!1,
           binding,type~BBaR0010p4), RNAP(dna,rna), DNA(downstream!2,binding,
           type~BBaR0010p2) -> \
       DNA(binding,type~BBaR0010p3,upstream!3,downstream!1), DNA(upstream!1,
           binding!2,type~BBaR0010p4), RNAP(dna!2,rna), DNA(downstream!3,binding
           ,type~BBaR0010p2) @ 'high RNAP binding rate'
'RNAP binding to R0010 (LacI on p2)' \
       DNA(binding,type~BBaR0010p3,upstream!2,downstream!1), DNA(upstream!1,
           binding,type~BBaR0010p4), RNAP(dna,rna), DNA(downstream!2,binding!3,
           type~BBaR0010p2), LacI(dna!3) -> \
       DNA(binding,type~BBaR0010p3,upstream!3,downstream!1), DNA(upstream!1,
```

```
          binding!2,type~BBaR0010p4), RNAP(dna!2,rna), DNA(downstream!3,binding
          !4,type~BBaR0010p2), LacI(dna!4) @ 'low RNAP binding rate'
'RNAP binding to R0010 (LacI on p3)' \
      DNA(binding!3,type~BBaR0010p3,upstream!2,downstream!1), DNA(upstream!1,
          binding,type~BBaR0010p4), RNAP(dna,rna), DNA(downstream!2,binding,
          type~BBaR0010p2), LacI(dna!3) -> \
      DNA(binding!4,type~BBaR0010p3,upstream!3,downstream!1), DNA(upstream!1,
          binding!2,type~BBaR0010p4), RNAP(dna!2,rna), DNA(downstream!3,binding
          ,type~BBaR0010p2), LacI(dna!4) @ 'low RNAP binding rate'
'RNAP binding to R0010 (LacI on p2 and p3)' \
      DNA(binding!3,type~BBaR0010p3,upstream!2,downstream!1), DNA(upstream!1,
          binding,type~BBaR0010p4), RNAP(dna,rna), DNA(downstream!2,binding!4,
          type~BBaR0010p2), LacI(dna!3), LacI(dna!4) -> \
      DNA(binding!4,type~BBaR0010p3,upstream!3,downstream!1), DNA(upstream!1,
          binding!2,type~BBaR0010p4), RNAP(dna!2,rna), DNA(downstream!3,binding
          !5,type~BBaR0010p2), LacI(dna!4), LacI(dna!5) @ 'low RNAP binding
          rate'
'RNAP binding to R0040 (no TetR)' \
      DNA(binding,type~BBaR0040p3,upstream!2,downstream!1), DNA(upstream!1,
          binding,type~BBaR0040p4), RNAP(dna,rna), DNA(downstream!2,binding,
          type~BBaR0040p2) -> \
      DNA(binding,type~BBaR0040p3,upstream!3,downstream!1), DNA(upstream!1,
          binding!2,type~BBaR0040p4), RNAP(dna!2,rna), DNA(downstream!3,binding
          ,type~BBaR0040p2) @ 'high RNAP binding rate'
'RNAP binding to R0040 (TetR on p2)' \
      DNA(binding,type~BBaR0040p3,upstream!2,downstream!1), DNA(upstream!1,
          binding,type~BBaR0040p4), RNAP(dna,rna), DNA(downstream!2,binding!3,
          type~BBaR0040p2), TetR(dna!3) -> \
      DNA(binding,type~BBaR0040p3,upstream!3,downstream!1), DNA(upstream!1,
          binding!2,type~BBaR0040p4), RNAP(dna!2,rna), DNA(downstream!3,binding
          !4,type~BBaR0040p2), TetR(dna!4) @ 'low RNAP binding rate'
'RNAP binding to R0040 (TetR on p3)' \
      DNA(binding!3,type~BBaR0040p3,upstream!2,downstream!1), DNA(upstream!1,
          binding,type~BBaR0040p4), RNAP(dna,rna), DNA(downstream!2,binding,
          type~BBaR0040p2), TetR(dna!3) -> \
      DNA(binding!4,type~BBaR0040p3,upstream!3,downstream!1), DNA(upstream!1,
          binding!2,type~BBaR0040p4), RNAP(dna!2,rna), DNA(downstream!3,binding
          ,type~BBaR0040p2), TetR(dna!4) @ 'low RNAP binding rate'
'RNAP binding to R0040 (TetR on p2 and p3)' \
      DNA(binding!3,type~BBaR0040p3,upstream!2,downstream!1), DNA(upstream!1,
          binding,type~BBaR0040p4), RNAP(dna,rna), DNA(downstream!2,binding!4,
          type~BBaR0040p2), TetR(dna!3), TetR(dna!4) -> \
```

```
       DNA(binding!4,type~BBaR0040p3,upstream!3,downstream!1), DNA(upstream!1,
           binding!2,type~BBaR0040p4), RNAP(dna!2,rna), DNA(downstream!3,binding
           !5,type~BBaR0040p2), TetR(dna!4), TetR(dna!5) @ 'low RNAP binding
           rate'
'RNAP binding to R0051 (no cI)' \
       DNA(binding,type~BBaR0051p3,upstream!2,downstream!1), DNA(upstream!1,
           binding,type~BBaR0051p4), RNAP(dna,rna), DNA(downstream!2,binding,
           type~BBaR0051p2) -> \
       DNA(binding,type~BBaR0051p3,upstream!3,downstream!1), DNA(upstream!1,
           binding!2,type~BBaR0051p4), RNAP(dna!2,rna), DNA(downstream!3,binding
           ,type~BBaR0051p2) @ 'high RNAP binding rate'
'RNAP binding to R0051 (cI on p2)' \
       DNA(binding,type~BBaR0051p3,upstream!2,downstream!1), DNA(upstream!1,
           binding,type~BBaR0051p4), RNAP(dna,rna), DNA(downstream!2,binding!3,
           type~BBaR0051p2), cI(dna!3) -> \
       DNA(binding,type~BBaR0051p3,upstream!3,downstream!1), DNA(upstream!1,
           binding!2,type~BBaR0051p4), RNAP(dna!2,rna), DNA(downstream!3,binding
           !4,type~BBaR0051p2), cI(dna!4) @ 'low RNAP binding rate'
'RNAP binding to R0051 (cI on p3)' \
       DNA(binding!3,type~BBaR0051p3,upstream!2,downstream!1), DNA(upstream!1,
           binding,type~BBaR0051p4), RNAP(dna,rna), DNA(downstream!2,binding,
           type~BBaR0051p2), cI(dna!3) -> \
       DNA(binding!4,type~BBaR0051p3,upstream!3,downstream!1), DNA(upstream!1,
           binding!2,type~BBaR0051p4), RNAP(dna!2,rna), DNA(downstream!3,binding
           ,type~BBaR0051p2), cI(dna!4) @ 'low RNAP binding rate'
'RNAP binding to R0051 (cI on p2 and p3)' \
       DNA(binding!3,type~BBaR0051p3,upstream!2,downstream!1), DNA(upstream!1,
           binding,type~BBaR0051p4), RNAP(dna,rna), DNA(downstream!2,binding!4,
           type~BBaR0051p2), cI(dna!3), cI(dna!4) -> \
       DNA(binding!4,type~BBaR0051p3,upstream!3,downstream!1), DNA(upstream!1,
           binding!2,type~BBaR0051p4), RNAP(dna!2,rna), DNA(downstream!3,binding
           !5,type~BBaR0051p2), cI(dna!4), cI(dna!5) @ 'low RNAP binding rate'



# Transcription.

%var: 'transcription initiation rate' 10
%var: 'transcription rate' 10
%var: 'transcription termination rate' 10

'Transcription initiation of R0051' \
```

```
        DNA(binding!1,type~BBaR0051p4,downstream!2), RNAP(dna!1,rna), DNA(
            upstream!2,binding) -> \
        DNA(binding,type~BBaR0051p4,downstream!3), RNAP(dna!1,rna!2), DNA(
            upstream!3,binding!1), RNA(binding,upstream,downstream!2,type~
            BBaR0051) @ 'transcription initiation rate'
'Transcription initiation of R0010' \
        DNA(binding!1,type~BBaR0010p4,downstream!2), RNAP(dna!1,rna), DNA(
            upstream!2,binding) -> \
        DNA(binding,type~BBaR0010p4,downstream!3), RNAP(dna!1,rna!2), DNA(
            upstream!3,binding!1), RNA(binding,upstream,downstream!2,type~
            BBaR0010) @ 'transcription initiation rate'
'Transcription initiation of R0040' \
        DNA(binding!1,type~BBaR0040p4,downstream!2), RNAP(dna!1,rna), DNA(
            upstream!2,binding) -> \
        DNA(binding,type~BBaR0040p4,downstream!3), RNAP(dna!1,rna!2), DNA(
            upstream!3,binding!1), RNA(binding,upstream,downstream!2,type~
            BBaR0040) @ 'transcription initiation rate'

'RBS BBa_B0034 transcription' \
        DNA(binding!1,downstream!2,type~BBaB0034), RNAP(dna!1,rna!3), DNA(
            upstream!2,binding), RNA(downstream!3) -> \
        DNA(binding,downstream!2,type~BBaB0034), RNAP(dna!1,rna!3), DNA(upstream
            !2,binding!1), RNA(downstream!4), RNA(binding,upstream!4,downstream
            !3,type~BBaB0034) @ 'transcription rate'

'C0012 transcription' \
        DNA(binding!1,downstream!2,type~BBaC0012), RNAP(dna!1,rna!3), DNA(
            upstream!2,binding), RNA(downstream!3) -> \
        DNA(binding,downstream!2,type~BBaC0012), RNAP(dna!1,rna!3), DNA(upstream
            !2,binding!1), RNA(downstream!4), RNA(binding,upstream!4,downstream
            !3,type~BBaC0012) @ 'transcription rate'
'C0051 transcription' \
        DNA(binding!1,downstream!2,type~BBaC0051), RNAP(dna!1,rna!3), DNA(
            upstream!2,binding), RNA(downstream!3) -> \
        DNA(binding,downstream!2,type~BBaC0051), RNAP(dna!1,rna!3), DNA(upstream
            !2,binding!1), RNA(downstream!4), RNA(binding,upstream!4,downstream
            !3,type~BBaC0051) @ 'transcription rate'
'C0040 transcription' \
        DNA(binding!1,downstream!2,type~BBaC0040), RNAP(dna!1,rna!3), DNA(
            upstream!2,binding), RNA(downstream!3) -> \
        DNA(binding,downstream!2,type~BBaC0040), RNAP(dna!1,rna!3), DNA(upstream
            !2,binding!1), RNA(downstream!4), RNA(binding,upstream!4,downstream
```

```
            !3,type~BBaC0040) @ 'transcription rate'

'Termination - B0011' \
       DNA(binding!1,type~BBaB0011), RNAP(dna!1,rna!2), RNA(downstream!2) -> \
       DNA(binding,type~BBaB0011), RNAP(dna,rna), RNA(downstream) @ '
            transcription termination rate'

%var: 'RNAP falloff rate' 1.0

'RNAP falloff' \
       DNA(binding!1,downstream!3), RNAP(dna!1,rna!2), RNA(downstream!2), DNA(
            upstream!3,binding!_) -> \
       DNA(binding,downstream!1), RNAP(dna,rna), RNA(downstream), DNA(upstream
            !1,binding!_) @ 'RNAP falloff rate'

# The following rule is not invoked in this particular model as BBaB0011 is
    always the last DNA agent in the sequence, hence no downstream agent
#'B0011 terminator transcription (readthrough)' \
#      DNA(binding!1,downstream!2,type~BBaB0011), RNAP(dna!1,rna!3), DNA(
    upstream!2,binding), RNA(downstream!3) -> \
#      DNA(binding,downstream!2,type~BBaB0011), RNAP(dna!1,rna!3), DNA(upstream
    !2,binding!1), RNA(downstream!4), RNA(binding,upstream!4,downstream!3,type~
    BBaB0011) @ 0.5

'Transcription of R0051 (readthrough)' \
       DNA(binding,type~BBaR0051p3,downstream!2,upstream!3), RNAP(dna!1,rna!5),
            DNA(upstream!6,binding), DNA(upstream!4,downstream!3,binding,type~
            BBaR0051p2), DNA(downstream!4,binding!1,type~BBaR0051p1), \
       RNA(downstream!5), DNA(upstream!2,downstream!6,binding,type~BBaR0051p4)
            -> \
       DNA(binding,type~BBaR0051p3,downstream!3,upstream!5), RNAP(dna!1,rna!6),
            DNA(upstream!7,binding!1), DNA(upstream!4,downstream!5,binding,type~
            BBaR0051p2), DNA(downstream!4,binding,type~BBaR0051p1), \
       RNA(downstream!2), DNA(upstream!3,downstream!7,binding,type~BBaR0051p4),
            RNA(binding,upstream!2,downstream!6,type~BBaR0051) @ 'transcription
            rate'
'Transcription of R0010 (readthrough)' \
       DNA(binding,type~BBaR0010p3,downstream!2,upstream!3), RNAP(dna!1,rna!5),
            DNA(upstream!6,binding), DNA(upstream!4,downstream!3,binding,type~
            BBaR0010p2), DNA(downstream!4,binding!1,type~BBaR0010p1), \
       RNA(downstream!5), DNA(upstream!2,downstream!6,binding,type~BBaR0010p4)
            -> \
```

```
        DNA(binding,type~BBaR0010p3,downstream!3,upstream!5), RNAP(dna!1,rna!6),
            DNA(upstream!7,binding!1), DNA(upstream!4,downstream!5,binding,type~
            BBaR0010p2), DNA(downstream!4,binding,type~BBaR0010p1), \
        RNA(downstream!2), DNA(upstream!3,downstream!7,binding,type~BBaR0010p4),
            RNA(binding,upstream!2,downstream!6,type~BBaR0010) @ 'transcription
            rate'
'Transcription of R0040 (readthrough)' \
        DNA(binding,type~BBaR0040p3,downstream!2,upstream!3), RNAP(dna!1,rna!5),
            DNA(upstream!6,binding), DNA(upstream!4,downstream!3,binding,type~
            BBaR0040p2), DNA(downstream!4,binding!1,type~BBaR0040p1), \
        RNA(downstream!5), DNA(upstream!2,downstream!6,binding,type~BBaR0040p4)
            -> \
        DNA(binding,type~BBaR0040p3,downstream!3,upstream!5), RNAP(dna!1,rna!6),
            DNA(upstream!7,binding!1), DNA(upstream!4,downstream!5,binding,type~
            BBaR0040p2), DNA(downstream!4,binding,type~BBaR0040p1), \
        RNA(downstream!2), DNA(upstream!3,downstream!7,binding,type~BBaR0040p4),
            RNA(binding,upstream!2,downstream!6,type~BBaR0040) @ 'transcription
            rate'


# Translation.

%var: 'ribosome binding rate' 0.000166

'RBS BBa_B0034 Ribosome binding' \
        RNA(binding,type~BBaB0034), Ribosome(rna) -> \
        RNA(binding!1,type~BBaB0034), Ribosome(rna!1) @ 'ribosome binding rate'

%var: 'translation initiation rate' 0.167

'LacI translation initiation' \
        RNA(binding!2,downstream!1), RNA(binding,upstream!1,type~BBaC0012),
            Ribosome(rna!2) -> \
        RNA(binding,downstream!1), RNA(binding!2,upstream!1,type~BBaC0012),
            Ribosome(rna!2) @ 'translation initiation rate'
'TetR translation initiation' \
        RNA(binding!2,downstream!1), RNA(binding,upstream!1,type~BBaC0040),
            Ribosome(rna!2) -> \
        RNA(binding,downstream!1), RNA(binding!2,upstream!1,type~BBaC0040),
            Ribosome(rna!2) @ 'translation initiation rate'
'cI translation initiation' \
        RNA(binding!2,downstream!1), RNA(binding,upstream!1,type~BBaC0051),
```

```
            Ribosome(rna!2) -> \
        RNA(binding,downstream!1), RNA(binding!2,upstream!1,type~BBaC0051),
            Ribosome(rna!2) @ 'translation initiation rate'

%var: 'translation rate' 10

'LacI translation' \
        RNA(binding!1,type~BBaC0012), Ribosome(rna!1) -> \
        RNA(binding,type~BBaC0012), Ribosome(rna), LacI(dna,lactose) @ '
            translation rate'
'TetR translation' \
        RNA(binding!1,type~BBaC0040), Ribosome(rna!1) -> \
        RNA(binding,type~BBaC0040), Ribosome(rna), TetR(dna,atc) @ 'translation
            rate'
'cI translation' \
        RNA(binding!1,type~BBaC0051), Ribosome(rna!1) -> \
        RNA(binding,type~BBaC0051), Ribosome(rna), cI(dna) @ 'translation rate'

%var: 'ribosome falloff rate' 0.01

'Ribosome falloff' Ribosome(rna!1), RNA(binding!1) -> Ribosome(rna), RNA(binding
    ) @ 'ribosome falloff rate'


# Degradation of various agents.

'RNA degradation' RNA(binding,downstream) -> @ 0.0058

%var: 'transcription factor degradation rate' 0.00115


'LacI degradation' LacI(dna) -> @ 'transcription factor degradation rate'
'TetR degradation' TetR(dna) -> @ 'transcription factor degradation rate'
'cI degradation' cI(dna) -> @ 'transcription factor degradation rate'


# Initial conditions, observables, and perturbations.

%var: 'operon count' 1
%var: 'RNA polymerase count' 700
%var: 'ribosome count' 18000
```

```
%init: 'RNA polymerase count' (RNAP(dna,rna))
%init: 'ribosome count' (Ribosome(rna))
%init: 'operon count' (DNA(upstream,downstream!4,binding,type~BBaR0051p1), DNA(
    upstream!4,downstream!5,binding,type~BBaR0051p2), DNA(upstream!5,downstream
    !6,binding,type~BBaR0051p3), \
                    DNA(upstream!6,downstream!7,binding,type~BBaR0051p4), DNA(
                        upstream!7,downstream!8,binding,type~BBaB0034), DNA(
                        upstream!8,downstream!9,binding,type~BBaC0012), \
                    DNA(upstream!9,downstream,binding,type~BBaB0011))
%init: 'operon count' (DNA(upstream,downstream!6,binding,type~BBaR0010p1), \
                    DNA(upstream!6,downstream!7,binding,type~BBaR0010p2), DNA(
                        upstream!7,downstream!8,binding,type~BBaR0010p3), DNA(
                        upstream!8,downstream!9,binding,type~BBaR0010p4), \
                    DNA(upstream!9,downstream!10,binding,type~BBaB0034), DNA(
                        upstream!10,downstream!11,binding,type~BBaC0040), DNA(
                        upstream!11,downstream,binding,type~BBaB0011))
%init: 'operon count' (DNA(upstream,downstream!4,binding,type~BBaR0040p1), DNA(
    upstream!4,downstream!5,binding,type~BBaR0040p2), DNA(upstream!5,downstream
    !6,binding,type~BBaR0040p3), \
                    DNA(upstream!6,downstream!7,binding,type~BBaR0040p4), DNA(
                        upstream!7,downstream!8,binding,type~BBaB0034), DNA(
                        upstream!8,downstream!9,binding,type~BBaC0051), \
                    DNA(upstream!9,downstream,binding,type~BBaB0011))

%obs: 'LacI' LacI()
%obs: 'TetR' TetR()
%obs: 'cI' cI()
```

A.11 Light-Based Communication in *E.coli*

```
# Edinburgh 2010 iGEM Team
#
# Complete single cell repressilating lights model - last update 30-03-2011
#
# This is the complete repressilator and light communication model within a
# single bacterial cell.
#
# Observations are levels of LacI, TetR and cI, and the three emitted lights
# red, blue and green.
#
# The core repressilator model was developed initially by Ty Thomson in 2009.
# The original version is available at:
# http://www.rulebase.org/showcase_books/182350-Rule-Based-Modeling-of-BioBrick-
    Parts
#
# DNA types: Within the model, the following DNA() types represent the various
# BioBrick sequences used
#
# BBaB0011 transcription terminator
# BBaB0034 ribosome binding site
# BBaC0012 lacI coding sequence
# BBaC0040 tetR coding sequence
# BBaC0051 lambda-cI coding sequence
# BBaC0BLUE blue luciferase coding sequence
# BBaC0GREEN green luciferase coding sequence
# BBaC0RED red luciferase coding sequence
# BBaK191007 trp promoter
# BBaR0010 lacI promoter
# BBaR0040 tetR promoter
# BBaR0051 lambda-cI promoter
# BBaR0082 ompC promoter
# BBaR0084 ompF promoter - alternate to ompC promoter, not active in model.
    Inserting it in place of ompC causes the oscillations to collapse.
# BBaR0PHO phoA promoter

%agent: BLUE()
%agent: BLUELuc()
%agent: CcaS(active~off~on,phob)
%agent: Cph8(active~off~on,ompr)
%agent: DNA(type~BBaB0011~BBaB0034~BBaC0012~BBaC0040~BBaC0051~BBaC0BLUE~
```

```
    BBaC0GREEN~BBaC0RED~BBaK191007p1~BBaK191007p2~BBaK191007p3~BBaR0010p1~
    BBaR0010p2~BBaR0010p3~BBaR0010p4~BBaR0040p1~BBaR0040p2~BBaR0040p3~BBaR0040p4
    ~BBaR0051p1~BBaR0051p2~BBaR0051p3~BBaR0051p4~BBaR0082p1~BBaR0082p2~
    BBaR0082p3~BBaR0082p4~BBaR0082p5~BBaR0084p1~BBaR0084p2~BBaR0084p3~BBaR0084p4
    ~BBaR0084p5~BBaR0PHOp1~BBaR0PHOp2~BBaR0PHOp3,downstream,binding,upstream)
%agent: GREEN()
%agent: GREENLuc()
%agent: LOVTAP(active~dark~light,dna)
%agent: LacI(lactose,dna)
%agent: OmpR(dna,site~p~u,cph8)
%agent: PhoB(site~p~u,ccas,dna)
%agent: RED()
%agent: REDLuc()
%agent: RNA(type~BBaB0034~BBaC0012~BBaC0040~BBaC0051~BBaC0BLUE~BBaC0GREEN~
    BBaC0RED~BBaK191007~BBaR0010~BBaR0040~BBaR0051~BBaR0082~BBaR0084~BBaR0PHO,
    upstream,binding,downstream)
%agent: RNAP(dna,rna)
%agent: Ribosome(rna)
%agent: TetR(atc,dna)
%agent: cI(dna)

# Signal transduction pathway for red light.

'Cph8 auto-activation' Cph8(active~off) -> Cph8(active~on) @ 0.01
'Cph8 auto-deactivation' Cph8(active~on) -> Cph8(active~off) @ 0.001
'Cph8 deactivation by RED light' Cph8(active~on), RED() -> Cph8(active~off), RED
    () @ 0.1

'OmpR auto-phosphorylation' OmpR(site~u) -> OmpR(site~p) @ 0.001
'OmpR auto-dephosphorylation' OmpR(dna,site~p) -> OmpR(dna,site~u) @ 0.03
'OmpR binding to Cph8' OmpR(dna,cph8,site~u), Cph8(active~on,ompr) -> OmpR(dna,
    cph8!1,site~u), Cph8(active~on,ompr!1) @ 0.01
'OmpR dissociation from Cph8' OmpR(cph8!1), Cph8(ompr!1) -> OmpR(cph8), Cph8(
    ompr) @ 0.01
'OmpR phosphorylation by Cph8' OmpR(cph8!1,site~u), Cph8(active~on,ompr!1) ->
    OmpR(cph8!1,site~p), Cph8(active~on,ompr!1) @ 0.1


# Signal transduction pathway for blue light.

'LOVTAP auto-activation' LOVTAP(active~dark) -> LOVTAP(active~light) @ 0.0001
'LOVTAP auto-deactivation' LOVTAP(dna,active~light) -> LOVTAP(dna,active~dark) @
```

```
     0.01
'LOVTAP activation by BLUE light' LOVTAP(active~dark), BLUE() -> LOVTAP(active~
    light), BLUE() @ 0.1


# Signal transduction pathway for green light.

'CcaS-PhoR auto-activation' CcaS(active~off) -> CcaS(active~on) @ 0.001
'CcaS-PhoR auto-deactivation' CcaS(active~on) -> CcaS(active~off) @ 0.01
'CcaS-PhoR activation by GREEN light' CcaS(active~off), GREEN() -> CcaS(active~
    on), GREEN() @ 0.1

'PhoB auto-phosphorylation' PhoB(site~u) -> PhoB(site~p) @ 0.001
'PhoB auto-dephosphorylation' PhoB(dna,site~p) -> PhoB(dna,site~u) @ 0.03
'PhoB binding to CcaS' PhoB(dna,ccas,site~u), CcaS(active~on,phob) -> PhoB(dna,
    ccas!1,site~u), CcaS(active~on,phob!1) @ 0.01
'PhoB dissociation from CcaS' PhoB(ccas!1), CcaS(phob!1) -> PhoB(ccas), CcaS(
    phob) @ 0.01
'PhoB phosphorylation by CcaS' PhoB(ccas!1,site~u), CcaS(active~on,phob!1) ->
    PhoB(ccas!1,site~p), CcaS(active~on,phob!1) @ 0.1


# Transcription factor binding to promoter region.

%var: 'transcription factor binding rate' 0.01

'LacI binding to R0010p2 (no LacI)' \
       DNA(binding,type~BBaR0010p3,upstream!2), LacI(dna,lactose), DNA(
           downstream!2,binding,type~BBaR0010p2) -> \
       DNA(binding,type~BBaR0010p3,upstream!3), LacI(dna!1,lactose), DNA(
           downstream!3,binding!1,type~BBaR0010p2) @ 'transcription factor
           binding rate'
'LacI binding to R0010p2 (LacI bound)' \
       DNA(binding!1,type~BBaR0010p3,upstream!2), LacI(dna!1), DNA(downstream!2,
           binding,type~BBaR0010p2), LacI(dna,lactose) -> \
       DNA(binding!2,type~BBaR0010p3,upstream!3), LacI(dna!2), DNA(downstream!3,
           binding!1,type~BBaR0010p2), LacI(dna!1,lactose) @ 'transcription
           factor binding rate'
'LacI binding to R0010p3 (no LacI)' \
       DNA(binding,type~BBaR0010p3,upstream!2), LacI(dna,lactose), DNA(
           downstream!2,binding,type~BBaR0010p2) -> \
       DNA(binding!1,type~BBaR0010p3,upstream!3), LacI(dna!1,lactose), DNA(
```

```
            downstream!3,binding,type~BBaR0010p2) @ 'transcription factor binding
             rate'
'LacI binding to R0010p3 (LacI bound)' \
       DNA(binding,type~BBaR0010p3,upstream!2), LacI(dna!1), DNA(downstream!2,
           binding!1,type~BBaR0010p2), LacI(dna,lactose) -> \
       DNA(binding!1,type~BBaR0010p3,upstream!3), LacI(dna!2), DNA(downstream!3,
           binding!2,type~BBaR0010p2), LacI(dna!1,lactose) @ 'transcription
           factor binding rate'
'TetR binding to R0040p2 (no TetR)' \
       DNA(binding,type~BBaR0040p3,upstream!2), TetR(dna,atc), DNA(downstream!2,
           binding,type~BBaR0040p2) -> \
       DNA(binding,type~BBaR0040p3,upstream!3), TetR(dna!1,atc), DNA(downstream
           !3,binding!1,type~BBaR0040p2) @ 'transcription factor binding rate'
'TetR binding to R0040p2 (TetR bound)' \
       DNA(binding!1,type~BBaR0040p3,upstream!2), TetR(dna!1), TetR(dna,atc),
           DNA(downstream!2,binding,type~BBaR0040p2) -> \
       DNA(binding!2,type~BBaR0040p3,upstream!3), TetR(dna!2), TetR(dna!1,atc),
           DNA(downstream!3,binding!1,type~BBaR0040p2) @ 'transcription factor
           binding rate'
'TetR binding to R0040p3 (no TetR)' \
       DNA(binding,type~BBaR0040p3,upstream!2), TetR(dna,atc), DNA(downstream!2,
           binding,type~BBaR0040p2) -> \
       DNA(binding!1,type~BBaR0040p3,upstream!3), TetR(dna!1,atc), DNA(
           downstream!3,binding,type~BBaR0040p2) @ 'transcription factor binding
            rate'
'TetR binding to R0040p3 (TetR bound)' \
       DNA(binding,type~BBaR0040p3,upstream!2), TetR(dna,atc), TetR(dna!1), DNA(
           downstream!2,binding!1,type~BBaR0040p2) -> \
       DNA(binding!2,type~BBaR0040p3,upstream!3), TetR(dna!2,atc), TetR(dna!1),
           DNA(downstream!3,binding!1,type~BBaR0040p2) @ 'transcription factor
           binding rate'
'cI binding to R0051p2 (no cI)' \
       DNA(binding,type~BBaR0051p3,upstream!2), cI(dna), DNA(downstream!2,
           binding,type~BBaR0051p2) -> \
       DNA(binding,type~BBaR0051p3,upstream!3), cI(dna!1), DNA(downstream!3,
           binding!1,type~BBaR0051p2) @ 'transcription factor binding rate'
'cI binding to R0051p2 (cI bound)' \
       DNA(binding!1,type~BBaR0051p3,upstream!2), cI(dna), cI(dna!1), DNA(
           downstream!2,binding,type~BBaR0051p2) -> \
       DNA(binding!2,type~BBaR0051p3,upstream!3), cI(dna!1), cI(dna!2), DNA(
           downstream!3,binding!1,type~BBaR0051p2) @ 'transcription factor
           binding rate'
```

```
'cI binding to R0051p3 (no cI)' \
       DNA(binding,type~BBaR0051p3,upstream!2), cI(dna), DNA(downstream!2,
           binding,type~BBaR0051p2) -> \
       DNA(binding!1,type~BBaR0051p3,upstream!3), cI(dna!1), DNA(downstream!3,
           binding,type~BBaR0051p2) @ 'transcription factor binding rate'
'cI binding to R0051p3 (cI bound)' \
       DNA(binding,type~BBaR0051p3,upstream!2), cI(dna!1), cI(dna), DNA(
           downstream!2,binding!1,type~BBaR0051p2) -> \
       DNA(binding!1,type~BBaR0051p3,upstream!3), cI(dna!2), cI(dna!1), DNA(
           downstream!3,binding!2,type~BBaR0051p2) @ 'transcription factor
           binding rate'
'OmpR binding to R0082p2' \
       DNA(binding,type~BBaR0082p2,downstream!2), DNA(upstream!2,type~BBaR0082p3
           ,downstream!3), DNA(upstream!3,type~BBaR0082p4), OmpR(cph8,site~p,dna
           ) -> \
       DNA(binding!1,type~BBaR0082p2,downstream!2), DNA(upstream!2,type~
           BBaR0082p3,downstream!3), DNA(upstream!3,type~BBaR0082p4), OmpR(cph8,
           site~p,dna!1) @ 'transcription factor binding rate'
'OmpR binding to R0082p3' \
       DNA(type~BBaR0082p2,downstream!2), DNA(upstream!2,binding,type~BBaR0082p3
           ,downstream!3), DNA(upstream!3,type~BBaR0082p4), OmpR(cph8,site~p,dna
           ) -> \
       DNA(type~BBaR0082p2,downstream!2), DNA(upstream!2,binding!1,type~
           BBaR0082p3,downstream!3), DNA(upstream!3,type~BBaR0082p4), OmpR(cph8,
           site~p,dna!1) @ 'transcription factor binding rate'
'OmpR binding to R0082p4' \
       DNA(type~BBaR0082p2,downstream!2), DNA(upstream!2,type~BBaR0082p3,
           downstream!3), DNA(upstream!3,binding,type~BBaR0082p4), OmpR(cph8,
           site~p,dna) -> \
       DNA(type~BBaR0082p2,downstream!2), DNA(upstream!2,type~BBaR0082p3,
           downstream!3), DNA(upstream!3,binding!1,type~BBaR0082p4), OmpR(cph8,
           site~p,dna!1) @ 'transcription factor binding rate'
'OmpR binding to R0084p2' \
       DNA(binding,type~BBaR0084p2,downstream!2), DNA(upstream!2,type~BBaR0084p3
           ,downstream!3), DNA(upstream!3,type~BBaR0084p4), OmpR(cph8,site~u,dna
           ) -> \
       DNA(binding!1,type~BBaR0084p2,downstream!2), DNA(upstream!2,type~
           BBaR0084p3,downstream!3), DNA(upstream!3,type~BBaR0084p4), OmpR(cph8,
           site~u,dna!1) @ 'transcription factor binding rate'
'OmpR binding to R0084p3' \
       DNA(type~BBaR0084p2,downstream!2), DNA(upstream!2,binding,type~BBaR0084p3
           ,downstream!3), DNA(upstream!3,type~BBaR0084p4), OmpR(cph8,site~u,dna
```

```
          ) -> \
       DNA(type~BBaR0084p2,downstream!2), DNA(upstream!2,binding!1,type~
          BBaR0084p3,downstream!3), DNA(upstream!3,type~BBaR0084p4), OmpR(cph8,
          site~u,dna!1) @ 'transcription factor binding rate'
'OmpR binding to R0084p4' \
       DNA(type~BBaR0084p2,downstream!2), DNA(upstream!2,type~BBaR0084p3,
          downstream!3), DNA(upstream!3,binding,type~BBaR0084p4), OmpR(cph8,
          site~u,dna) -> \
       DNA(type~BBaR0084p2,downstream!2), DNA(upstream!2,type~BBaR0084p3,
          downstream!3), DNA(upstream!3,binding!1,type~BBaR0084p4), OmpR(cph8,
          site~u,dna!1) @ 'transcription factor binding rate'
'LOVTAP binding to K191007' \
       DNA(binding,type~BBaK191007p2,downstream!2), DNA(upstream!2,binding,type~
          BBaK191007p3), LOVTAP(active~light,dna) -> \
       DNA(binding!1,type~BBaK191007p2,downstream!2), DNA(upstream!2,binding,
          type~BBaK191007p3), LOVTAP(active~light,dna!1) @ 'transcription
          factor binding rate'
'PhoB binding to R0PHO' \
       DNA(binding,type~BBaR0PHOp2,downstream!2), DNA(upstream!2,binding,type~
          BBaR0PHOp3), PhoB(ccas,site~p,dna) -> \
       DNA(binding!1,type~BBaR0PHOp2,downstream!2), DNA(upstream!2,binding,type~
          BBaR0PHOp3), PhoB(ccas,site~p,dna!1) @ 'transcription factor binding
          rate'

%var: 'sole LacI transcription factor unbinding rate' 2.24
%var: 'non-sole LacI transcription factor unbinding rate' 0.09
%var: 'sole TetR transcription factor unbinding rate' 2.24
%var: 'non-sole TetR transcription factor unbinding rate' 0.09
%var: 'sole cI transcription factor unbinding rate' 2.24
%var: 'non-sole cI transcription factor unbinding rate' 0.09

%var: 'OmpR transcription factor unbinding rate' 0.03
%var: 'LOVTAP transcription factor unbinding rate' 0.03
%var: 'PhoB transcription factor unbinding rate' 0.03

'Reverse LacI binding to R0010p2 (no LacI)' \
       DNA(binding,type~BBaR0010p3,upstream!3), LacI(dna!1,lactose), DNA(
          downstream!3,binding!1,type~BBaR0010p2) -> \
       DNA(binding,type~BBaR0010p3,upstream!2), LacI(dna,lactose), DNA(
          downstream!2,binding,type~BBaR0010p2) @ 'sole LacI transcription
          factor unbinding rate'
'Reverse LacI binding to R0010p2 (LacI bound)' \
```

```
        DNA(binding!2,type~BBaR0010p3,upstream!3), LacI(dna!2), DNA(downstream!3,
            binding!1,type~BBaR0010p2), LacI(dna!1,lactose) -> \
        DNA(binding!1,type~BBaR0010p3,upstream!2), LacI(dna!1), DNA(downstream!2,
            binding,type~BBaR0010p2), LacI(dna,lactose) @ 'non-sole LacI
            transcription factor unbinding rate'
'Reverse LacI binding to R0010p3 (no LacI)' \
        DNA(binding!1,type~BBaR0010p3,upstream!3), LacI(dna!1,lactose), DNA(
            downstream!3,binding,type~BBaR0010p2) -> \
        DNA(binding,type~BBaR0010p3,upstream!2), LacI(dna,lactose), DNA(
            downstream!2,binding,type~BBaR0010p2) @ 'sole LacI transcription
            factor unbinding rate'
'Reverse LacI binding to R0010p3 (LacI bound)' \
        DNA(binding!1,type~BBaR0010p3,upstream!3), LacI(dna!2), DNA(downstream!3,
            binding!2,type~BBaR0010p2), LacI(dna!1,lactose) -> \
        DNA(binding,type~BBaR0010p3,upstream!2), LacI(dna!1), DNA(downstream!2,
            binding!1,type~BBaR0010p2), LacI(dna,lactose) @ 'non-sole LacI
            transcription factor unbinding rate'
'Reverse TetR binding to R0040p2 (no TetR)' \
        DNA(binding,type~BBaR0040p3,upstream!3), TetR(dna!1,atc), DNA(downstream
            !3,binding!1,type~BBaR0040p2) -> \
        DNA(binding,type~BBaR0040p3,upstream!2), TetR(dna,atc), DNA(downstream!2,
            binding,type~BBaR0040p2) @ 'sole TetR transcription factor unbinding
            rate'
'Reverse TetR binding to R0040p2 (TetR bound)' \
        DNA(binding!2,type~BBaR0040p3,upstream!3), TetR(dna!2), TetR(dna!1,atc),
            DNA(downstream!3,binding!1,type~BBaR0040p2) -> \
        DNA(binding!1,type~BBaR0040p3,upstream!2), TetR(dna!1), TetR(dna,atc),
            DNA(downstream!2,binding,type~BBaR0040p2) @ 'non-sole TetR
            transcription factor unbinding rate'
'Reverse TetR binding to R0040p3 (no TetR)' \
        DNA(binding!1,type~BBaR0040p3,upstream!3), TetR(dna!1,atc), DNA(
            downstream!3,binding,type~BBaR0040p2) -> \
        DNA(binding,type~BBaR0040p3,upstream!2), TetR(dna,atc), DNA(downstream!2,
            binding,type~BBaR0040p2) @ 'sole TetR transcription factor unbinding
            rate'
'Reverse TetR binding to R0040p3 (TetR bound)' \
        DNA(binding!2,type~BBaR0040p3,upstream!3), TetR(dna!2,atc), TetR(dna!1),
            DNA(downstream!3,binding!1,type~BBaR0040p2) -> \
        DNA(binding,type~BBaR0040p3,upstream!2), TetR(dna,atc), TetR(dna!1), DNA(
            downstream!2,binding!1,type~BBaR0040p2) @ 'non-sole TetR
            transcription factor unbinding rate'
'Reverse cI binding to R0051p2 (no cI)' \
```

```
       DNA(binding,type~BBaR0051p3,upstream!3), cI(dna!1), DNA(downstream!3,
           binding!1,type~BBaR0051p2) -> \
       DNA(binding,type~BBaR0051p3,upstream!2), cI(dna), DNA(downstream!2,
           binding,type~BBaR0051p2) @ 'sole cI transcription factor unbinding
           rate'
'Reverse cI binding to R0051p2 (cI bound)' \
       DNA(binding!2,type~BBaR0051p3,upstream!3), cI(dna!1), cI(dna!2), DNA(
           downstream!3,binding!1,type~BBaR0051p2) -> \
       DNA(binding!1,type~BBaR0051p3,upstream!2), cI(dna), cI(dna!1), DNA(
           downstream!2,binding,type~BBaR0051p2) @ 'non-sole cI transcription
           factor unbinding rate'
'Reverse cI binding to R0051p3 (no cI)' \
       DNA(binding!1,type~BBaR0051p3,upstream!3), cI(dna!1), DNA(downstream!3,
           binding,type~BBaR0051p2) -> \
       DNA(binding,type~BBaR0051p3,upstream!2), cI(dna), DNA(downstream!2,
           binding,type~BBaR0051p2) @ 'sole cI transcription factor unbinding
           rate'
'Reverse cI binding to R0051p3 (cI bound)' \
       DNA(binding!1,type~BBaR0051p3,upstream!3), cI(dna!2), cI(dna!1), DNA(
           downstream!3,binding!2,type~BBaR0051p2) -> \
       DNA(binding,type~BBaR0051p3,upstream!2), cI(dna!1), cI(dna), DNA(
           downstream!2,binding!1,type~BBaR0051p2) @ 'non-sole cI transcription
           factor unbinding rate'
'Reverse OmpR binding to R0082p2' \
       DNA(binding!1,type~BBaR0082p2,downstream!2), DNA(upstream!2,type~
           BBaR0082p3,downstream!3), DNA(upstream!3,type~BBaR0082p4), OmpR(cph8,
           site~p,dna!1) -> \
       DNA(binding,type~BBaR0082p2,downstream!2), DNA(upstream!2,type~BBaR0082p3
           ,downstream!3), DNA(upstream!3,type~BBaR0082p4), OmpR(cph8,site~p,dna
           ) @ 'OmpR transcription factor unbinding rate'
'Reverse OmpR binding to R0082p3' \
       DNA(type~BBaR0082p2,downstream!2), DNA(upstream!2,binding!1,type~
           BBaR0082p3,downstream!3), DNA(upstream!3,type~BBaR0082p4), OmpR(cph8,
           site~p,dna!1) -> \
       DNA(type~BBaR0082p2,downstream!2), DNA(upstream!2,binding,type~BBaR0082p3
           ,downstream!3), DNA(upstream!3,type~BBaR0082p4), OmpR(cph8,site~p,dna
           ) @ 'OmpR transcription factor unbinding rate'
'Reverse OmpR binding to R0082p4' \
       DNA(type~BBaR0082p2,downstream!2), DNA(upstream!2,type~BBaR0082p3,
           downstream!3), DNA(upstream!3,binding!1,type~BBaR0082p4), OmpR(cph8,
           site~p,dna!1) -> \
       DNA(type~BBaR0082p2,downstream!2), DNA(upstream!2,type~BBaR0082p3,
```

```
            downstream!3), DNA(upstream!3,binding,type~BBaR0082p4), OmpR(cph8,
            site~p,dna) @ 'OmpR transcription factor unbinding rate'
'Reverse OmpR binding to R0084p2' \
       DNA(binding!1,type~BBaR0084p2,downstream!2), DNA(upstream!2,type~
            BBaR0084p3,downstream!3), DNA(upstream!3,type~BBaR0084p4), OmpR(cph8,
            site~u,dna!1) -> \
       DNA(binding,type~BBaR0084p2,downstream!2), DNA(upstream!2,type~BBaR0084p3
            ,downstream!3), DNA(upstream!3,type~BBaR0084p4), OmpR(cph8,site~u,dna
            ) @ 'OmpR transcription factor unbinding rate'
'Reverse OmpR binding to R0084p3' \
       DNA(type~BBaR0084p2,downstream!2), DNA(upstream!2,binding!1,type~
            BBaR0084p3,downstream!3), DNA(upstream!3,type~BBaR0084p4), OmpR(cph8,
            site~u,dna!1) -> \
       DNA(type~BBaR0084p2,downstream!2), DNA(upstream!2,binding,type~BBaR0084p3
            ,downstream!3), DNA(upstream!3,type~BBaR0084p4), OmpR(cph8,site~u,dna
            ) @ 'OmpR transcription factor unbinding rate'
'Reverse OmpR binding to R0084p4' \
       DNA(type~BBaR0084p2,downstream!2), DNA(upstream!2,type~BBaR0084p3,
            downstream!3), DNA(upstream!3,binding!1,type~BBaR0084p4), OmpR(cph8,
            site~u,dna!1) -> \
       DNA(type~BBaR0084p2,downstream!2), DNA(upstream!2,type~BBaR0084p3,
            downstream!3), DNA(upstream!3,binding,type~BBaR0084p4), OmpR(cph8,
            site~u,dna) @ 'OmpR transcription factor unbinding rate'
'Reverse LOVTAP binding to K191007' \
       DNA(binding!1,type~BBaK191007p2,downstream!2), DNA(upstream!2,binding,
            type~BBaK191007p3), LOVTAP(active~light,dna!1) -> \
       DNA(binding,type~BBaK191007p2,downstream!2), DNA(upstream!2,binding,type~
            BBaK191007p3), LOVTAP(active~light,dna) @ 'LOVTAP transcription
            factor unbinding rate'
'Reverse PhoB binding to R0PHO' \
       DNA(binding!1,type~BBaR0PHOp2,downstream!2), DNA(upstream!2,binding,type~
            BBaR0PHOp3), PhoB(ccas,site~p,dna!1) -> \
       DNA(binding,type~BBaR0PHOp2,downstream!2), DNA(upstream!2,binding,type~
            BBaR0PHOp3), PhoB(ccas,site~p,dna) @ 'PhoB transcription factor
            unbinding rate'

# RNAP binding to promoter regions.

%var: 'high RNAP binding rate' 0.0007
%var: 'low RNAP binding rate' 7e-07

'RNAP binding to R0010 (no LacI)' \
```

```
       DNA(binding,type~BBaR0010p3,upstream!2,downstream!1), DNA(upstream!1,
           binding,type~BBaR0010p4), RNAP(dna,rna), DNA(downstream!2,binding,
           type~BBaR0010p2) -> \
       DNA(binding,type~BBaR0010p3,upstream!3,downstream!1), DNA(upstream!1,
           binding!2,type~BBaR0010p4), RNAP(dna!2,rna), DNA(downstream!3,binding
           ,type~BBaR0010p2) @ 'high RNAP binding rate'
'RNAP binding to R0010 (LacI on p2)' \
       DNA(binding,type~BBaR0010p3,upstream!2,downstream!1), DNA(upstream!1,
           binding,type~BBaR0010p4), RNAP(dna,rna), DNA(downstream!2,binding!3,
           type~BBaR0010p2), LacI(dna!3) -> \
       DNA(binding,type~BBaR0010p3,upstream!3,downstream!1), DNA(upstream!1,
           binding!2,type~BBaR0010p4), RNAP(dna!2,rna), DNA(downstream!3,binding
           !4,type~BBaR0010p2), LacI(dna!4) @ 'low RNAP binding rate'
'RNAP binding to R0010 (LacI on p3)' \
       DNA(binding!3,type~BBaR0010p3,upstream!2,downstream!1), DNA(upstream!1,
           binding,type~BBaR0010p4), RNAP(dna,rna), DNA(downstream!2,binding,
           type~BBaR0010p2), LacI(dna!3) -> \
       DNA(binding!4,type~BBaR0010p3,upstream!3,downstream!1), DNA(upstream!1,
           binding!2,type~BBaR0010p4), RNAP(dna!2,rna), DNA(downstream!3,binding
           ,type~BBaR0010p2), LacI(dna!4) @ 'low RNAP binding rate'
'RNAP binding to R0010 (LacI on p2 and p3)' \
       DNA(binding!3,type~BBaR0010p3,upstream!2,downstream!1), DNA(upstream!1,
           binding,type~BBaR0010p4), RNAP(dna,rna), DNA(downstream!2,binding!4,
           type~BBaR0010p2), LacI(dna!3), LacI(dna!4) -> \
       DNA(binding!4,type~BBaR0010p3,upstream!3,downstream!1), DNA(upstream!1,
           binding!2,type~BBaR0010p4), RNAP(dna!2,rna), DNA(downstream!3,binding
           !5,type~BBaR0010p2), LacI(dna!4), LacI(dna!5) @ 'low RNAP binding
           rate'
'RNAP binding to R0040 (no TetR)' \
       DNA(binding,type~BBaR0040p3,upstream!2,downstream!1), DNA(upstream!1,
           binding,type~BBaR0040p4), RNAP(dna,rna), DNA(downstream!2,binding,
           type~BBaR0040p2) -> \
       DNA(binding,type~BBaR0040p3,upstream!3,downstream!1), DNA(upstream!1,
           binding!2,type~BBaR0040p4), RNAP(dna!2,rna), DNA(downstream!3,binding
           ,type~BBaR0040p2) @ 'high RNAP binding rate'
'RNAP binding to R0040 (TetR on p2)' \
       DNA(binding,type~BBaR0040p3,upstream!2,downstream!1), DNA(upstream!1,
           binding,type~BBaR0040p4), RNAP(dna,rna), DNA(downstream!2,binding!3,
           type~BBaR0040p2), TetR(dna!3) -> \
       DNA(binding,type~BBaR0040p3,upstream!3,downstream!1), DNA(upstream!1,
           binding!2,type~BBaR0040p4), RNAP(dna!2,rna), DNA(downstream!3,binding
           !4,type~BBaR0040p2), TetR(dna!4) @ 'low RNAP binding rate'
```

```
'RNAP binding to R0040 (TetR on p3)' \
       DNA(binding!3,type~BBaR0040p3,upstream!2,downstream!1), DNA(upstream!1,
           binding,type~BBaR0040p4), RNAP(dna,rna), DNA(downstream!2,binding,
           type~BBaR0040p2), TetR(dna!3) -> \
       DNA(binding!4,type~BBaR0040p3,upstream!3,downstream!1), DNA(upstream!1,
           binding!2,type~BBaR0040p4), RNAP(dna!2,rna), DNA(downstream!3,binding
           ,type~BBaR0040p2), TetR(dna!4) @ 'low RNAP binding rate'
'RNAP binding to R0040 (TetR on p2 and p3)' \
       DNA(binding!3,type~BBaR0040p3,upstream!2,downstream!1), DNA(upstream!1,
           binding,type~BBaR0040p4), RNAP(dna,rna), DNA(downstream!2,binding!4,
           type~BBaR0040p2), TetR(dna!3), TetR(dna!4) -> \
       DNA(binding!4,type~BBaR0040p3,upstream!3,downstream!1), DNA(upstream!1,
           binding!2,type~BBaR0040p4), RNAP(dna!2,rna), DNA(downstream!3,binding
           !5,type~BBaR0040p2), TetR(dna!4), TetR(dna!5) @ 'low RNAP binding
           rate'
'RNAP binding to R0051 (no cI)' \
       DNA(binding,type~BBaR0051p3,upstream!2,downstream!1), DNA(upstream!1,
           binding,type~BBaR0051p4), RNAP(dna,rna), DNA(downstream!2,binding,
           type~BBaR0051p2) -> \
       DNA(binding,type~BBaR0051p3,upstream!3,downstream!1), DNA(upstream!1,
           binding!2,type~BBaR0051p4), RNAP(dna!2,rna), DNA(downstream!3,binding
           ,type~BBaR0051p2) @ 'high RNAP binding rate'
'RNAP binding to R0051 (cI on p2)' \
       DNA(binding,type~BBaR0051p3,upstream!2,downstream!1), DNA(upstream!1,
           binding,type~BBaR0051p4), RNAP(dna,rna), DNA(downstream!2,binding!3,
           type~BBaR0051p2), cI(dna!3) -> \
       DNA(binding,type~BBaR0051p3,upstream!3,downstream!1), DNA(upstream!1,
           binding!2,type~BBaR0051p4), RNAP(dna!2,rna), DNA(downstream!3,binding
           !4,type~BBaR0051p2), cI(dna!4) @ 'low RNAP binding rate'
'RNAP binding to R0051 (cI on p3)' \
       DNA(binding!3,type~BBaR0051p3,upstream!2,downstream!1), DNA(upstream!1,
           binding,type~BBaR0051p4), RNAP(dna,rna), DNA(downstream!2,binding,
           type~BBaR0051p2), cI(dna!3) -> \
       DNA(binding!4,type~BBaR0051p3,upstream!3,downstream!1), DNA(upstream!1,
           binding!2,type~BBaR0051p4), RNAP(dna!2,rna), DNA(downstream!3,binding
           ,type~BBaR0051p2), cI(dna!4) @ 'low RNAP binding rate'
'RNAP binding to R0051 (cI on p2 and p3)' \
       DNA(binding!3,type~BBaR0051p3,upstream!2,downstream!1), DNA(upstream!1,
           binding,type~BBaR0051p4), RNAP(dna,rna), DNA(downstream!2,binding!4,
           type~BBaR0051p2), cI(dna!3), cI(dna!4) -> \
       DNA(binding!4,type~BBaR0051p3,upstream!3,downstream!1), DNA(upstream!1,
           binding!2,type~BBaR0051p4), RNAP(dna!2,rna), DNA(downstream!3,binding
```

```
          !5,type~BBaR0051p2), cI(dna!4), cI(dna!5) @ 'low RNAP binding rate'
'RNAP binding to R0082 (no OmpR)' \
       DNA(binding,type~BBaR0082p2,downstream!1), DNA(upstream!1,binding,type~
           BBaR0082p3,downstream!2), DNA(upstream!2,binding,type~BBaR0082p4,
           downstream!3), DNA(upstream!3,binding,type~BBaR0082p5), \
       RNAP(dna,rna) -> \
       DNA(binding,type~BBaR0082p2,downstream!1), DNA(upstream!1,binding,type~
           BBaR0082p3,downstream!2), DNA(upstream!2,binding,type~BBaR0082p4,
           downstream!3), DNA(upstream!3,binding!4,type~BBaR0082p5), \
       RNAP(dna!4,rna) @ 'low RNAP binding rate'
'RNAP binding to R0082 (OmpR on p2)' \
       DNA(binding!5,type~BBaR0082p2,downstream!1), DNA(upstream!1,binding,type~
           BBaR0082p3,downstream!2), DNA(upstream!2,binding,type~BBaR0082p4,
           downstream!3), DNA(upstream!3,binding,type~BBaR0082p5), \
       RNAP(dna,rna), OmpR(dna!5) -> \
       DNA(binding!5,type~BBaR0082p2,downstream!1), DNA(upstream!1,binding,type~
           BBaR0082p3,downstream!2), DNA(upstream!2,binding,type~BBaR0082p4,
           downstream!3), DNA(upstream!3,binding!4,type~BBaR0082p5), \
       RNAP(dna!4,rna), OmpR(dna!5) @ 'low RNAP binding rate'
'RNAP binding to R0082 (OmpR on p3)' \
       DNA(binding,type~BBaR0082p2,downstream!1), DNA(upstream!1,binding!5,type~
           BBaR0082p3,downstream!2), DNA(upstream!2,binding,type~BBaR0082p4,
           downstream!3), DNA(upstream!3,binding,type~BBaR0082p5), \
       RNAP(dna,rna), OmpR(dna!5) -> \
       DNA(binding,type~BBaR0082p2,downstream!1), DNA(upstream!1,binding!5,type~
           BBaR0082p3,downstream!2), DNA(upstream!2,binding,type~BBaR0082p4,
           downstream!3), DNA(upstream!3,binding!4,type~BBaR0082p5), \
       RNAP(dna!4,rna), OmpR(dna!5) @ 'low RNAP binding rate'
'RNAP binding to R0082 (OmpR on p4)' \
       DNA(binding,type~BBaR0082p2,downstream!1), DNA(upstream!1,binding,type~
           BBaR0082p3,downstream!2), DNA(upstream!2,binding!5,type~BBaR0082p4,
           downstream!3), DNA(upstream!3,binding,type~BBaR0082p5), \
       RNAP(dna,rna), OmpR(dna!5) -> \
       DNA(binding,type~BBaR0082p2,downstream!1), DNA(upstream!1,binding,type~
           BBaR0082p3,downstream!2), DNA(upstream!2,binding!5,type~BBaR0082p4,
           downstream!3), DNA(upstream!3,binding!4,type~BBaR0082p5), \
       RNAP(dna!4,rna), OmpR(dna!5) @ 'low RNAP binding rate'
'RNAP binding to R0082 (OmpR on p2 and p3)' \
       DNA(binding!5,type~BBaR0082p2,downstream!1), DNA(upstream!1,binding!6,
           type~BBaR0082p3,downstream!2), DNA(upstream!2,binding,type~BBaR0082p4
           ,downstream!3), DNA(upstream!3,binding,type~BBaR0082p5), \
       RNAP(dna,rna), OmpR(dna!5), OmpR(dna!6) -> \
```

```
       DNA(binding!5,type~BBaR0082p2,downstream!1), DNA(upstream!1,binding!6,
           type~BBaR0082p3,downstream!2), DNA(upstream!2,binding,type~BBaR0082p4
           ,downstream!3), DNA(upstream!3,binding!4,type~BBaR0082p5), \
       RNAP(dna!4,rna), OmpR(dna!5), OmpR(dna!6) @ 'high RNAP binding rate'
'RNAP binding to R0082 (OmpR on p2 and p4)' \
       DNA(binding!5,type~BBaR0082p2,downstream!1), DNA(upstream!1,binding,type~
           BBaR0082p3,downstream!2), DNA(upstream!2,binding!6,type~BBaR0082p4,
           downstream!3), DNA(upstream!3,binding,type~BBaR0082p5), \
       RNAP(dna,rna), OmpR(dna!5), OmpR(dna!6) -> \
       DNA(binding!5,type~BBaR0082p2,downstream!1), DNA(upstream!1,binding,type~
           BBaR0082p3,downstream!2), DNA(upstream!2,binding!6,type~BBaR0082p4,
           downstream!3), DNA(upstream!3,binding!4,type~BBaR0082p5), \
       RNAP(dna!4,rna), OmpR(dna!5), OmpR(dna!6) @ 'high RNAP binding rate'
'RNAP binding to R0082 (OmpR on p3 and p4)' \
       DNA(binding,type~BBaR0082p2,downstream!1), DNA(upstream!1,binding!5,type~
           BBaR0082p3,downstream!2), DNA(upstream!2,binding!6,type~BBaR0082p4,
           downstream!3), DNA(upstream!3,binding,type~BBaR0082p5), \
       RNAP(dna,rna), OmpR(dna!5), OmpR(dna!6) -> \
       DNA(binding,type~BBaR0082p2,downstream!1), DNA(upstream!1,binding!5,type~
           BBaR0082p3,downstream!2), DNA(upstream!2,binding!6,type~BBaR0082p4,
           downstream!3), DNA(upstream!3,binding!4,type~BBaR0082p5), \
       RNAP(dna!4,rna), OmpR(dna!5), OmpR(dna!6) @ 'high RNAP binding rate'
'RNAP binding to R0082 (OmpR on p2 and p3 and p4)' \
       DNA(binding!5,type~BBaR0082p2,downstream!1), DNA(upstream!1,binding!6,
           type~BBaR0082p3,downstream!2), DNA(upstream!2,binding!7,type~
           BBaR0082p4,downstream!3), DNA(upstream!3,binding,type~BBaR0082p5), \
       RNAP(dna,rna), OmpR(dna!5), OmpR(dna!6), OmpR(dna!7) -> \
       DNA(binding!5,type~BBaR0082p2,downstream!1), DNA(upstream!1,binding!6,
           type~BBaR0082p3,downstream!2), DNA(upstream!2,binding!7,type~
           BBaR0082p4,downstream!3), DNA(upstream!3,binding!4,type~BBaR0082p5),
           \
       RNAP(dna!4,rna), OmpR(dna!5), OmpR(dna!6), OmpR(dna!7) @ 'high RNAP
           binding rate'
'RNAP binding to R0084 (no OmpR)' \
       DNA(binding,type~BBaR0084p2,downstream!1), DNA(upstream!1,binding,type~
           BBaR0084p3,downstream!2), DNA(upstream!2,binding,type~BBaR0084p4,
           downstream!3), DNA(upstream!3,binding,type~BBaR0084p5), RNAP(dna,rna)
            -> \
       DNA(binding,type~BBaR0084p2,downstream!1), DNA(upstream!1,binding,type~
           BBaR0084p3,downstream!2), DNA(upstream!2,binding,type~BBaR0084p4,
           downstream!3), DNA(upstream!3,binding!4,type~BBaR0084p5), RNAP(dna!4,
           rna) @ 'high RNAP binding rate'
```

```
'RNAP binding to R0084 (OmpR on p2)' \
       DNA(binding!5,type~BBaR0084p2,downstream!1), DNA(upstream!1,binding,type~
           BBaR0084p3,downstream!2), DNA(upstream!2,binding,type~BBaR0084p4,
           downstream!3), DNA(upstream!3,binding,type~BBaR0084p5), \
       RNAP(dna,rna), OmpR(dna!5) -> \
       DNA(binding!5,type~BBaR0084p2,downstream!1), DNA(upstream!1,binding,type~
           BBaR0084p3,downstream!2), DNA(upstream!2,binding,type~BBaR0084p4,
           downstream!3), DNA(upstream!3,binding!4,type~BBaR0084p5), \
       RNAP(dna!4,rna), OmpR(dna!5) @ 'high RNAP binding rate'
'RNAP binding to R0084 (OmpR on p3)' \
       DNA(binding,type~BBaR0084p2,downstream!1), DNA(upstream!1,binding!5,type~
           BBaR0084p3,downstream!2), DNA(upstream!2,binding,type~BBaR0084p4,
           downstream!3), DNA(upstream!3,binding,type~BBaR0084p5), \
       RNAP(dna,rna), OmpR(dna!5) -> \
       DNA(binding,type~BBaR0084p2,downstream!1), DNA(upstream!1,binding!5,type~
           BBaR0084p3,downstream!2), DNA(upstream!2,binding,type~BBaR0084p4,
           downstream!3), DNA(upstream!3,binding!4,type~BBaR0084p5), \
       RNAP(dna!4,rna), OmpR(dna!5) @ 'high RNAP binding rate'
'RNAP binding to R0084 (OmpR on p4)' \
       DNA(binding,type~BBaR0084p2,downstream!1), DNA(upstream!1,binding,type~
           BBaR0084p3,downstream!2), DNA(upstream!2,binding!5,type~BBaR0084p4,
           downstream!3), DNA(upstream!3,binding,type~BBaR0084p5), \
       RNAP(dna,rna), OmpR(dna!5) -> \
       DNA(binding,type~BBaR0084p2,downstream!1), DNA(upstream!1,binding,type~
           BBaR0084p3,downstream!2), DNA(upstream!2,binding!5,type~BBaR0084p4,
           downstream!3), DNA(upstream!3,binding!4,type~BBaR0084p5), \
       RNAP(dna!4,rna), OmpR(dna!5) @ 'high RNAP binding rate'
'RNAP binding to R0084 (OmpR on p2 and p3)' \
       DNA(binding!5,type~BBaR0084p2,downstream!1), DNA(upstream!1,binding!6,
           type~BBaR0084p3,downstream!2), DNA(upstream!2,binding,type~BBaR0084p4
           ,downstream!3), DNA(upstream!3,binding,type~BBaR0084p5), \
       RNAP(dna,rna), OmpR(dna!5), OmpR(dna!6) -> \
       DNA(binding!5,type~BBaR0084p2,downstream!1), DNA(upstream!1,binding!6,
           type~BBaR0084p3,downstream!2), DNA(upstream!2,binding,type~BBaR0084p4
           ,downstream!3), DNA(upstream!3,binding!4,type~BBaR0084p5), \
       RNAP(dna!4,rna), OmpR(dna!5), OmpR(dna!6) @ 'low RNAP binding rate'
'RNAP binding to R0084 (OmpR on p2 and p4)' \
       DNA(binding!5,type~BBaR0084p2,downstream!1), DNA(upstream!1,binding,type~
           BBaR0084p3,downstream!2), DNA(upstream!2,binding!6,type~BBaR0084p4,
           downstream!3), DNA(upstream!3,binding,type~BBaR0084p5), \
       RNAP(dna,rna), OmpR(dna!5), OmpR(dna!6) -> \
       DNA(binding!5,type~BBaR0084p2,downstream!1), DNA(upstream!1,binding,type~
```

```
            BBaR0084p3,downstream!2), DNA(upstream!2,binding!6,type~BBaR0084p4,
            downstream!3), DNA(upstream!3,binding!4,type~BBaR0084p5), \
        RNAP(dna!4,rna), OmpR(dna!5), OmpR(dna!6) @ 'low RNAP binding rate'
'RNAP binding to R0084 (OmpR on p3 and p4)' \
        DNA(binding,type~BBaR0084p2,downstream!1), DNA(upstream!1,binding!5,type~
            BBaR0084p3,downstream!2), DNA(upstream!2,binding!6,type~BBaR0084p4,
            downstream!3), DNA(upstream!3,binding,type~BBaR0084p5), \
        RNAP(dna,rna), OmpR(dna!5), OmpR(dna!6) -> \
        DNA(binding,type~BBaR0084p2,downstream!1), DNA(upstream!1,binding!5,type~
            BBaR0084p3,downstream!2), DNA(upstream!2,binding!6,type~BBaR0084p4,
            downstream!3), DNA(upstream!3,binding!4,type~BBaR0084p5), \
        RNAP(dna!4,rna), OmpR(dna!5), OmpR(dna!6) @ 'low RNAP binding rate'
'RNAP binding to R0084 (OmpR on p2 and p3 and p4)' \
        DNA(binding!5,type~BBaR0084p2,downstream!1), DNA(upstream!1,binding!6,
            type~BBaR0084p3,downstream!2), DNA(upstream!2,binding!7,type~
            BBaR0084p4,downstream!3), DNA(upstream!3,binding,type~BBaR0084p5), \
        RNAP(dna,rna), OmpR(dna!5), OmpR(dna!6), OmpR(dna!7) -> \
        DNA(binding!5,type~BBaR0084p2,downstream!1), DNA(upstream!1,binding!6,
            type~BBaR0084p3,downstream!2), DNA(upstream!2,binding!7,type~
            BBaR0084p4,downstream!3), DNA(upstream!3,binding!4,type~BBaR0084p5),
            \
        RNAP(dna!4,rna), OmpR(dna!5), OmpR(dna!6), OmpR(dna!7) @ 'low RNAP
            binding rate'
'RNAP binding to K191007 (no LOVTAP) ' \
        DNA(binding,type~BBaK191007p2,downstream!1), DNA(upstream!1,binding,type~
            BBaK191007p3), RNAP(dna,rna) -> \
        DNA(binding,type~BBaK191007p2,downstream!1), DNA(upstream!1,binding!2,
            type~BBaK191007p3), RNAP(dna!2,rna) @ 'high RNAP binding rate'
'RNAP binding to K191007 (LOVTAP on p2) ' \
        DNA(binding!1,type~BBaK191007p2,downstream!2), DNA(upstream!2,binding,
            type~BBaK191007p3), LOVTAP(dna!1), RNAP(dna,rna) -> \
        DNA(binding!1,type~BBaK191007p2,downstream!2), DNA(upstream!2,binding!3,
            type~BBaK191007p3), LOVTAP(dna!1), RNAP(dna!3,rna) @ 'low RNAP
            binding rate'
'RNAP binding to R0PHO (no PhoB) ' \
        DNA(binding,type~BBaR0PHOp2,downstream!1), DNA(upstream!1,binding,type~
            BBaR0PHOp3), RNAP(dna,rna) -> \
        DNA(binding,type~BBaR0PHOp2,downstream!1), DNA(upstream!1,binding!2,type~
            BBaR0PHOp3), RNAP(dna!2,rna) @ 'high RNAP binding rate'
'RNAP binding to R0PHO (PhoB on p2) ' \
        DNA(binding!1,type~BBaR0PHOp2,downstream!2), DNA(upstream!2,binding,type~
            BBaR0PHOp3), PhoB(dna!1), RNAP(dna,rna) -> \
```

```
       DNA(binding!1,type~BBaR0PHOp2,downstream!2), DNA(upstream!2,binding!3,
           type~BBaR0PHOp3), PhoB(dna!1), RNAP(dna!3,rna) @ 'low RNAP binding
           rate'



# Transcription.

%var: 'transcription initiation rate' 10
%var: 'transcription rate' 10
%var: 'transcription termination rate' 10

'Transcription initiation of R0051' \
       DNA(binding!1,type~BBaR0051p4,downstream!2), RNAP(dna!1,rna), DNA(
           upstream!2,binding) -> \
       DNA(binding,type~BBaR0051p4,downstream!3), RNAP(dna!1,rna!2), DNA(
           upstream!3,binding!1), RNA(binding,upstream,downstream!2,type~
           BBaR0051) @ 'transcription initiation rate'
'Transcription initiation of R0010' \
       DNA(binding!1,type~BBaR0010p4,downstream!2), RNAP(dna!1,rna), DNA(
           upstream!2,binding) -> \
       DNA(binding,type~BBaR0010p4,downstream!3), RNAP(dna!1,rna!2), DNA(
           upstream!3,binding!1), RNA(binding,upstream,downstream!2,type~
           BBaR0010) @ 'transcription initiation rate'
'Transcription initiation of R0040' \
       DNA(binding!1,type~BBaR0040p4,downstream!2), RNAP(dna!1,rna), DNA(
           upstream!2,binding) -> \
       DNA(binding,type~BBaR0040p4,downstream!3), RNAP(dna!1,rna!2), DNA(
           upstream!3,binding!1), RNA(binding,upstream,downstream!2,type~
           BBaR0040) @ 'transcription initiation rate'
'Transcription initiation of R0082' \
       DNA(binding!1,type~BBaR0082p5,downstream!2), RNAP(dna!1,rna), DNA(
           upstream!2,binding) -> \
       DNA(binding,type~BBaR0082p5,downstream!3), RNAP(dna!1,rna!2), DNA(
           upstream!3,binding!1), RNA(binding,upstream,downstream!2,type~
           BBaR0082) @ 'transcription initiation rate'
'Transcription initiation of R0084' \
       DNA(binding!1,type~BBaR0084p5,downstream!2), RNAP(dna!1,rna), DNA(
           upstream!2,binding) -> \
       DNA(binding,type~BBaR0084p5,downstream!3), RNAP(dna!1,rna!2), DNA(
           upstream!3,binding!1), RNA(binding,upstream,downstream!2,type~
           BBaR0084) @ 'transcription initiation rate'
```

```
'Transcription initiation of K191007' \
       DNA(binding!1,type~BBaK191007p3,downstream!2), RNAP(dna!1,rna), DNA(
           upstream!2,binding) -> \
       DNA(binding,type~BBaK191007p3,downstream!3), RNAP(dna!1,rna!2), DNA(
           upstream!3,binding!1), RNA(binding,upstream,downstream!2,type~
           BBaK191007) @ 'transcription initiation rate'
'Transcription initiation of R0PHO' \
       DNA(binding!1,type~BBaR0PHOp3,downstream!2), RNAP(dna!1,rna), DNA(
           upstream!2,binding) -> \
       DNA(binding,type~BBaR0PHOp3,downstream!3), RNAP(dna!1,rna!2), DNA(
           upstream!3,binding!1), RNA(binding,upstream,downstream!2,type~
           BBaR0PHO) @ 'transcription initiation rate'

'RBS BBa_B0034 transcription' \
       DNA(binding!1,downstream!2,type~BBaB0034), RNAP(dna!1,rna!3), DNA(
           upstream!2,binding), RNA(downstream!3) -> \
       DNA(binding,downstream!2,type~BBaB0034), RNAP(dna!1,rna!3), DNA(upstream
           !2,binding!1), RNA(downstream!4), RNA(binding,upstream!4,downstream
           !3,type~BBaB0034) @ 'transcription rate'

'C0012 transcription' \
       DNA(binding!1,downstream!2,type~BBaC0012), RNAP(dna!1,rna!3), DNA(
           upstream!2,binding), RNA(downstream!3) -> \
       DNA(binding,downstream!2,type~BBaC0012), RNAP(dna!1,rna!3), DNA(upstream
           !2,binding!1), RNA(downstream!4), RNA(binding,upstream!4,downstream
           !3,type~BBaC0012) @ 'transcription rate'
'C0051 transcription' \
       DNA(binding!1,downstream!2,type~BBaC0051), RNAP(dna!1,rna!3), DNA(
           upstream!2,binding), RNA(downstream!3) -> \
       DNA(binding,downstream!2,type~BBaC0051), RNAP(dna!1,rna!3), DNA(upstream
           !2,binding!1), RNA(downstream!4), RNA(binding,upstream!4,downstream
           !3,type~BBaC0051) @ 'transcription rate'
'C0040 transcription' \
       DNA(binding!1,downstream!2,type~BBaC0040), RNAP(dna!1,rna!3), DNA(
           upstream!2,binding), RNA(downstream!3) -> \
       DNA(binding,downstream!2,type~BBaC0040), RNAP(dna!1,rna!3), DNA(upstream
           !2,binding!1), RNA(downstream!4), RNA(binding,upstream!4,downstream
           !3,type~BBaC0040) @ 'transcription rate'
'C0RED transcription' \
       DNA(binding!1,downstream!2,type~BBaC0RED), RNAP(dna!1,rna!3), DNA(
           upstream!2,binding), RNA(downstream!3) -> \
       DNA(binding,downstream!2,type~BBaC0RED), RNAP(dna!1,rna!3), DNA(upstream
```

```
            !2,binding!1), RNA(downstream!4), RNA(binding,upstream!4,downstream
            !3,type~BBaC0RED) @ 'transcription rate'
'C0BLUE transcription' \
       DNA(binding!1,downstream!2,type~BBaC0BLUE), RNAP(dna!1,rna!3), DNA(
           upstream!2,binding), RNA(downstream!3) -> \
       DNA(binding,downstream!2,type~BBaC0BLUE), RNAP(dna!1,rna!3), DNA(upstream
           !2,binding!1), RNA(downstream!4), RNA(binding,upstream!4,downstream
           !3,type~BBaC0BLUE) @ 'transcription rate'
'C0GREEN transcription' \
       DNA(binding!1,downstream!2,type~BBaC0GREEN), RNAP(dna!1,rna!3), DNA(
           upstream!2,binding), RNA(downstream!3) -> \
       DNA(binding,downstream!2,type~BBaC0GREEN), RNAP(dna!1,rna!3), DNA(
           upstream!2,binding!1), RNA(downstream!4), RNA(binding,upstream!4,
           downstream!3,type~BBaC0GREEN) @ 'transcription rate'

'Termination - B0011' \
       DNA(binding!1,type~BBaB0011), RNAP(dna!1,rna!2), RNA(downstream!2) -> \
       DNA(binding,type~BBaB0011), RNAP(dna,rna), RNA(downstream) @ '
           transcription termination rate'

%var: 'RNAP falloff rate' 1.0

'RNAP falloff' \
       DNA(binding!1,downstream!3), RNAP(dna!1,rna!2), RNA(downstream!2), DNA(
           upstream!3,binding!_) -> \
       DNA(binding,downstream!1), RNAP(dna,rna), RNA(downstream), DNA(upstream
           !1,binding!_) @ 'RNAP falloff rate'

# The following rule is not invoked in this particular model as BBaB0011 is
    always the last DNA agent in the sequence, hence no downstream agent
#'B0011 terminator transcription (readthrough)' \
#      DNA(binding!1,downstream!2,type~BBaB0011), RNAP(dna!1,rna!3), DNA(
    upstream!2,binding), RNA(downstream!3) -> \
#      DNA(binding,downstream!2,type~BBaB0011), RNAP(dna!1,rna!3), DNA(upstream
    !2,binding!1), RNA(downstream!4), RNA(binding,upstream!4,downstream!3,type~
    BBaB0011) @ 0.5

'Transcription of R0051 (readthrough)' \
       DNA(binding,type~BBaR0051p3,downstream!2,upstream!3), RNAP(dna!1,rna!5),
           DNA(upstream!6,binding), DNA(upstream!4,downstream!3,binding,type~
           BBaR0051p2), DNA(downstream!4,binding!1,type~BBaR0051p1), \
       RNA(downstream!5), DNA(upstream!2,downstream!6,binding,type~BBaR0051p4)
```

```
            -> \
        DNA(binding,type~BBaR0051p3,downstream!3,upstream!5), RNAP(dna!1,rna!6),
            DNA(upstream!7,binding!1), DNA(upstream!4,downstream!5,binding,type~
            BBaR0051p2), DNA(downstream!4,binding,type~BBaR0051p1), \
        RNA(downstream!2), DNA(upstream!3,downstream!7,binding,type~BBaR0051p4),
            RNA(binding,upstream!2,downstream!6,type~BBaR0051) @ 'transcription
            rate'
'Transcription of R0010 (readthrough)' \
        DNA(binding,type~BBaR0010p3,downstream!2,upstream!3), RNAP(dna!1,rna!5),
            DNA(upstream!6,binding), DNA(upstream!4,downstream!3,binding,type~
            BBaR0010p2), DNA(downstream!4,binding!1,type~BBaR0010p1), \
        RNA(downstream!5), DNA(upstream!2,downstream!6,binding,type~BBaR0010p4)
            -> \
        DNA(binding,type~BBaR0010p3,downstream!3,upstream!5), RNAP(dna!1,rna!6),
            DNA(upstream!7,binding!1), DNA(upstream!4,downstream!5,binding,type~
            BBaR0010p2), DNA(downstream!4,binding,type~BBaR0010p1), \
        RNA(downstream!2), DNA(upstream!3,downstream!7,binding,type~BBaR0010p4),
            RNA(binding,upstream!2,downstream!6,type~BBaR0010) @ 'transcription
            rate'
'Transcription of R0040 (readthrough)' \
        DNA(binding,type~BBaR0040p3,downstream!2,upstream!3), RNAP(dna!1,rna!5),
            DNA(upstream!6,binding), DNA(upstream!4,downstream!3,binding,type~
            BBaR0040p2), DNA(downstream!4,binding!1,type~BBaR0040p1), \
        RNA(downstream!5), DNA(upstream!2,downstream!6,binding,type~BBaR0040p4)
            -> \
        DNA(binding,type~BBaR0040p3,downstream!3,upstream!5), RNAP(dna!1,rna!6),
            DNA(upstream!7,binding!1), DNA(upstream!4,downstream!5,binding,type~
            BBaR0040p2), DNA(downstream!4,binding,type~BBaR0040p1), \
        RNA(downstream!2), DNA(upstream!3,downstream!7,binding,type~BBaR0040p4),
            RNA(binding,upstream!2,downstream!6,type~BBaR0040) @ 'transcription
            rate'
'Transcription of R0082 (readthrough)' \
        DNA(binding!1,type~BBaR0082p1,downstream!2), DNA(upstream!2,binding,type~
            BBaR0082p2,downstream!3), DNA(upstream!3,binding,type~BBaR0082p3,
            downstream!4), DNA(upstream!4,binding,type~BBaR0082p4,downstream!5),
            \
        DNA(upstream!5,binding,type~BBaR0082p5,downstream!6), DNA(upstream!6,
            binding), RNAP(dna!1,rna!7), RNA(downstream!7) -> \
        DNA(binding,type~BBaR0082p1,downstream!2), DNA(upstream!2,binding,type~
            BBaR0082p2,downstream!3), DNA(upstream!3,binding,type~BBaR0082p3,
            downstream!4), DNA(upstream!4,binding,type~BBaR0082p4,downstream!5),
            \
```

```
        DNA(upstream!5,binding,type~BBaR0082p5,downstream!6), DNA(upstream!6,
            binding!1), RNAP(dna!1,rna!7),RNA(downstream!8),RNA(binding,upstream
            !8,downstream!7,type~BBaR0082) @ 'transcription rate'
'Transcription of R0084 (readthrough)' \
        DNA(binding!1,type~BBaR0084p1,downstream!2), DNA(upstream!2,binding,type~
            BBaR0084p2,downstream!3), DNA(upstream!3,binding,type~BBaR0084p3,
            downstream!4), \
        DNA(upstream!4,binding,type~BBaR0084p4,downstream!5), DNA(upstream!5,
            binding,type~BBaR0084p5,downstream!6), DNA(upstream!6,binding), RNAP(
            dna!1,rna!7), RNA(downstream!7) -> \
        DNA(binding,type~BBaR0084p1,downstream!2), DNA(upstream!2,binding,type~
            BBaR0084p2,downstream!3), DNA(upstream!3,binding,type~BBaR0084p3,
            downstream!4), \
        DNA(upstream!4,binding,type~BBaR0084p4,downstream!5), DNA(upstream!5,
            binding,type~BBaR0084p5,downstream!6), DNA(upstream!6,binding!1),
            RNAP(dna!1,rna!7),RNA(downstream!8), \
        RNA(binding,upstream!8,downstream!7,type~BBaR0084) @ 'transcription rate'
'Transcription of K191007 (readthrough)' \
        DNA(binding!1,type~BBaK191007p1,downstream!2), DNA(upstream!2,binding,
            type~BBaK191007p2,downstream!3), DNA(upstream!3,binding,type~
            BBaK191007p3,downstream!4), DNA(upstream!4,binding), \
        RNAP(dna!1,rna!5), RNA(downstream!5) -> \
        DNA(binding,type~BBaK191007p1,downstream!2), DNA(upstream!2,binding,type~
            BBaK191007p2,downstream!3), DNA(upstream!3,binding,type~BBaK191007p3,
            downstream!4), DNA(upstream!4,binding!1), \
        RNAP(dna!1,rna!5), RNA(downstream!6),RNA(binding,upstream!6,downstream!5,
            type~BBaK191007) @ 'transcription rate'
'Transcription of R0PHO (readthrough)' \
        DNA(binding!1,type~BBaR0PHOp1,downstream!2), DNA(upstream!2,binding,type~
            BBaR0PHOp2,downstream!3), DNA(upstream!3,binding,type~BBaR0PHOp3,
            downstream!4), DNA(upstream!4,binding), \
        RNAP(dna!1,rna!5), RNA(downstream!5) -> \
        DNA(binding,type~BBaR0PHOp1,downstream!2), DNA(upstream!2,binding,type~
            BBaR0PHOp2,downstream!3), DNA(upstream!3,binding,type~BBaR0PHOp3,
            downstream!4), DNA(upstream!4,binding!1), \
        RNAP(dna!1,rna!5), RNA(downstream!6),RNA(binding,upstream!6,downstream!5,
            type~BBaR0PHO) @ 'transcription rate'


# Translation.

%var: 'ribosome binding rate' 0.000166
```

```
'RBS BBa_B0034 Ribosome binding' \
       RNA(binding,type~BBaB0034), Ribosome(rna) -> \
       RNA(binding!1,type~BBaB0034), Ribosome(rna!1) @ 'ribosome binding rate'

%var: 'translation initiation rate' 0.167

'LacI translation initiation' \
       RNA(binding!2,downstream!1), RNA(binding,upstream!1,type~BBaC0012),
           Ribosome(rna!2) -> \
       RNA(binding,downstream!1), RNA(binding!2,upstream!1,type~BBaC0012),
           Ribosome(rna!2) @ 'translation initiation rate'
'TetR translation initiation' \
       RNA(binding!2,downstream!1), RNA(binding,upstream!1,type~BBaC0040),
           Ribosome(rna!2) -> \
       RNA(binding,downstream!1), RNA(binding!2,upstream!1,type~BBaC0040),
           Ribosome(rna!2) @ 'translation initiation rate'
'cI translation initiation' \
       RNA(binding!2,downstream!1), RNA(binding,upstream!1,type~BBaC0051),
           Ribosome(rna!2) -> \
       RNA(binding,downstream!1), RNA(binding!2,upstream!1,type~BBaC0051),
           Ribosome(rna!2) @ 'translation initiation rate'
'REDLuc translation initiation' \
       RNA(binding!2,downstream!1), RNA(binding,upstream!1,type~BBaC0RED),
           Ribosome(rna!2) -> \
       RNA(binding,downstream!1), RNA(binding!2,upstream!1,type~BBaC0RED),
           Ribosome(rna!2) @ 'translation initiation rate'
'BLUELuc translation initiation' \
       RNA(binding!2,downstream!1), RNA(binding,upstream!1,type~BBaC0BLUE),
           Ribosome(rna!2) -> \
       RNA(binding,downstream!1), RNA(binding!2,upstream!1,type~BBaC0BLUE),
           Ribosome(rna!2) @ 'translation initiation rate'
'GREENLuc translation initiation' \
       RNA(binding!2,downstream!1), RNA(binding,upstream!1,type~BBaC0GREEN),
           Ribosome(rna!2) -> \
       RNA(binding,downstream!1), RNA(binding!2,upstream!1,type~BBaC0GREEN),
           Ribosome(rna!2) @ 'translation initiation rate'

%var: 'translation rate' 10

'LacI translation' \
       RNA(binding!1,type~BBaC0012), Ribosome(rna!1) -> \
```

```
       RNA(binding,type~BBaC0012), Ribosome(rna), LacI(dna,lactose) @ '
           translation rate'
'TetR translation' \
       RNA(binding!1,type~BBaC0040), Ribosome(rna!1) -> \
       RNA(binding,type~BBaC0040), Ribosome(rna), TetR(dna,atc) @ 'translation
           rate'
'cI translation' \
       RNA(binding!1,type~BBaC0051), Ribosome(rna!1) -> \
       RNA(binding,type~BBaC0051), Ribosome(rna), cI(dna) @ 'translation rate'
'REDLuc translation' \
       RNA(binding!1,type~BBaC0RED), Ribosome(rna!1) -> \
       RNA(binding,type~BBaC0RED), Ribosome(rna), REDLuc() @ 'translation rate'
'BLUELuc translation' \
       RNA(binding!1,type~BBaC0BLUE), Ribosome(rna!1) -> \
       RNA(binding,type~BBaC0BLUE), Ribosome(rna), BLUELuc() @ 'translation rate
           '
'GREENLuc translation' \
       RNA(binding!1,type~BBaC0GREEN), Ribosome(rna!1) -> \
       RNA(binding,type~BBaC0GREEN), Ribosome(rna), GREENLuc() @ 'translation
           rate'

%var: 'ribosome falloff rate' 0.01

'Ribosome falloff' Ribosome(rna!1), RNA(binding!1) -> Ribosome(rna), RNA(binding
    ) @ 'ribosome falloff rate'


# Creation and communication of light.

%var: 'light creation rate' 0.003

'Creation of RED light' REDLuc() -> REDLuc(), RED() @ 'light creation rate'
'Creation of BLUE light' BLUELuc() -> BLUELuc(), BLUE() @ 'light creation rate'
'Creation of GREEN light' GREENLuc() -> GREENLuc(), GREEN() @ 'light creation
    rate'

#'Communication of RED light' -> RED() @ 0.0
#'Communication of BLUE light' -> BLUE() @ 0.0
#'Communication of GREEN light' -> GREEN() @ 0.0


# Degradation of various agents.
```

```
'RNA degradation' RNA(binding,downstream) -> @ 0.0058

%var: 'transcription factor degradation rate' 0.00115


'LacI degradation' LacI(dna) -> @ 'transcription factor degradation rate'
'TetR degradation' TetR(dna) -> @ 'transcription factor degradation rate'
'cI degradation' cI(dna) -> @ 'transcription factor degradation rate'

%var: 'luciferase degradation rate' 0.00115

'REDLuc degradation' REDLuc() -> @ 'luciferase degradation rate'
'BLUELuc degradation' BLUELuc() -> @ 'luciferase degradation rate'
'GREENLuc degradation' GREENLuc() -> @ 'luciferase degradation rate'

%var: 'light dissipation rate' 0.05

'RED light dissipation' RED() -> @ 'light dissipation rate'
'BLUE light dissipation' BLUE() -> @ 'light dissipation rate'
'GREEN light dissipation' GREEN() -> @ 'light dissipation rate'


# Initial conditions, observables, and perturbations.

%var: 'operon count' 1
%var: 'RNA polymerase count' 700
%var: 'ribosome count' 18000
%var: 'transcription factor count' 50

%init: 'RNA polymerase count' (RNAP(dna,rna))
%init: 'ribosome count' (Ribosome(rna))
%init: 'transcription factor count' (Cph8(ompr,active~on))
%init: 'transcription factor count' (OmpR(dna,cph8,site~p))
%init: 'transcription factor count' (CcaS(phob,active~on))
%init: 'transcription factor count' (PhoB(dna,ccas,site~p))
%init: 'transcription factor count' (LOVTAP(dna,active~dark))
%init: 'operon count' (DNA(upstream,downstream!1,binding,type~BBaR0PHOp1), DNA(
    upstream!1,downstream!2,binding,type~BBaR0PHOp2), DNA(upstream!2,downstream
    !3,binding,type~BBaR0PHOp3), \
                    DNA(upstream!3,downstream!4,binding,type~BBaR0051p1), DNA(
                        upstream!4,downstream!5,binding,type~BBaR0051p2), DNA(
```

```
                    upstream!5,downstream!6,binding,type~BBaR0051p3), \
                DNA(upstream!6,downstream!7,binding,type~BBaR0051p4), DNA(
                    upstream!7,downstream!8,binding,type~BBaB0034), DNA(
                    upstream!8,downstream!9,binding,type~BBaC0012), \
                DNA(upstream!9,downstream,binding,type~BBaB0011))
%init: 'operon count' (DNA(upstream,downstream!1,binding,type~BBaR0082p1), DNA(
    upstream!1,downstream!2,binding,type~BBaR0082p2), DNA(upstream!2,downstream
    !3,binding,type~BBaR0082p3), \
                DNA(upstream!3,downstream!4,binding,type~BBaR0082p4), DNA(
                    upstream!4,downstream!5,binding,type~BBaR0082p5), DNA(
                    upstream!5,downstream!6,binding,type~BBaR0010p1), \
                DNA(upstream!6,downstream!7,binding,type~BBaR0010p2), DNA(
                    upstream!7,downstream!8,binding,type~BBaR0010p3), DNA(
                    upstream!8,downstream!9,binding,type~BBaR0010p4), \
                DNA(upstream!9,downstream!10,binding,type~BBaB0034), DNA(
                    upstream!10,downstream!11,binding,type~BBaC0040), DNA(
                    upstream!11,downstream,binding,type~BBaB0011))
%init: 'operon count' (DNA(upstream,downstream!1,binding,type~BBaK191007p1), DNA
    (upstream!1,downstream!2,binding,type~BBaK191007p2), DNA(upstream!2,
    downstream!3,binding,type~BBaK191007p3), \
                DNA(upstream!3,downstream!4,binding,type~BBaR0040p1), DNA(
                    upstream!4,downstream!5,binding,type~BBaR0040p2), DNA(
                    upstream!5,downstream!6,binding,type~BBaR0040p3), \
                DNA(upstream!6,downstream!7,binding,type~BBaR0040p4), DNA(
                    upstream!7,downstream!8,binding,type~BBaB0034), DNA(
                    upstream!8,downstream!9,binding,type~BBaC0051), \
                DNA(upstream!9,downstream,binding,type~BBaB0011))
%init: 'operon count' (DNA(upstream,downstream!1,binding,type~BBaR0010p1), DNA(
    upstream!1,downstream!2,binding,type~BBaR0010p2), DNA(upstream!2,downstream
    !3,binding,type~BBaR0010p3), \
                DNA(upstream!3,downstream!4,binding,type~BBaR0010p4), DNA(
                    upstream!4,downstream!5,binding,type~BBaB0034), DNA(
                    upstream!5,downstream!6,binding,type~BBaC0RED), \
                DNA(upstream!6,downstream,binding,type~BBaB0011))
%init: 'operon count' (DNA(upstream,downstream!1,binding,type~BBaR0040p1), DNA(
    upstream!1,downstream!2,binding,type~BBaR0040p2), DNA(upstream!2,downstream
    !3,binding,type~BBaR0040p3), \
                DNA(upstream!3,downstream!4,binding,type~BBaR0040p4), DNA(
                    upstream!4,downstream!5,binding,type~BBaB0034), DNA(
                    upstream!5,downstream!6,binding,type~BBaC0BLUE), \
                DNA(upstream!6,downstream,binding,type~BBaB0011))
%init: 'operon count' (DNA(upstream,downstream!1,binding,type~BBaR0051p1), DNA(
```

```
    upstream!1,downstream!2,binding,type~BBaR0051p2), DNA(upstream!2,downstream
    !3,binding,type~BBaR0051p3), \
                    DNA(upstream!3,downstream!4,binding,type~BBaR0051p4), DNA(
                        upstream!4,downstream!5,binding,type~BBaB0034), DNA(
                        upstream!5,downstream!6,binding,type~BBaC0GREEN), \
                    DNA(upstream!6,downstream,binding,type~BBaB0011))

%obs: 'Red' RED()
%obs: 'Green' GREEN()
%obs: 'Blue' BLUE()
%obs: 'TetR' TetR()
%obs: 'LacI' LacI()
%obs: 'cI' cI()
```