VERIFICAÇÃO PROBABILÍSTICA DE MODELOS PARA MODELAGEM E ANÁLISE DE INTERAÇÕES DE TOXINAS COM SISTEMAS DE TRANSPORTE TRANSMEMBRÂNICO DE ÍONS

FERNANDO AUGUSTO FERNANDES BRAZ

VERIFICAÇÃO PROBABILÍSTICA DE MODELOS PARA MODELAGEM E ANÁLISE DE INTERAÇÕES DE TOXINAS COM SISTEMAS DE TRANSPORTE TRANSMEMBRÂNICO DE ÍONS

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação do Instituto de Ciências Exatas da Universidade Federal de Minas Gerais como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação.

Orientador: Sérgio Vale Aguiar Campos
Coorientador: Alessandra Faria Aguiar Campos

Belo Horizonte

Março de 2013

FERNANDO AUGUSTO FERNANDES BRAZ

PROBABILISTIC MODEL CHECKING FOR MODELING AND ANALYSIS OF TOXINS INTERACTIONS WITH TRANSMEMBRANE IONIC TRANSPORT SYSTEMS

Dissertation presented to the Graduate Program in Computer Science of the Federal University of Minas Gerais. Computer Science Department. in partial fulfillment of the requirements for the degree of Master in Computer Science.

Advisor: Sérgio Vale Aguiar Campos
Co-Advisor: Alessandra Faria Aguiar Campos

Belo Horizonte

March 2013

Braz, Fernando Augusto Fernandes

B827p Probabilistic Model Checking for Modeling and Analysis of Toxins Interactions with Transmembrane Ionic Transport Systems / Fernando Augusto Fernandes Braz. — Belo Horizonte, 2013
xxxiii, 104 f. : il. ; 29cm

Dissertação (mestrado) — Universidade Federal de Minas Gerais. Departamento de Ciência da Computação.
Orientador: Sérgio Vale Aguiar Campos

1. Computação - Teses. 2. Modelagem de informações - Teses. 3. Sistemas de recuperação da informação - Biologia. - Teses. I. Orientador. II. Título.

CDU 519.6*74 (043)

UNIVERSIDADE FEDERAL DE MINAS GERAIS
INSTITUTO DE CIÊNCIAS EXATAS
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

FOLHA DE APROVAÇÃO

VERIFICAÇÃO PROBABILÍSTICA DE MODELOS PARA MODELAGEM E ANÁLISE DE INTERAÇÕES DE TOXINAS COM SISTEMAS DE TRANSPORTE TRANSMEMBRÂNICO DE ÍONS (PROBABILISTIC MODEL CHECKING FOR MODELING AND ANALYSIS OF THE INTERACTION OF DRUGS AND TOXINS WITH TRANSMEMBRANE IONIC TRANSPORT SYSTEMS)

FERNANDO AUGUSTO FERNANDES BRAZ

Dissertação defendida e aprovada pela banca examinadora constituída pelos Senhores:

Prof. Sérgio Vale Aguiar Campos - Orientador
Departamento de Ciência da Computação - UFMG

Dra. Alessandra Conceição Faria Aguiar Campos - Coorientador
Bolsista Pós-Doc/ ICB - UFMG

Prof. Antônio Carlos Guimarães de Almeida
Departamento de Engenharia Biomédica - UFSJ

Prof. Mark Alan Junho Song
Departamento de Ciência da Computação - PUCMG

Belo Horizonte, 15 de março de 2013.

I dedicate this work to all who supported me.

Agradecimentos

Agradeço à minha querida mãe, pelo eterno apoio. Seu exemplo de luta, sacrifício e determinação me marcou, e será sempre meu norte.

Agradeço ao meu amor Juliana, pelo carinho e companhia. Caminhamos lado a lado nessa aventura chamada pós-graduação, e espero que seja a primeira de muitas.

Agradeço ao Júnior, irmão que sempre me incentivou. Sem ele eu não teria o mesmo interesse pelos computadores e provavelmente seria um engenheiro civil.

Agradeço ao meu orientador Sérgio Campos, pela paciência comigo e meus terríveis textos, pelo exemplo de pessoa, e pelos conselhos e casos que ilustravam o que eu precisava entender para avançar na pesquisa (por mais que eu não entendesse).

Agradeço à minha co-orientadora Alessandra Faria-Campos, pelo incentivo e críticas construtivas e pela ajuda na revisão dos artigos e pôsteres.

Agradeço ao professor Jader Cruz, pela disponibilidade para reuniões sobre nossos resultados e pela ajuda para responder os revisores.

Agradeço ao professor Mark Alan Junho Song, que me apresentou à pesquisa e aos métodos formais durante minha iniciação científica, e por aceitar o convite para integrar a banca da defesa desta dissertação.

Agradeço ao professor Antônio Carlos Almeida, pela participação na banca da defesa desta dissertação, pelas críticas construtivas que enriqueceram este trabalho.

Agradeço ao professor Antônio Márcio Rodrigues, que juntamente do professor Antônio Carlos Almeida, nos recebeu prontamente em São João Del Rei, dispostos a esclarecer quaisquer dúvidas sobre seus trabalhos, que embasaram esta dissertação. Espero podermos colaborar em um futuro próximo.

Agradeço à Mirlaine, pelo seu trabalho anterior e poucas porém importantes conversas, que tornaram possível a realização desse trabalho.

Agradeço aos colegas do laboratório LUAR – Cristiano, Herbert, Lucas Amaral, Lucas Hanke, Moema, Nikolas, Vinícius, César e João. Um agradecimento especial ao Paulo e Marcelo, que sempre me ajudaram nos servidores. Outro agradecimento especial ao Bruno, companheiro de pesquisa.

Agradeço aos professores do DCC, pela qualidade das disciplinas ministradas.

Agradeço aos funcionários do DCC, particularmente a Sheila, Renata e Linda, que sempre me ajudaram prontamente quando precisava de alguma coisa.

Finalmente, agradeço ao DCC e a UFMG propriamente ditos, pela oportunidade de cursar uma pós-graduação de extrema excelência, pois enquanto estudar for um previlégio para poucos, é sinal de que ainda temos muito que evoluir.

"The first and best victory is to conquer self."
(Plato, Greek Philosopher)

Abstract

Probabilistic Model Checking (PMC) is a formal verification technique used for the specification and analysis of stochastic systems. It allows the exhaustive and automatic exploration of a system state space, checking whether it respects a set of properties. It can provide valuable insight, complementary to other approaches, such as simulations.

PMC can be applied directly to biological systems which present stochastic behavior, including transmembrane ionic transport systems. These systems are responsible for transporting ions across the cell membrane, for example, the sodium-potassium-pump, which participates in several biological processes, such as heart muscle contraction. They can be affected by different diseases, syndromes and toxins.

This dissertation proposes the use of PMC to model and analyze the interactions of a toxin called palytoxin (PTX) with the sodium-potassium-pump. This toxin completely disrupts the behavior of the pump.

We have built four models, each one focused on a different aspect of the system. Using a parametric study we have investigated different scenarios, such as diseases which decrease the concentration of cell energy (ATP) or fatal exposures to PTX.

These models suggested different behaviors, such as that high concentrations of ATP and potassium inhibit PTX action, while sodium enhances it. Individuals with ATP depletion, such as in brain disorders, may be more susceptible to the toxin, and the known inhibitory effect of potassium on PTX action has been observed.

On the other hand, electrolyte disturbances (for example, diabetes insipidus) could make an individual more susceptible to the toxin. Since PTX is found in an environment with a high concentration of sodium, this does not seem to be a coincidence.

We have also enhanced the kinetic model, which is used for describing the system reactions, with probabilities, creating a heat map. The map reveals unexpected situations, such as a frequent reaction between unlikely pump states, which suggests that either these states are temporary; or there is an unknown state between those two.

Since electrolyte levels in the blood can be manipulated up to a certain degree, while the ATP concentration simply can not be stimulated directly, the study of their role and capability to change the behavior of our models is even more important. This type of analysis can provide a better understanding of how cell transport systems behave, being complementary to other approaches such as simulations, and can lead to the discovery and development of drugs.

Palavras-chave: *Probabilistic Model Checking*, *Systems Biology*, *Ion Channels and Ionic Pumps*, *Blockers and Openers.*

Resumo Estendido

Verificação de modelos probabilísticos (do inglês Probabilistic Model Checking, ou PMC) é uma técnica que tem sido amplamente utilizada para especificação e análise de sistemas que apresentam características não-determinística, estocásticas e dinâmicas. Essa técnica pode ser diretamente aplicada a sistemas biológicos que apresentam essas características, onde eventos tais como reações químicas ocorrem simultaneamente, disparando outros eventos, e geralmente de forma aleatória. PMC consiste em verificar de forma exaustiva e automática se um sistema modelado formalmente respeita um conjunto de propriedades lógicas descritas em lógicas probabilísticas. Nessa dissertação, é proposto o uso de PMC para modelar e analisar a influência de toxinas tais como a palitoxina em sistemas de transporte transmembrânico de íons, estruturas celulares responsáveis por transportar íons através da membrana plasmática, cujo funcionamento perfeito é necessário para a saúde de um indivíduo, caso contrário o mesmo pode ser colocado em risco, e doenças podem se apresentar.

Introdução

Nesse capítulo é apresentada a motivação dessa dissertação, cujo objetivo é entender melhor o comportamento de sistemas biológicos tais como a bomba de sódio e potássio e sua interações com toxinas como a palitoxina através de modelos formais. Esses sistemas biológicos são estudados utilizando experimentos laboratoriais. No entanto, o sucesso desses experimentos é comprometido por uma série de fatores, desde falha humana na execução dos mesmos, até falha dos equipamentos ou substâncias mal preparadas. Portanto, o uso de abordagens alternativas tais como modelos matemáticos e métodos computacionais poderia reduzir esses custos e auxiliar o biológo em sua pesquisa. Dessa forma, apresentamos a área de biologia sistêmica, que consiste na criação de modelos e técnicas com o propósito de entender melhor sistemas biológicos. Ao final do capítulo também é apresentada a organização da dissertação e as principais contribuições acadêmicas que surgiram da mesma.

Modelagem Matemática

Nesse capítulo apresentamos a modelagem matemática de sistemas biológicos, que pode ser classificada de acordo com o comportamento do modelo (determinístico ou estocástico) e o tipo de seus valores ou dados (discreto ou contínuo). Essa modelagem matemática é apresentada para servir de base no entendimento dos modelos computacionais, e em quais situações eles são adequados. Processos estocásticos também são brevemente discutidos, tais como cadeias de Markov, onde é explicada sua necessidade dependendo das condições do sistema, por exemplo, uma baixa concentração de ligantes. Em seguida, são apresentados formalimos pertinentes aos próprios sistemas biológicos e que devem ser incorporados na modelagem, tais como a química discreta para obter o número de elementos a partir de sua concentração, e a lei da ação de massas, para reações que envolvem mais de um ligante individual.

Modelos Computacionais

Nesse capítulo apresentamos alguns exemplos de modelos computacionais. De forma similar a modelagem matemática, modelos computacionais podem ser classificados de diversas formas, por exemplo, o comportamento do modelo, que pode ser previsível ou aleatório, ou seus dados, que podem ser discretos ou contínuos. Essas duas classificações combinam-se em quatro diferentes tipos de abordagens para modelagem computacional, cada um apropriado para uma situação ou sistema sendo modelado. Alguns exemplos de modelos computacional também são discutidos, por exemplo, um sistema que apresenta comportamento estocástico e dados discretos pode ser eficientemente modelado como um modelo booleano probabilístico.

Verificação de Modelos

Nesse capítulo apresentamos a verificação de modelos, uma técnica utilizada para modelar e analisar sistemas, verificando se os mesmos respeitam um conjunto de propriedades. A verificação simbólica de modelos é apresentada, assim como seus conceitos e assuntos relacionados, tais como a representação de modelos através da estrutura de Kripke, e a estrutura de dados diagramas de decisão binária. As propriedades a serem verificadas são especificadas utilizando tipos especiais de lógica, as lógicas temporais, que também são apresentadas. Utilizando essas lógicas, é possível especificar diferentes tipos de propriedades, tais como “se uma mensagem é enviada, essa mensagem eventualmente é recebida”, ou “um sistema distribuído nunca entra em deadlock”.

A verificação de modelos probabilísticos também é apresentada, uma extensão da técnica anterior para permitir a modelagem e análise de sistemas estocásticos. Diferentes estruturas são necessárias, como os diagramas de decisão binária com múltiplos terminais, e as lógicas probabilísticas. Alguns dos verificadores de modelos, as ferramentas que implementam essas técnicas, são examinados, como é o caso do PRISM, utilizado na construção e verificação dos nossos modelos.

Sistemas de Transporte Transmembrânico de Íons

Nesse capítulo descrevemos sistemas de transporte de íons transmembrânico (canais e bombas iônicas), estruturas celulares que são responsáveis por transportar íons entre os meios externo e interno da célula. Canais iônicos transportam os íons de forma passiva, ou seja, em favor de seus gradientes de concentração (da mais alta para a mais baixa concentração). Bombas iônicas, por outro lado, transportam os íons de forma ativa, contra seus gradientes de concentração. Também apresentamos doenças, síndromes e toxinas que bloqueiam ou abrem esses sistemas, modificando seu comportamento normal e comprometendo a saúde da célula. Por exemplo, a toxina palitoxina se liga a bomba de sódio e potássio, abrindo a mesma, o que permite o movimento de íons de forma descontrolada.

Trabalhos Relacionados

Nesse capítulo apresentamos os trabalhos relacionados a essa dissertação, no que diz respeito a estudos experimentais e computacionais de sistemas de transporte transmembrânico de íons [Artigas and Gadsby, 2002]. Abordagens computacionais podem ser alternativas eficientes e econômicas para experimentos laboratoriais, por exemplo, utilizando simulações com equações diferenciais ordinárias para representar o comportamento de uma bomba acoplada com uma toxina [Rodrigues et al., 2008b]. Também apresentamos abordagens formais para estudar sistemas biológicos em geral [Kwiatkowska et al., 2008].

Análise Formal das Interações da Bomba de Sódio e Potássio com a Palitoxina

Nesse capítulo apresentamos a modelagem e análise formal de modelos eletrofisiológicos. Nosso estudo de caso é a bomba de sódio e potássio e suas interações com a toxina palitoxina, que abre a bomba e compromete diversos processos biológicos. Nós construímos quatro modelos diferentes, seguindo os resultados experimentais de Artigas e Gadsby, e os resultados com simulações de Rodrigues e colaboradores. Cada modelo explora um aspecto diferente do nosso estudo de caso, tal como as reações relacionadas a energia celular e o efeito inibitório do potássio sobre a ação da palitoxina. Nossos modelos foram construídos utilizando o verificador de modelos probabilísticos PRISM.

Discussão e Resultados

Nesse capítulo apresentamos os resultados obtidos nessa dissertação. Nós construímos quatro modelos formais das interações entre a toxina palitoxina (PTX) e a bomba de sódio e potássio utilizando uma abordagem de verificação de modelos probabilísticos através do verificador de modelos PRISM. Essa toxina perturba o comportamento da bomba, comprometendo diversos processos biológicos. Cada modelo foca em um aspecto do sistema. Por exemplo, nosso primeiro modelo foca no papel da energia celular, enquanto que o segundo modelo cobre as reações relacionadas ao sódio e potássio.

Depois disso nós realizamos um estudo paramétrico das dimensões dos nossos modelos, que permitem a investigação de condições extremas, tais como doenças que aumentam as concentrações de ligantes ou exposições fatais a PTX. Nós utilizamos propriedades quantitativas para quantificar cada aspecto do modelo, tais como as probabilidades de estados e reações

Cada modelo forneceu sugestões para o comportamento do sistema, tais como altas concentrações de ATP inibem a ação da PTX, o sódio aumenta a ação da PTX, e o potássio inibe a ação da PTX. Nós também quantificamos a corrente induzida pela troca de íons através da equação de fluxo Goldman-Hodgkin-Katz (GHK). O modelo completo reforçou nossos resultados anteriores.

Contribuições Adicionais

Nesse capítulo apresentamos as contribuições adicionais dessa dissertação, tais como a ferramenta *dot2heatmap*, que ajuda a produzir mapas de calor de grafos descritos na linguagem DOT utilizandos resultados obtidos através de PMC. Também apresentamos a ferramenta *MCHelper*, uma implementação inicial de um ambiente de apoio a verificação de modelos, que ajuda a gerenciar e tratar as informações e resultados produzidos através de PMC. Finalmente, também mostramos a ferramenta *PrismRecipes*, que ajuda a escrever modelos e propriedades do PRISM, que frequentemente apresentam certas regularidades e repetições.

Conclusões

Nesse capítulo apresentamos nossas conclusões a respeito da dissertação, e também indicamos possíveis trabalhos futuros nessa linha de pesquisa.

Nós modelamos e analisamos as interações da toxina Palitoxina (PTX) com a bomba de sódio e potássio. Essa toxina compromete o funcionamento correto da bomba, que participa de diversos processos biológicos importantes, tais como controle do volume celular e contração do músculo cardíaco. Nós construímos quatro modelos, seguindo os trabalhos de Artigas e Gadsby, e Rodrigues e colaboradores. Cada modelo se concentra em um aspecto do modelo completo, por exemplo, um modelo analisa o papel da energia celular (ATP), enquanto outro modelo foca nos íons sódio e potássio.

Nossos modelos sugerem que altas concentrações de ATP ou potássio podem inibir a ação da PTX, enquanto que altas concentrações de sódio aumentam a ação da PTX. Também fomos capazes de mensurar a corrente induzida pelos íons utilizando a equação de Goldman-Hodgkin-Katiz (GHK), assim como construir mapas de calor ao associar probabilidades aos estados e reações do modelo cinético. Esses mapas de calor revelaram situações inusitadas, tais como reações prováveis entre estados improváveis, o que sugere que os estados são temporários ou existe um estado desconhecido.

Indicamos como trabalhos futuros: a validação experimental dos nossos resultados; a expansão do modelo para novos estados e reações; a exploração sistemática de outros cenários; a adaptação do modelo para outras toxinas ou mesmo fármacos; uma análise aproximada dos modelos utilizando outras técnicas de verificação formal, o que ajudaria em análises macroscópicas; uma abordagem hierárquica que integrasse outras abordagens, tais como simulações matemáticas; uma abordagem mista ou híbrida que alternasse entre diferentes tipos de comportamento, por exemplo, entre determinístico

e estocástico; e, finalmente, modelar outros sistemas biológicos tais como vias de sinalização celular.

List of Figures

List of Tables

Contents

Chapter 1

Introduction

1.1 Challenges of Laboratory Experimentation

Biological systems and processes such as cell division, wine fermentation or DNA replication have been extensively investigated in order to discover their basic elements, as well as the interaction between those, which are responsible for characteristics and functions exhibited by life. It is desirable to discover the mechanisms created by nature, explain why some individuals show certain features, and find treatments or even cure for diseases in human beings such as cancer and AIDS.

Research of biological systems involves their investigation through laboratory experiments or tests, which depend on several factors for success. Experiments use different equipments – going from a simple microscope to next generation sequencing; techniques, compounds, such as enzymes, antibodies and solutions; and human resources. Each of these factors contributes (or even compromises) to achieve the expected results.

Experiments are expensive, demanding financial resources for the acquisiton of infrastructure and remuneration of personnel. Moreover, they demand time, and often they must be repeated several times in order to confirm a result. The execution of an experiment can be long depending on the technique. For example, the evolution of an inflammatory process in a biological model may require periodic sampling from 4 to 72 hours.

Besides the financial cost, trials may also present errors due to several reasons. Defects can occur in equipments due to the time of use or manufacturing error. Techniques can be poorly executed, and are prone to human error or inexperience of its practitioners. Substances used in the investigation may have problems, for example, a solution prepared incorrectly which is outside the desired concentration, or a perishable antibody which was not stored at the adequate temperature.

A considerable amount of trials must be performed to verify a hypothesis about a biological system and not draw conclusions based on false positives. One sample which confirms the hypothesis is not enough, since other samples might refute it. It is also desirable to perform a wide study and observe the behavior of the system under different conditions, which increases the number of experiments.

All these reasons – costs, time, possible errors, necessary redundancy and human resources – make the task of biologists even more challenging. Therefore, it is essential to find alternatives which can help biologists in their daily effort.

1.2 Systems Biology

Systems biology is an emergent and interdisciplinary research field which investigates, through computational and numerical methods, the interactions between biological models, such as kinetic models, metabolic networks and cell signalling pathways [Alon, 2006]. Some of these methods include deterministic and stochastic simulations, and model checking. It is necessary to study the interaction of these biological elements, because sometimes when they are analyzed separately, they do not indicate properties which occur during their interaction – it is the case of chemical reactions such as the hydrolysis of the water molecule in hydrogen and oxygen ions through the application of an electric current.

Methods from systems biology can help a biologist in the investigation of his biological models, exploring them in an exhaustive and systematic way, searching for test suggestions and experimentally neglected behaviors. Suppose that a biologist can perform several distinct experiments, however, he does not have resources to perform them all, and must prioritize one over another. A computational approach could obtain preliminary and discriminant results. The reduction in the number of experiments has as a consequence the reduction of costs, both financial and the time which would be spent performing all the experiments.

Moreover, the model could be adjusted to simulate possible scenarios difficult to obtain experimentally, such as an increased concentration of a ligand, or even its removal. The increased concentration is particularly useful to determine, for example, the toxicity of a drug, as well as its lethal dosage. The removal of a ligand allows the evaluation of the true mechanisms of interaction of present drugs, without interaction or interference of the removed drug.

Cost reduction is particularly important in research and development of drugs because the costs of that industry grow every year. Currently it costs on average US$4 billion dollars to research, develop and bring a new drug to the market [DiMasi and Grabowski, 2012]. In a previous study, this value was US$ 802 million dollars [DiMasi et al., 2003] – in about 9 years, the costs increased nearly 5 times. Table 1.1 shows the current costs of the development process of a new drug for several international pharmaceutical industries.

Company	Drugs Approved	Research Cost per Drug (US$Mil)	Total Research Cost 1997-2011 (US$Mil)
AstraZeneca	5	11.790,93	58.955
GlaxoSmithKline	10	8.170,81	81.708
Sanofi	8	7.909,26	63,274
Roche Holding AG	11	7.803,77	85.841
Pfizer Inc.	14	7.727,03	108.178
Johnson & Johnson	15	5.885,65	88.285
Eli Lilly & Co.	11	4.577,04	50.347
Abbott Laboratories	8	4.496,21	35.970
Merck & Co Inc	16	4.209,99	67.360
Bristol-Myers Squibb Co.	11	4.152,26	45.675
Novartis AG	21	3.983,13	83.646
Amgen Inc.	9	3.692,14	33.229

Table 1.1: Research costs to develop a new drug for different pharmaceutical industries. Source: InnoThink Center For Research In Biomedical Innovation; Thomson Reuters Fundamentals via FactSet Research Systems.

Besides the cost, a considerable period of time is also necessary – between 12 and 15 years for a drug to go from a research laboratory to the patient, and in average only five of 5000 drugs advance to clinical trials. Only one of these five drugs will be approved for usage. If computational approaches were more used, it would be possible to reduce the number of failed drugs which advance to the final stages of clinical trials. Figure 1.1 shows a summary of the process and the stages necessary for the drug approval [Nash, 2008].

It must be understood that is not that simple – the adoption of computational approaches also incurs in costs – computational infrastructure and formation and remuneration of interdisciplinary human resources. However, that is a reality which is closer each day. Partnerships between departments of distinct natures are increasingly common, as well as the creation of research centers and graduate courses on topics such as bioinformatics, computational biology and systems biology [Calvert et al., 2010].

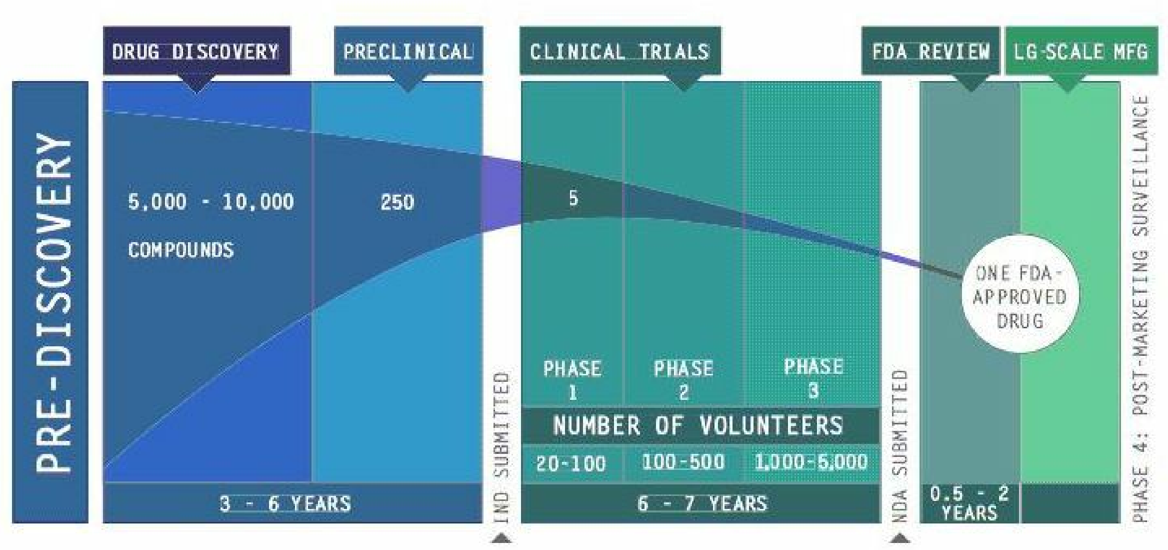


Figure 1.1: The stages of drug discovery process – going from pre-discovery which starts with 5 to 10 thousand compounds, passing through clinical trials, and finally reaching the approval of FDA (*Food and Drug Administration*), the United States drug regulatory agency [Nash, 2008].

1.3 Probabilistic Model Checking

Probabilistic model checking (PMC) is a formal verification technique to model and analyze stochastic systems, checking if they respect a set of given properties [Parker, 2002; Kwiatkowska et al., 2011]. It automatic and exhaustively explores a formal model, visiting the complete state space, which allows the detection of rare events [Clarke et al., 1999].

PMC has been successfully used in engineering applications, such as circuit designs and communication protocols. For example, one might check if two circuits are equivalent, a new wireless protocol is fair, a flight control system does crash or a message sent leads to its receival. The properties are specified using special types of logics, such as temporal, probabilistic and reward-based logics, which allow the reasoning over the sequence, probability and quantification of events, respectively.

This technique could be used in a systems biology study, by modeling biological systems which present probabilistic behavior. Stochastic biology systems are common, for example, in small and non-uniform environments, chemical reactions might or not occur, depending on a probability distribution.

The use of PMC in systems biology offers several advantages, being complementary to other computational approaches. For example, since the search is exhaustive, erratic behaviors can be observed. Also, deterministic and stochastic simulations, which are often used in systems biology, might not observe some rare event due to local minima, being limited by the number of iterations and other factors [Calvert et al., 2010]. In this dissertation we have used the PRISM model checking tool, which exhaustively checks if a stochastic model respects a set of probabilistic properties.

1.4 Palytoxin Interactions with the Sodium-Potassium-Pump

In the plasma membrane of all animal cells there are structures called transmembrane ionic transport systems, which are responsible for exchanging ions between the external and internal sides of the cell. One of these systems is the sodium-potassium-pump, which exchanges three internal sodium ions for two external potassium ions [Aidley and Stanfield, 1996].

The internal concentrations of sodium and potassium ions must be kept low and high, respectively, while the external concentrations are the opposite. However, the ions tend to move from a high concentration environment to a low one, therefore the pump moves ions against their concentration gradient. This ion movement costs cell energy, through the hydrolyzation of Adenosine Triphosphate (ATP). The pump plays a major role on several biological processes, such as cellular volume control and heart muscle contraction, therefore maintaining the appropriate concentration levels is essential for the health of an organism [Hille, 2001].

The pump can be affected by diseases, syndromes and toxins. One of these toxins is palytoxin (PTX), which opens the pump, disrupting its behavior. The control of ion movement is lost, compromising important biological processes. If untreated, it could lead to death [Artigas and Gadsby, 2002].

In this dissertation we have modeled and analyzed the interactions of PTX with the sodium-potassium-pump using a probabilistic model checking approach. We have built four models using the PRISM tool, each one focusing on a different aspect of the system. We have also used a parametric study to investigate different conditions, such as diseases which decrease the concentration of sodium or fatal exposures of PTX. Our models have suggested that ATP and potassium inhibit PTX action, while sodium enhances it. Finally, we have enhanced with probabilities the kinetic models, creating heat maps, which could be visual tools for biologists.

1.5 Major Contributions

Some of the work in this dissertation has been published previously in jointly authored papers. In [Braz et al., 2012b], an initial model for the palytoxin effects on the sodium-potassium pump was presented. This model includes the cell energy (ATP) interactions with the system, which suggests that ATP might inhibit PTX action, as described in Section 8.2. The model description was largely based on the works of [Artigas and Gadsby, 2002] and [Rodrigues et al., 2008b].

The role of sodium and potassium interactions with the palytoxin-sodium-potassium pump complex was evaluated in [Braz et al., 2012a]. The description was once again based on the works of [Artigas, 2003b] and [Rodrigues et al., 2008a]. This work indicated that sodium enhances the PTX action, which is particularly interesting since the marine species that produces the toxin is naturally found in a sodium-rich environment. On the other hand, it suggested that potassium inhibits PTX action, which could be used to reduce toxin action.

The publications that are not part of this dissertation include [Ferreira et al., 2012], where the authors have built a mobility model for vehicular networks using PMC.

1.6 Organization

This dissertation is structured in ten chapters, described below. At the end of each chapter there is a summary to briefly review its contents.

Chapter 2: Mathematical Modeling. The next chapter reviews the mathematical formalisms for the description of biological systems and processes. It classifies them in their behavior, which can be either deterministic or stochastic, and the nature of their values, which can be discrete or continuous. Stochastic processes are also briefly described since our models are Markov chains. Biological modeling techniques are also covered, such as the discrete chemistry to convert concentrations into number of ligands, the individual approach to model single systems and the law of mass action, which explains the behavior of solutions.

Chapter 3: Computational Models. This chapter reviews the current literature of computational methods for the description and analysis of biological systems. These methods use different combinations of the formalisms discussed in the previous chapter. For example, Petri nets, which present deterministic behavior and discrete states, and stochastic differential equations, which show stochastic behavior and continuous values.

Chapter 4: Model Checking. This chapter presents model checking, a formal, exhaustive and automatic verification approach which allows verifying if a model satisfies a set of logic properties. The symbolic model checking is first described, covering its symbolic representation and temporal logics. Since our models are stochastic, probabilistic model checking is covered, including its probabilistic and reward-based logics. Model checkers of Markov chains are reviewed. The PRISM model checking tool was used in this work, therefore we have included a brief description of its features.

Chapter 5: Transmembrane Ionic Transport Systems. This chapter describes ion channels and ionic pumps. These systems are responsible for exchanging ions across the cell membrane and play a major role in several biological processes, such as cell volume control and heart muscle contraction. Our case study is the sodium-potassium pump, which is responsible for exchanging three sodium ions from the internal side of the cell for two potassium ions from the external side, at the cost of the cell energy (phosphorylation of an Adenosine Triphosphate molecule). Transport systems can be blocked (locked in a particular state) or opened (losing ion movement control) by different compounds such as drugs and toxins. One of these toxins, the Palytoxin, is further described since it is exposed to the pump in our model in order to study its effects on the health of the individual.

Chapter 6: Related Work. This chapter covers the related work of this dissertation. Since it has an interdisciplinary nature – computer science and biology – the references have been divided in two groups. The first group named Analysis of Transmembrane Ionic Transport Systems consists of experimental and simulational techniques. These works were studied in order to better understand ion channels and ionic pumps under different views. The second group entitled Modeling and Formal Analysis of Biological Systems consists of works in the model checking area which analyzed biological systems or stochastic systems in general. Once again, these works were examined to understand the modeling methodologies and property specification of biological systems using model checking.

Chapter 7: Formal Analysis of Transmembrane Ionic Transport Systems. This chapter presents the computational approach to model and analyze our case study, the sodium-potassium-pump interactions with the palytoxin.

The approach is the formal verification through probabilistic model checking using the PRISM model checker. The components of each of our models are described, including their respective PRISM model codes. First we cover the sodium-potassium-pump part, then the palytoxin extension.

Chapter 8: Discussion and Results. This chapter presents the results obtained through our four models – each of the first three models covers one aspect of our case study, while the complete model is presented lastly. The first model describes the ATP interactions with the PTX-pump complex. This model does not have sodium or potassium. The second model covers the sodium and potassium interactions with the PTX-pump complex, while the third model describes only the potassium interactions. These two models do not have ATP and the difference between them is that the first one analyzes the dynamics of both ions, while the second one focuses on the known inhibitory effect of potassium on palytoxin. Finally, the complete model includes every aspect of the case study. Our models suggested that ATP and potassium inhibits PTX action, while sodium enhances it. An increased potassium concentration also changes the ATP binding site, which could be an explanation for its inhibitory effect. Finally, we have enhanced the kinetic models with probabilities, creating heat maps, which could be visual tools for biologists.

Chapter 9: Additional Contributions. This chapter presents other works which have originated from this dissertation. One of these is the tool called `dot2heatmap` to automate the creation of heat maps from kinetic models. Another contribution is the `MCHelper`, a support environment for model checking, which helps parsing results and logs. This tool was jointly developed by the author and João Sales Amaral. Finally, the tool `PrismRecipes` is briefly described, which helps writing PRISM models and properties that often show regularities and repetitions.

Chapter 10: Conclusions This chapter summarizes the main results of this dissertation, and discusses possible future works.

Chapter 2

Mathematical Modeling

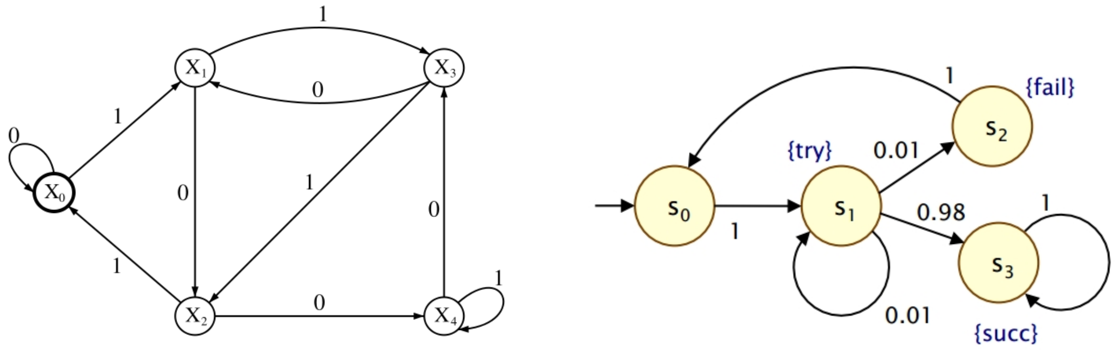
Outline. In this chapter we present a brief background on the mathematical modeling of biological systems, which can be classified regarding their behavior (deterministic or stochastic) and the type of its values (discrete or continuous). Stochastic processes are also covered, explaining their necessity based on the system's conditions, usually low concentrations of involved components. Biological systems have their own formalisms that must be incorporated into the modeling, such as discrete chemistry and the law of mass action.

2.1 Classification

The mathematical formalism for the description of biological systems and processes can be characterized under two aspects: the model **behavior**, which can be deterministic or stochastic (Figure 2.1) and the type of model **values**, such as time and measurements, which can be discrete or continuous (Figure 2.2).

The model behavior can be **deterministic**, in other words, there are no random variables which participate in the evolution of the modeled system. Given an initial condition or input, the model always produces the same result. The next state is defined by the current state. Deterministic behaviors can be described, for example, using ordinary differential equations, which can be solved through numerical analysis approaches, such as the Simplex algorithm and the Newton-Raphson method. Some examples of deterministic systems include: the game of billiards, a car engine, a factory production line and computers in general [Filho, 2007].

On the other hand, the model behavior can be **stochastic**, i.e., there are random elements in the system which actively participate in its evolution process. The next state is no longer defined only by the current state, and it is now defined by a probability



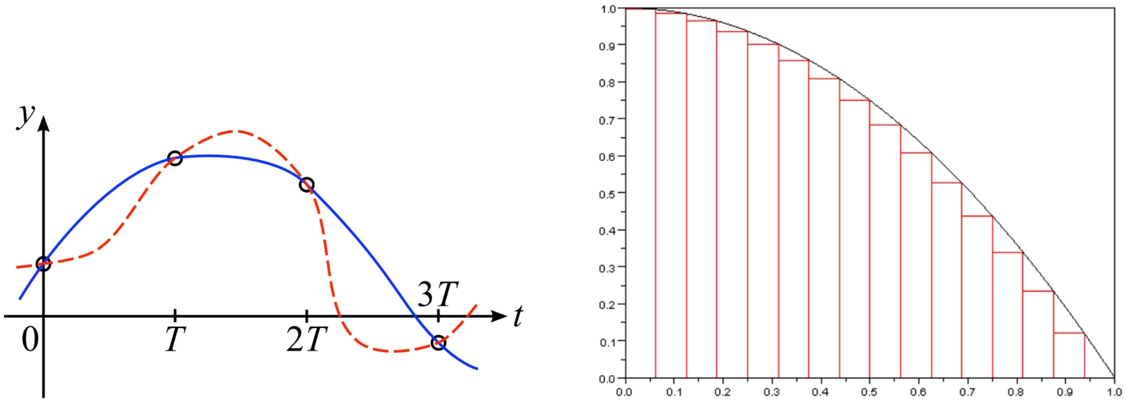
(a) Deterministic behavior. Given a state and an input character, the deterministic automaton transits from that state to only another state.

(b) Stochastic behavior. The transition from one state to another is defined by a probability distribution.

Figure 2.1: Different types of behaviors for mathematical modeling.

distribution to the next states. The outcome of the model is an estimate of the true characteristics of the system. Examples of stochastic systems: a coin toss, the body movement and the human brain.

Values for time, state, space and measurements in general can be **discrete**, in other words, belong to a discrete set. For example, the passage of time can be performed periodically using a fixed time interval between measurements. It is assumed that the current state of the system does not change during the transition to the next state. However, this assumption is not appropriate to all models, since it might not observe a local minima between subsequent measurements, as demonstrated in Figure 2.2a – the system (dashed trace) does not behave as the model or interpolation (solid trace).



(a) Discrete and periodic measurements of time T obtain the blue interpolated model. However, the red trace is the true behavior of the system.

(b) Approximation methods perform several measurements to obtain the continuous behavior of the model with the minimum error.

Figure 2.2: Different types of time measurements – discrete or continuous.

Finally, these values can be **continuous**. The passage of time, for example, although in theory it is measured continuously, in practice it is performed in small discrete time intervals. This is due to the limitations of the approximation method and the computer itself, which is inherently discrete (floating point representation incurs in rounding errors). Certain models must be described in a continuous way, otherwise the results would be incorrect, such as a set of chemical reactions which have distinct rates (different time intervals for measurements) and depend on the ligand concentrations (which are continuous). However, in order to reduce the approximation error, several measurements are necessary, which has an elevated computational cost.

2.2 Stochastic Processes

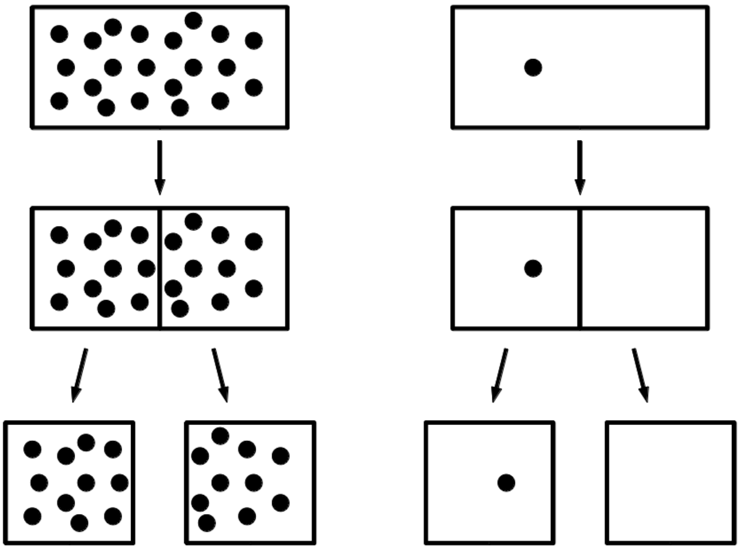
Biological systems sometimes can not be represented deterministically, for example, stochastic models described by ordinary differential equations. That is because of certain assumptions inherent to these forms of modeling. One of these assumptions is *the deterministic behavior itself*, which does not consider random fluctuations that usually do not belong to deterministic systems. In small systems these fluctuations occur, for example, chemical reactions are random events which may or may not happen.

Another assumption of deterministic methods is that *the model variables are continuous*, for example, a ligand of the model is expressed by its concentration (moles per liter or $\frac{mol}{L}$). That is a simplification, since the biological elements being represented – molecules, ions, proteins, etc. – are countable, i.e., have a discrete nature.

This assumption is reasonable as long as the number of elements is sufficiently large. However, if the elements are in the dozens or hundreds, then the discretization should be considered, as demonstrated by Figure 2.3. In that case, the system conditions might not be uniform and certain regions of the environment could have a higher number of elements than others regions.

Systems which present stochastic (or random behavior) could be described by a **stochastic process**, which is a set of random variables used to represent the evolution of the system over time. There are different types of stochastic processes, each one with its particularities and appropriate to certain purposes. Among them, there are continuous-time Markov chains.

The Markov chains are particularly interesting because they present the **Markov property**, which means that the next state is defined only by the current state and it is independent of past states. This property makes the approximation of a Markov



(a) High concentration.

(b) Low concentration.

Figure 2.3: Discretization should be considered when the concentrations of involved elements become sufficiently low, because the system ceases to be uniform. Dividing a high concentration environment yields two compartments with similar conditions (a), while this is not necessarily true for low concentrations (b).

chain computationally tractable since it is not necessary to visit all the previous states, or the system history, in order to define the next state.

These systems can also be described by random functions which receive as input different parameters and produce as output random variables, following certain probability distributions, such as the Gaussian or Bernoulli distributions. However, it is challenging to identify the distribution and parameters which fit the model behavior.

Some examples of modeling which use stochastic processes include stock market (investors making random choices), fluctuations in exchange rates (dynamic and chaotic economic factors) and medical data such as electrocardiography (EKG), electroencephalography (EEG), blood pressure or temperature (noise and fluctuations).

2.3 Biological Systems

Biological systems have their own traditional formalisms that must be incorporated into the modeling. The **discrete chemistry** obtains the number of involved elements (molecules, ions, proteins, etc.) from their concentrations. The **individual approach** is used to represent individual biological structures and their interactions. The **law of mass action** is used to describe chemical reactions where their unique involved elements have more than one quantity (for example, two ions).

2.3.1 Discrete Chemistry

The components of our models are molecules (for example, Adenosine Triphosphate, or ATP), ions (sodium or potassium) and the sodium-potassium-pump itself (also known as the Na^+/K^+-ATPase). Each of these components can interact with each other through several chemical reactions, such as two potassium ions binding to the pump.

Molecules and ions are incredibly small, and they are counted using the mole unit. One mole is equivalent to the Avogadro constant (N_A), or approximately 6.022×10^{23} units per mol. For example, one mole of ATP is 6.022×10^{23} molecules of ATP.

However, these components are often quantified using concentrations, which is a continuous measure. This concentration is called molar concentration or molarity, representing the moles per liter or $\frac{mol}{L}$. This is also denoted as Molar or M.

For example, the concentration of ATP is 5 mM (miliMolar) under normal physiological conditions. Instead of concentrations, our models have to use discrete variables to represent each of the molecules, for example, three sodium ions outside the cell and two potassium ions inside the cell. Therefore, we must convert the initial concentration of molecules and ions from molarity (M) to their corresponding number of molecules.

The concentration of ligands (molecules and ions) given in molarity ([X]) can be converted into quantities of ligands (#X) using the following biological definition:

$$\#X = [X] \times V \times N_{\mathrm{A}} \tag{2.1}$$

where V is the cell volume and N_{A} is the Avogadro constant (6.022×10^{23} mol^{-1}).

Each reaction of our model has an associated rate, which determines its frequency and likeliness to happen. These rates are also called stochastic rates, because there is some probability per unit time that the reaction will happen. For example, a phosphate molecule binding to the pump happens 1.90 times per second (or 1.90 s^{-1}), while the pump closing its door to the outside and opening its door to the inside of the cell happens 100 times per second (1.00×10^2 s^{-1}).

These stochastic reaction rates have been obtained after several experimental procedures and they are from [Rodrigues et al., 2008b,a, 2009a,b]. The concentration of a ligand such as ATP is denoted in brackets, for example, $[ATP]^i$, while the ^{i} indicates that it is the intracellular concentration. Another example is $[Na^+]^o$ for the extracellular sodium concentration. The ligands concentrations (for example, $[ATP]^i = 0.005$ M,

$[P]^i = 0.00495$ M and $[ADP]^i = 0.00006$ M) are from [Chapman et al., 1983]. The *cell volume* is from [Hernández and Chifflet, 2000].

2.3.2 Individual Approach

The authors of [Calder et al., 2010] present a computational modeling approach using formal verification with the PRISM model checker. They view their model – signaling pathways – as a distributed system, associating concurrent computational processes with each of the proteins in the pathway. In other words, the proteins are processes and the reactions are transitions.

They treat these components as populations, in order to capture the behavior of a whole set of ligands. Processes interact, or communicate with each other synchronously, by participating in reactions which build up and break down proteins. A producer can participate in a reaction when there is enough species for a reaction, a consumer can participate when it is ready to be replenished. A reaction occurs only when all the producers and consumers are ready to participate.

They view the protein species as a process, rather than each molecule as a process. This corresponds to a population type model (rather than an individuals type model). In traditional population models, species are represented as molar concentrations. In their approach, concentrations can vary in granularity; the coarsest possible discretisation being two values (representing, for example, enough and not enough, or high and low). Time is the only continuous variable, all others are discrete.

In this work, we use their modeling approach, therefore, our ligands are viewed as processes, and transitions connect the system states. However, we use the individual approach instead of the populational approach. This means that each of the ligands is modeled individually, for example, using discrete variables, which store the current number of a particular type of ligand.

2.3.3 Law of Mass Action

The law of mass action is a mathematical model that explains and predicts behaviors of solutions in dynamic equilibrium, such as the concentration gradients of sodium and potassium ions, as well as their interactions with the sodium-potassium-pump. This law states that a reaction rate is proportional to the concentration of its reagents. Therefore, we must take into account the reagent concentrations when defining a reaction rate. Considering the discrete chemistry conversion previously discussed and an ATP molecule binding to the pump:

$$\mathrm{ATP}^{i} + E_1 \xrightarrow{rp'_1} \mathrm{ATP}_{high} \sim E1 \tag{2.2}$$

The final rate rp_1 is given as follows:

$$rp_1 = rp'_1 \times \#(E_1) \times \#(\mathrm{ATP}^{i}) \tag{2.3}$$

The reaction rate is also multiplied by the current concentration of the involved ligand to the power of the number of ligands. For example, a reaction which involves three sodium ions would have to be multiplied by $([\mathrm{Na}^{+}]^{i})^{3}$. We have used the construct `pow(x,y)` from PRISM to represent the law of mass action. For example, a reaction involving three sodium ions would have a transition rate multiplied by `pow(naIn,3)`.

Summary. In this chapter we have presented the mathematical modeling of biological systems, which can be classified regarding their behavior (deterministic or stochastic) and the type of its values (discrete or continuous). Stochastic processes have also been covered, such as Markov chains, explaining their necessity based on the system's conditions, for example, low concentration of involved components). Biological systems have their own formalisms that must be incorporated into the modeling, such as discrete chemistry and the law of mass action, which have also been presented.

Chapter 3

Computational Models

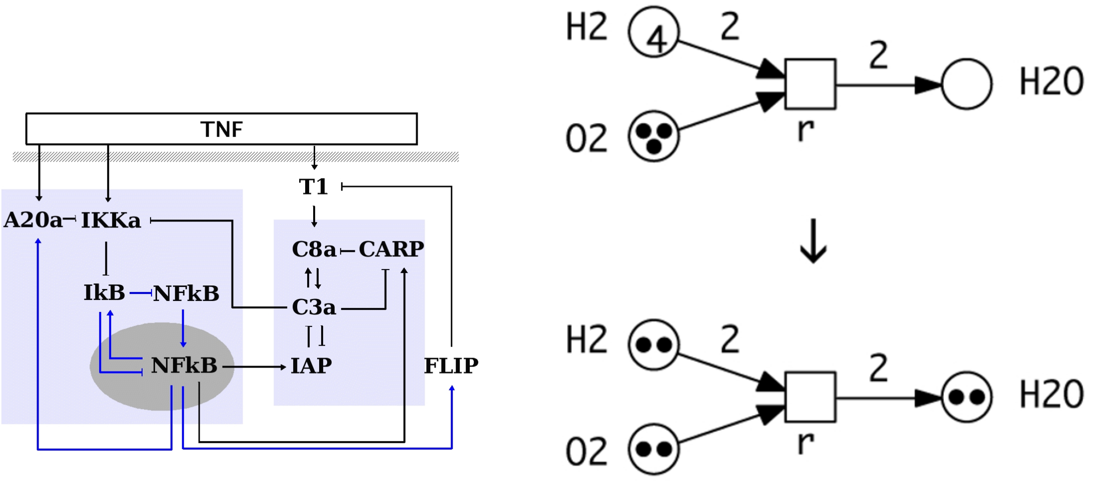
Outline. In this chapter we present some examples of computational models, which are useful tools to describe and reason over the behavior of a system. Similarly to mathematical models, computational models can be classified in various ways, for example, the behavior of the model, which can be predictable or ruled by chance, or the data, which can be discrete or continuous. These two classifications combine in four different types of computational modeling approaches, summarized in Table 3.1, each one suitable for a situation or system being modeled. Some examples of these computational models are discussed with greater detail below, for example, a system which presents stochastic behavior and discrete data can be effectively modeled as a probabilistic boolean model.

Behavior	Data	
	Discrete	Continuous
Deterministic	Boolean Models Petri Nets	Ordinary Differential Equations Partial Differential Equations
Stochastic	Master Equation Probabilistic Boolean Models	Approximations of Master Equation Dynamical Bayesian Models

Table 3.1: Different types of computational models, classified by its behavior (deterministic or stochastic) and data (discrete or continuous).

3.1 Deterministic and Discrete

Deterministic and discrete models are suited for systems which present well-defined and predictable outcomes and the number of elements is finite. Examples of these models are **boolean models** and **Petri nets**. There are several systems which present these characteristics, such as a cell signaling pathways and gene regulatory networks.



(a) Boolean model for a cell signaling pathway. Rules define the transitions between finite states. Circuits are built by linking components through their interfaces or connections. Adapted from [Chaves et al., 2009].

(b) Petri net representing a chemical reaction. The number of *tokens* represents the number of molecules available. It takes two molecules of H_2 and one of O_2 to produce two of H_2O. Adapted from [Murata, 1989].

Figure 3.1: Deterministic and discrete models.

Boolean models (Figure 3.1a) use predicate logic to describe the structure of the system being modeled (e.g. one state leads to another state) and check certain conditions (e.g. the model is never in two states). Variables assume values from a finite number of states, which can be two states, such as the true or false from boolean logic, or a set of states, for example, identifiers representing the stage that the cell is in a biological process such as cell division. Logic or predicate rules use boolean operators such as conjunction, disjunction and negation, and define the allowed transitions between states. Logic circuits are built to link each component of the model through their inputs and outputs. The Figure 3.1a presents a boolean model for the cell signaling pathway of the apoptosis – the cell death [Chaves et al., 2009].

A Petri net (Figure 3.1b) defines the structure of a distributed system as a directed graph. This graph consists of different types of nodes – position nodes, transition nodes and arc nodes connecting position nodes with transition ones. Each position node can store resources called *tokens*, which are processed by transition nodes. The Figure 3.1b presents a Petri net modeling a simple chemical reaction (adapted from [Heiner and Sriram, 2010]). The number of *tokens* represents the number of molecules available. It takes two molecules of H_2 and one of O_2 to produce two of H_2O [Murata, 1989].

3.2 Deterministic and Continuous

Systems which present deterministic outcomes and continuous data can be effectively modeled using deterministic and continuous models. Continuous values appear in, for example, concentrations of elements or rates of transitions. The representation of continuous values usually takes a single floating point variable, which although is cheaper than discrete approaches which represent elements individually, it comes with the price of representation and approximation errors.

Computers are discrete in nature, and extreme values such as an incredibly small number can be wrongly represented with a rounding error. Also, most of the continuous models are based on approximation methods, which means that eventually the method must stop in order to return its computation, for example, due to a limit of its iterations or a convergence value (i.e. new iterations are not changing the current result). This in turn could also lead to approximation errors.

Examples of these models are **ordinary differential equations** and **partial differential equations**. These models are appropriate to quantitatively analyze large systems where uniformity is observed for the presence of elements (i.e. elements are evenly distributed) and interactions between them.

Differential equations are ordinary if its functions are of only one variable (Equation 3.1). Otherwise, they are partial (Equation 3.2). Both types can be solved by techniques such as the Laplace and Fourier transforms, and the methods of finite elements, Euler and Runge-Kutta [Boyce, 2008].

$$\frac{d}{dx}x^2 = 2x \tag{3.1}$$

$$\frac{\partial}{\partial x}x^2 y = 2xy \tag{3.2}$$

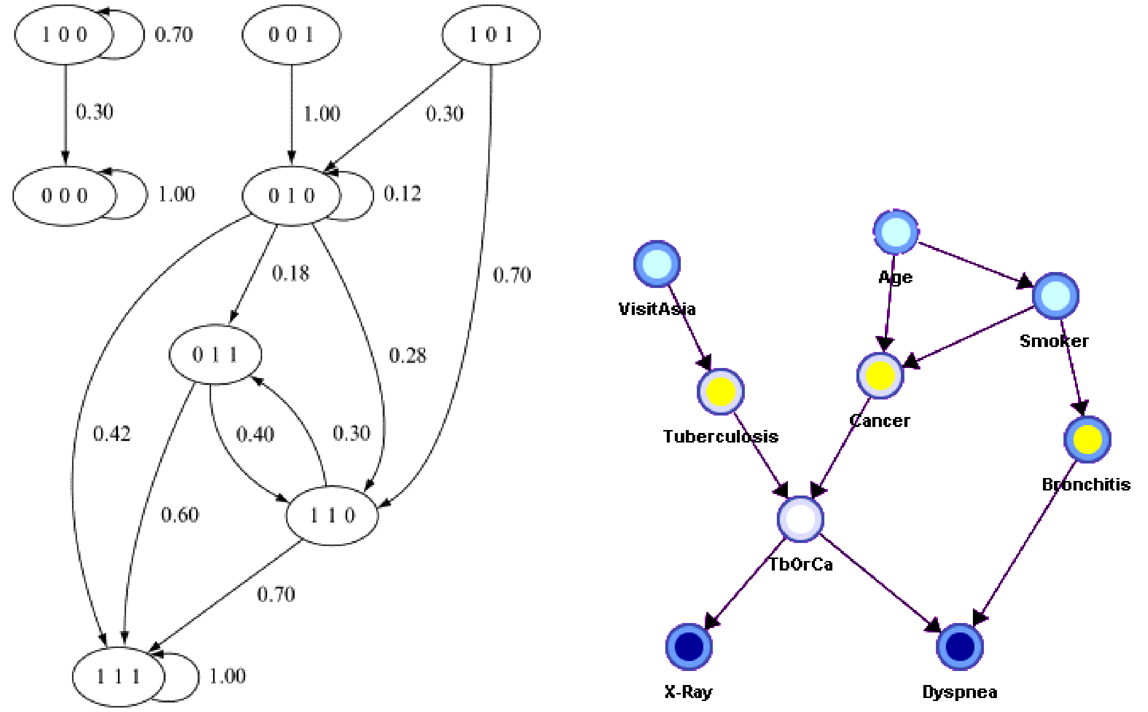
3.3 Stochastic and Discrete

Stochastic and discrete models are appropriate for small models which are random in nature and do not present uniformity in the presence of elements or their interactions. Examples of these models are the **Master Equation**, **probabilistic boolean models** and **discrete Bayesian models**.

Master equations are used to describe the system evolution over time. The system is in exactly one state of a finite set of states at a given time. The transitions between states are probabilistic. The equations are a set of first order differential equations for the fluctuation or rate over time of the probability for the system to be in each one of the different states.

$$\dot{p}(\mathbf{x};t) = -p(\mathbf{x};t)\sum_{\mu=1}^{M} a_\mu(x) + \sum_{\mu=1}^{M} p(\mathbf{x}-s_\mu;t)a_u(\mathbf{x}-s_\mu) \tag{3.3}$$

The Equation 3.3 presents the master chemical equation to estimate the probability of the system being in state $\mathbf{x}$ at time t. There are N distinct species identified by $\{S_1,\ldots,S_N\}$. The state space is described by an integer vector $\mathbf{x} = (x_1,\ldots,x_n)^T$, where x_i is the population of species S_i. There are M reactions $R_\mu : \mu \in \{1,2,\ldots,M\}$ which cause transitions defined by the stoichiometry matrix $S = [s_1\ s_2\ \cdots\ s_M]$. The probability that R_μ is the next reaction which will happen in the next dt time units is given by $a_\mu(\mathbf{x})dt$.



(a) Probabilistic boolean model. Discrete states are linked by probabilistic transitions. Adapted from [Laubenbacher and Jarrah, 2009].

(b) Bayesian networks. This graphical model exploits the causal relationship between variables and their independence. Adapted from [Bayesia, 2013].

Figure 3.2: Deterministic and discrete models.

Probabilistic boolean models are similar to boolean models, previously discussed in the Deterministic and Discrete section. The states still are discrete boolean vectors for the values of each variable. The difference is that there is a probability distribution associated with every rule (Figure 3.2a), instead of a deterministic transition [Laubenbacher and Jarrah, 2009].

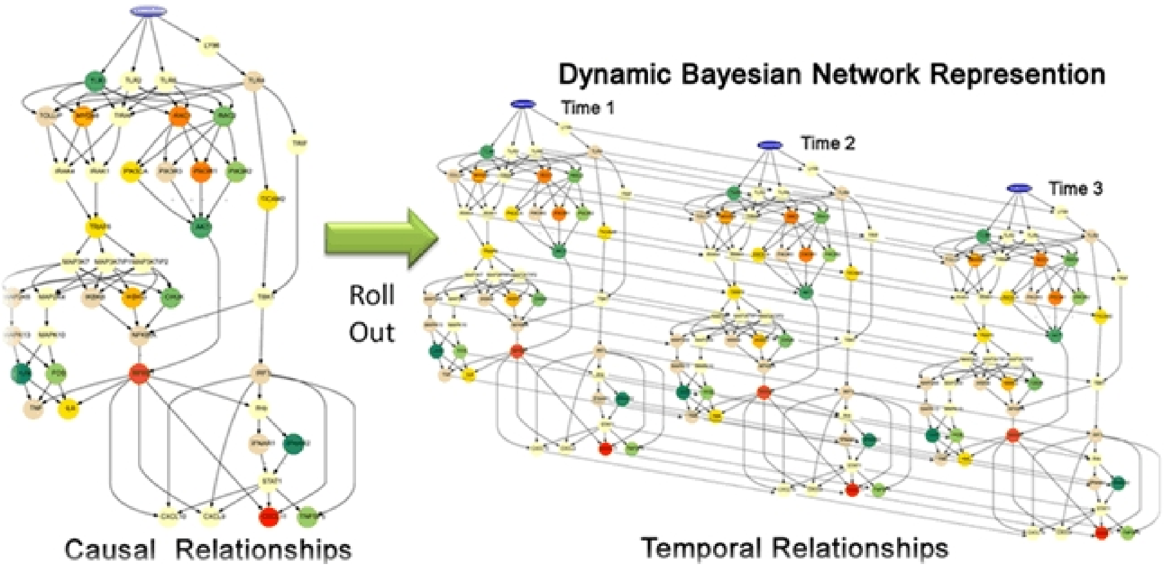


Figure 3.3: Dynamic Bayesian networks. This is an extension to bayesian networks, which allows reasoning over time. Adapted from [Next Generation Pharmaceutical, 2013].

Another example of discrete and stochastic model is the Bayesian network. This type of model is a graphical one (e.g. graph-based) which exploit the causal relationships between variables and their independence (Figure 3.2b). The advantage is that instead of explicitly representing the full joint probability distribution table (every combination of events), it is possible to represent only smaller ones, based on the dependence of variables. In order to reason over time using Bayesian networks, one has to extend to dynamic ones (Figure 3.3) [Bayesia, 2013].

3.4 Stochastic and Continuous

Finally, stochastic and continuous models are suited for large and random systems, where there is uniformity. Examples of these models are the **approximations of the Master Equation** and **dynamic Bayesian models**.

The Fokker-Planck equation is an approximations of the Master Equation which describes the time evolution of a continuous probability distribution. It is often used in the study of particle positioning and diffusion [Risken, 1984]. Another approach is the use of Stochastic Differential Equations (SDE), which are differential equaitons in which one or more of the terms is a stochastic process.

Dynamic Bayesian models, previously described, can also be used as continuous and stochastic models. In order to do so, one has to discretize the variables, or use probability density functions, such as the Gaussian distribution. Another approach is the use of soft thresholds, using probit and logit distributions [Next Generation Pharmaceutical, 2013].

Summary. In this chapter we have presented some examples of computational models. Similarly to mathematical models, computational models can be classified in various ways, for example, the behavior of the model, which can be predictable or ruled by chance, or the data, which can be discrete or continuous. These two classifications combine in four different types of computational modeling approaches, each one suitable for a situation or system being modeled. Some examples of these computational models have also been discussed, for example, a system which presents stochastic behavior and discrete data can be effectively modeled as a probabilistic boolean model.

Chapter 4

Model Checking

Outline. In this chapter we present some of the background on model checking that is relevant to this dissertation. We introduce symbolic model checking, Kripke structures, Binary Decision Diagrams and its temporal logics, Computational Tree Logic (CTL and CTL*) and Linear Time Temporal Logic (LTL). We also describe probabilistic model checking, and its probabilistic logics, Probabilistic Computational Tree Logic (PCTL), Continuous Stochastic Logic (CSL) and reward-based extensions. Finally, we consider the model checkers of Markov chains, including the PRISM model checking tool, and outline the PRISM modeling language and property specification.

4.1 Introduction

The model checking technique is a formal verification procedure to automatic and exhaustively check if a given model of a system respects a given specification.

The systems usually modeled are hardware and software ones, such as the correction of circuit designs (such as logical and arithmetic units, processor pipelines) and critical software (such as aircraft, spacecraft, and elevators), respectively. However, other applications such as reliability of communication protocols, and more recently biological systems.

The specification is often given in special types of logics, such as temporal and probabilistic logics, which allow specifying properties about the sequence and probability of certain events, respectively. The specification often contains safety requirements such as the absence of deadlocks and critical states that can cause the system to crash.

The first section presents the symbolic version of model checking, along with its related concepts such as binary decision diagrams and temporal logics. This section is largely based on [Clarke et al., 1999] and [Song, 2004].

The probabilistic version called probabilistic model checking (PMC) is described in the following section, including its special representation using multi-terminal binary decision diagrams and more appropriate probabilistic logics. This section is largely based on [Parker, 2002] and [Crepalde, 2011].

PMC is used in this dissertation using the PRISM model checker to study a biological model of the sodium-potassium-pump and its interactions with the toxin palytoxin. Therefore, we have included an additional discussion on other available PMC tools, as well as a brief description of PRISM and its modeling and property languages.

4.2 Symbolic Model Checking

This technique was first proposed independently by Edmund. M. Clarke and Ernest. A. Emerson [Emerson and Clarke, 1980; Clarke and Emerson, 1982; Clarke et al., 1986] and by Jean P. Queille and Joseph Sifakis [Queille and Sifakis, 1982]. In 2007, Clarke, Emerson and Sifakis shared the Turing Award for their contribution on founding the field of model checking.

The systems are modeled in a finite state machine, described in a precise high level modeling language, and properties are specified in temporal logics. Given a model M, a set of initial states S_0 and a property ϕ, the verification algorithm automatically checks if the models respects the property ϕ. This is performed by exhaustively exploring the transitions between states, checking the specified properties.

These are other methods such as tests and simulations to analyze and check systems. However, the model checking technique offers several advantages. First, it is completely automatic: after modeling the system and specifying the desired properties, the model checker performs the analysis without human interaction.

Furthermore, model checking guarantees that the model respects the specified properties since the state space is completely searched. This allows the detection of even small errors which might pass unnoticed by other techniques such as emulation, simulation and tests. Finally, if the model does not respect some given property, the model checker usually produces a counter-example, which is useful to understand and correct the error.

However, there is no free lunch. This exhaustive exploration of the model state space causes the classical model checking problem of the state space explosion. The number of states grows exponentially with the number of variables. For example, the composition of N variables of size k each yields k^N states. This is an active research topic, and several efforts have been made to reduce the state space. The first and one of the most important contributions has been made in [McMillan, 1992], which proposed using Binary Decision Diagrams (BDDs), originally created by [Bryant, 1986], to symbolically represent the transition relations between states.

This implicit representation encodes each state as the attribution of boolean values to the system variables. Therefore, the transitions can be expressed as boolean formulas in terms of two sets of variables, one set encoding the previous state and another set encoding the current state. This avoids the explicit construction of graphs of system states, which provided a more compact description of the model, increasing the size of models from 10^5 to 10^{20} states.

Furthermore, several improvements have been made to cope with the state space explosion, allowing the verification of systems with 10^{120} states, such as Partial Order Reduction [Godefroid et al., 1996], Symmetry Reduction [Clarke et al., 1998], Compositional Reasoning [Berezin et al., 1998], Statistical or Approximate Model Checking [Younes, 2005; Clarke et al., 2008] and Bisimulation Minimisation [C. Dehnert and Parker, 2013].

4.2.1 Kripke Structure

The representation used in symbolic model checking to capture the behavior of a system is a directed state transition graph called **Kripke structure**. It is a variation of a non-deterministic finite state machine whose states or nodes represent the reachable states of the system and whose edges represent allowed transitions between states.

Atomic propositions of the model are boolean expressions over the variables, constants and other symbols of model. These propositions assume the value of either true or false. Atomic propositions are self contained and do not include other propositions.

Let AP be the set of atomic propositions. A Kripke structure M over AP is a 4-tuple $M = \langle S, I, R, L \rangle$, as defined by [Clarke et al., 1999], where:

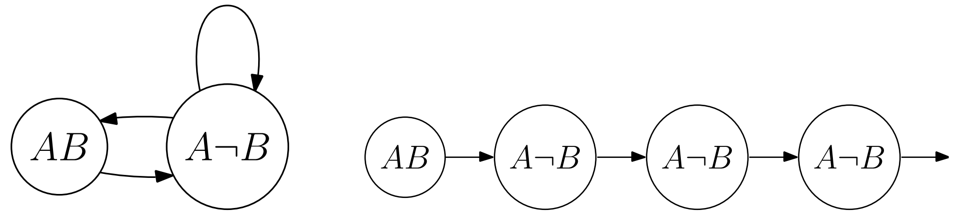
- S is a finite set of states.
- $I \subseteq S$ is the set of initial states.
- $R \subseteq (S \times S)$ is a transition relation that must be total, i.e., for each state $s \in S$, there is a state $s' \in S$ such that $R(s, s')$.

- $L : S \rightarrow 2^{AP}$ is a function that labels each state with the set of atomic propositions which are true in that state.

A *path* in a Kripke structure M from a state s_0 is a an infinite sequence of states $\pi = s_0 s_1 s_2 ...$ such that $s_0 = s$ and the relation $R(s_i, s_{i+1})$ holds for all $i \geq 0$.

The Figure 4.1 shows a graphical representation of a Kripke structure for a model and a computational path in its structure. The components of M which define the Kripke structure are:

- $S : \{S_0, S_1\}$
- $I : \{S_0\}$
- $R : \{(S_0, S_1), (S_1, S_0), (S_1, S_1)\}$
- $L(S_0) = \{A, B\}, L(S_1) = \{A, \neg B\}$



(a) Generic Kripke structure with two states.

(b) A computational path in the Kripke structure.

Figure 4.1: A Kripke structure and a computational path in it.

4.2.2 First Order Representations

As described by Clarke et al. [1999], a Kripke structure can be represented through boolean functions, which employ logical operators such as negation ($\neg$), disjunction ($\vee$), conjunction ($\wedge$) and implication ($\implies$).

Let $V = \{v_1, \cdots, v_n\}$ be the finite set of **system variables** and $D = \{D_1, \cdots, D_n\}$ be the finite set of **domains** of these variables, such as D_i represents the set of possible values for v_i. A **system state** is the evaluation of all system variables at a specific instant in time.

For example, suppose a system with three boolean system variables (a three bit counter), i.e., let $V = \{v_1, v_2, v_3\}$ and $D_1 = D_2 = D_3 = \{0, 1\}$. There are eight possible system states ($2^3 = 8$), such as ($v_1 = 0, v_2 = 0, v_3 = 0$), ($v_1 = 0, v_2 = 0, v_3 = 1$) and ($v_1 = 0, v_2 = 1, v_3 = 1$). Their respective representations as boolean functions are $(\neg v_1 \wedge \neg v_2 \wedge \neg v_3)$, $(\neg v_1 \wedge \neg v_2 \wedge v_3)$ and $(\neg v_1 \wedge v_2 \wedge v_3)$, where v_i is an 1-valued variable (or true) and $\neg v_i$ is a 0-valued variabled (or false). These representations are **symbolic**, therefore the name symbolic model checking.

The transitions between system states must also be represented as boolean functions. In order to do that, a second set of system variables is created to represent the system variables in the next (future) state. Therefore, V is the set of system variables in the current state, and V' is the set of system variables in the next (future) state. For each $v \in V$, a variable $v' \in V'$ for the next state is created. A transition can be viewed as an ordered pair for the evaluation of system variables in V and V', which can be represented as a conjunction of boolean functions.

Let f be the boolean function for the current system state s and f' be the boolean function for the next (future) system state s', then the transition from state s to state s' is represented by the conjunction of both boolean functions, therefore, $f \wedge f'$. For example, the transition from a state where $(v_1 = 0, v_2 = 0, v_3 = 0)$ to a state where $(v_1 = 0, v_2 = 0, v_3 = 1)$ is represented by the boolean function $(\neg v_1 \wedge \neg v_2 \wedge \neg v_3) \wedge (\neg v_1' \wedge \neg v_2' \wedge v_3')$, as shown in Figure 4.2.

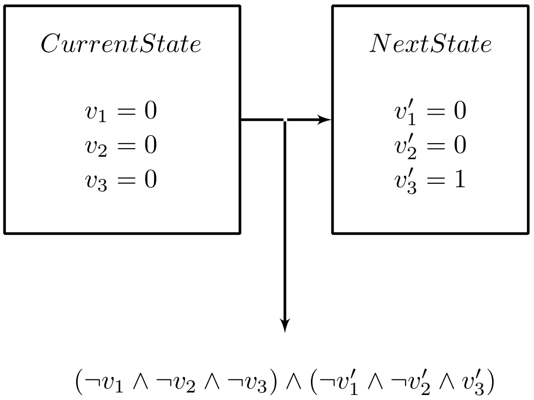

$CurrentState$
$v_1 = 0$
$v_2 = 0$
$v_3 = 0$
$NextState$
$v_1' = 0$
$v_2' = 0$
$v_3' = 1$
$(\neg v_1 \wedge \neg v_2 \wedge \neg v_3) \wedge (\neg v_1' \wedge \neg v_2' \wedge v_3')$

Figure 4.2: The symbolic representation of a transition.

Boolean functions can represent a set of states and a set of transitions. If $f_1, f_2, \dots f_n$ represent all the transitions of a Kripke structure, the boolean function for the set of all transitions is described by the disjunction of all f_i, i.e., $f_R = f_1 \vee f_2 \vee \cdots \vee f_n$. The same reasoning is used to make the set of all system states of a Kripke structure.

The coupling of several transitions into a simple boolean function, which simplifies the process of graph traversal, is inherent to the representation of boolean functions as BDDs, covered in the next section. This is one of the main reasons for the efficiency of BDDs in symbolic model checking algorithms.

Furthermore, the set of atomic propositions AP must be described in order to create specifications for the system. An atomic proposition is an expression of the form $v = d$, where $v \in V$ and $d \in D$. A proposition $v = d$ will be true in a system state s, if the boolean function of s becomes true when v assumes the value d.

In order to illustrate first order representations, a symbolic representation of a Kripke structure is presented in Figure 4.1. In this example, let $V = \{A, B\}$ and $D_A = D_B = \{0, 1\}$ for the model. Furthermore, it is necessary to create two variables $V' = \{A', B'\}$ to represent future states. The symbolic representations of the system states s_0 and s_1 are given by the boolean functions $A \wedge B$, and $A \wedge \neg B$, respectively. Finally, the transition from state s_0 to state s_1 is given by $R(s_0, s_1) \equiv A \wedge B \wedge A' \wedge \neg B'$.

The boolean function which represents the complete transition relation of the model is composed of three disjunctions, representing the number of transitions of the Kripke structure:

$$(A \wedge B \wedge A' \wedge \neg B') \vee (A \wedge \neg B \wedge A' \wedge B') \vee (A \wedge \neg B \wedge A' \wedge \neg B')$$

In this example, the labeling function L contains the following mappings: $L(s_0) = \{A = 1, B = 1\}$ and $L(s_1) = \{A = 1, B = 0\}$. As $A = 1$ and $B = 1 \in L(s_0)$, when A and B assume, respectively, the values 1 and 1 in system state s_0, the boolean function which represents this state $(A \wedge B)$ becomes true.

Although previous definitions have considered only a boolean domain $D = \{0, 1\}$ for all variables, it is possible to use other domains such as integer values by simply encoding each element to a boolean domain. The encoding is performed by using j bits to encode the domain D_i of each variable v_i as a binary number and represent v_i as j boolean variables.

For example, suppose that the variables v_i assume integer values in the domain $D = \{0, 1, 2, 3, 4, 5, 6, 7\}$, then only three bits are necessary to encode each variable v_i. Therefore, the atomic proposition $(v_1 = 2)$ can be represented by the conjunction of three new boolean variables $(\neg v_{1.1} \wedge v_{1.2} \wedge \neg v_{1.3})$, where each one corresponds to a bit of the binary codification of the integer value of 2 (010). Finally, the boolean function $(v_1 = 2) \wedge (v_2 = 3) \wedge (v_3 = 4)$ would be represented in a binary domain as shown below:

$$(\neg v_{1.1} \wedge v_{1.2} \wedge \neg v_{1.3}) \wedge (\neg v_{2.1} \wedge v_{2.2} \wedge v_{2.3}) \wedge (v_{3.1} \wedge \neg v_{3.2} \wedge \neg v_{3.3})$$

4.2.3 Binary Decision Diagrams

A data structure called Binary Decision Diagram (BDD) is used to represent boolean functions in a compact, efficient and canonical form, as first described in [Bryant, 1986]. A BDD is compact because it discards redundant information; efficient because it allows rapid graph traversal; and canonical because it is easy to check if two boolean functions are equivalent.

This data structure is often used in symbolic model checking to represent finite states systems, although there are other representations, such as explicit-state representation (all variables are always represented) and conjunctive normal form (a conjunction of disjunctions).

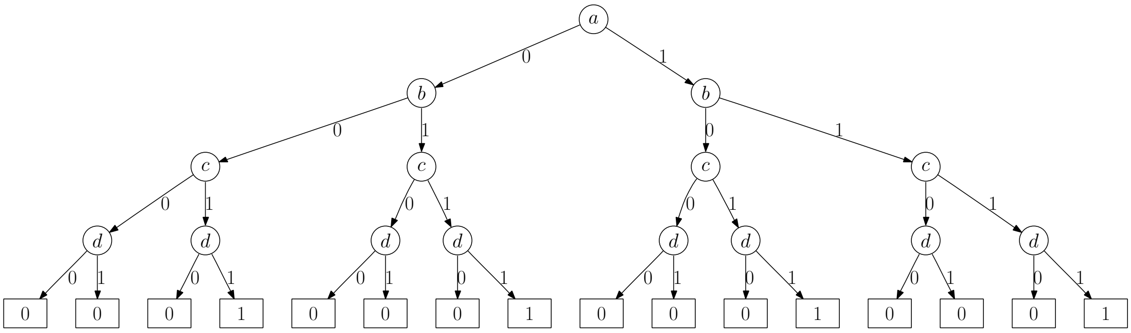


Figure 4.3: A binary decision tree for the boolean function $(a \wedge b) \vee (c \wedge d)$.

BDDs represent boolean functions as a special type of binary decision trees, which is a directed tree with two types of vertices: terminal and non-terminal vertices (shortened as terminal and non-terminal, respectively). In a binary decision tree, a non-terminal vertex v is labeled with a boolean variable, given by $var(v)$, which has two successors: $zero(v)$, when $var(v) = 0$, and $one(v)$, when $var(v) = 1$. A terminal vertex v assumes only the value zero or one (simply zero- or one-terminal, respectively), given by function $value(v)$. The tree edges are labeled with the value zero or one.

The Figure 4.3 shows an example of a binary decision tree for the boolean function $(a \wedge b) \vee (c \wedge d)$. A path in the tree starts at the root vertex and the choice between $zero(v)$ and $one(v)$ directs to a terminal that represents the function evaluation. In the example, the variable evaluation $(a = 1, b = 0, c = 1, d = 0)$ leads to a zero-terminal, which means that the boolean function is false for that assignment, while a different assignment leading to a one-terminal means that the function is true.

The binary decision trees can have a lot of redundant information. For example, in the Figure 4.3 there are eight subtrees whose roots are labeled with a boolean function d. However, only three of them are unique: in the first unique subtree, assigning any value to the variable d leads to a zero-terminal. On the other hand, in the second unique subtree, assignment any value to d leads to a one-terminal. Finally, in the third unique subtree, the assignment of zero to d leads to a zero-terminal, while assigning one to d leads to a one-terminal.

Therefore, a BDD is obtained by merging identical subtrees and eliminating nodes with repeated information. The resulting structure is a directed acyclic graph (DAG) called BDD which, unlike trees, allow shared vertices and substructures. It is worth to emphasize the canonical aspect of BDDs, which means that two functions are equal if, and only if, their associated BDDs are isomorphic[1].

[Bryant, 1986] first showed how to obtain a canonical representation of boolean functions by imposing two restrictions on BDDs. Foremost, the variables must appear in the BDD in the same order along the path from its root to a terminal. Secondly, isomorphic subtrees or vertices should not exist in the BDD.

The first restriction is satisfied by fixing an order for the variables which label the vertices of the BDD ($var_1 < var_2 < \cdots < var_n$). This means that if a vertex u precedes a non-terminal v (starting from the BDD root), then the variable which labels the vertex u precedes the variable which labels v in the order ($var(u) < var(v)$).

The second restriction is satisfied by repeatedly applying three rules of transformation that do not change the function represented by the BDD:

1. Remove duplicated terminals: keep only two terminals, one zero-terminal and another one-terminal. Redirect all input edges from the removed terminals to these two unique terminals;

2. Remove duplicated non-terminals: if two non-terminals u and v are labeled with the same variable ($var(u) = var(v)$) and have the same successors ($zero(u) = zero(v) \wedge one(u) = one(v)$), remove the vertex v and redirect its input edges to the vertex u;

3. Remove redundant children: if a non-terminal v has $zero(v) = one(v)$, then remove v and redirect all of its input vertices to $zero(v)$.

The canonical form of the BDD, obtained by imposing these two restrictions, is called Ordered Binary Decision Diagram (OBDD). The Figure 4.4 shows the OBDD for the binary decision tree of 4.3, considering the variable order $a < b < c < d$.

Although BDDs have several advantages, it has its own disadvantages. The main one is the order of the variables which appear in the boolean function being represented. Depending on it, the BDD can be heavily compressed or completely

[1] Two BDDs are isomorphic if there is an injective function h that maps terminals and non-terminal from a BDD to the other.

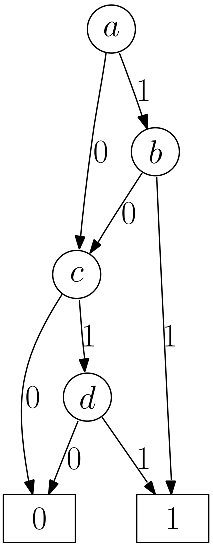


Figure 4.4: A reduced binary decision diagram.

redundant. However, the problem of choosing the variable ordering which minimizes the BDD size is co-NP-complete [Bryant, 1986].

There are heuristics to approximate this problem, such as the one presented by [Bollig and Wegener, 1996], whom also briefly reviews several other heuristics based on local search and simulated annealing. Nonetheless, variable orderings are often found empirically since one can expect that it is related to the semantics of the model.

Another issue with BDDs is the space complexity. Since BDDs are essentially an exhaustive representation of the model, in the worst case it is exponential to the number of variables of the boolean function [Bryant, 1986].

4.2.4 Temporal Logic

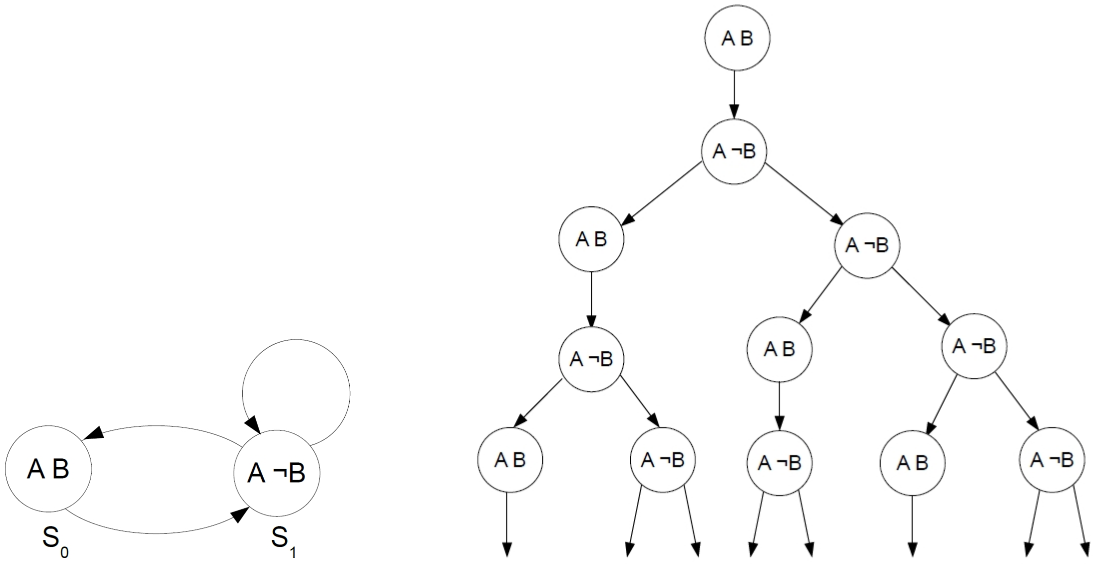
There are several different types and flavors of logics, from the classical and first mathematical formalism created by George Boole, and named after him, the boolean algebra, to other non-orthodox logics, such as the more modern modal logic, the epismetic logic, or the logic of knowledge. These logics are appropriate to different types of systems.

However, reactive and parallel systems present additional challenges on their reasoning. They often can not be understood from its current state, demanding the analysis of a sequence of events. For example, one simple reactive system such as an alternating bit communication protocol can be reasoned only using temporal properties, such as “if a message is sent, it is eventually received”. The message can be received in the next time unit, or in the next ten time units. Therefore, to reason about such systems we need to be able to state temporal properties.

One could describe and reason logical propositions in terms of time and the sequence of events, which is called temporal logic and its applied to model checking, where the behavior of the system being modeled is represented as a state-transition graph such as the BDD reasoning on the time evolution of the model.

There are several temporal logics, such as the Computation Tree Logic (CTL and CTL*) and Linear Time Logic (LTL). Each logic has its own set of logical operators. The main difference between LTL and CTL, is that in LTL the reasoning is on an infinite computational path, while in CTL the reasoning on a tree of infinite paths starting at a root node s_0. The CTL* is a superset of CTL and LTL.

Temporal logics (TL) borrow quantifiers from predicate logic (or first-order logic), such as for all ($\forall$) and exists ($\exists$), naming them path quantifiers, which allow reasoning on the computational tree created from unfolding the Kripke structure (Figure 4.5). They also introduce temporal operators, which allow reasoning on the sequence of states. TL can also use logical operators from boolean algebra, such as negation ($\neg$), disjunction ($\vee$), conjunction ($\wedge$) and implication ($\implies$).



(a) A Kripke structure.

(b) An infinite computational tree.

Figure 4.5: Unfolding a Kripke structure in an infinite computational tree.

Path quantifiers reason on a computational path, or sequence of states. They are used to specify that every computational path from the current state respects the given property. Path quantifiers are shown below. LTL does not support the **E** path quantifier, because there is a single computational path.

- The "For All" path quantifier: **A** Φ – for every path, Φ is true.
- The "Exists" path quantifier: **E** Φ – there exists a path where Φ is true.

Temporal operators reason on a sequence of states. They are used to check that one or more states hold the given property. The Figures below show the computational paths associated with each property. The black dot represents the state in which the property ϕ is true. Only for Figure 4.9, the black and red dots represent when the properties ϕ_1 and ϕ_2 are true, respectively.

- The "Eventually" operator: **F** ϕ – ϕ is true in a future state (Figure 4.6).

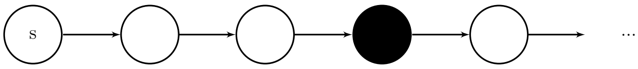


Figure 4.6: The "Eventually" operator: **F** ϕ.

- The "Globally" operator: **G** ϕ – ϕ is true in all future states (Figure 4.7).

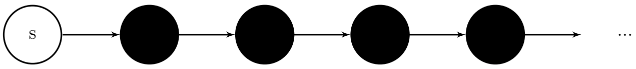


Figure 4.7: The "Globally" operator: **G** ϕ.

- The "Next" operator: **X** ϕ – ϕ is true in the next state (Figure 4.8).

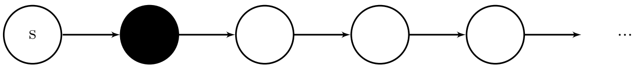


Figure 4.8: The "Next" operator: **X** ϕ.

- The "Until" operator: ϕ_1 **U** ϕ_2 – ϕ_1 is true until ϕ_2 becomes true (Figure 4.9).

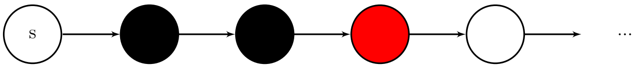
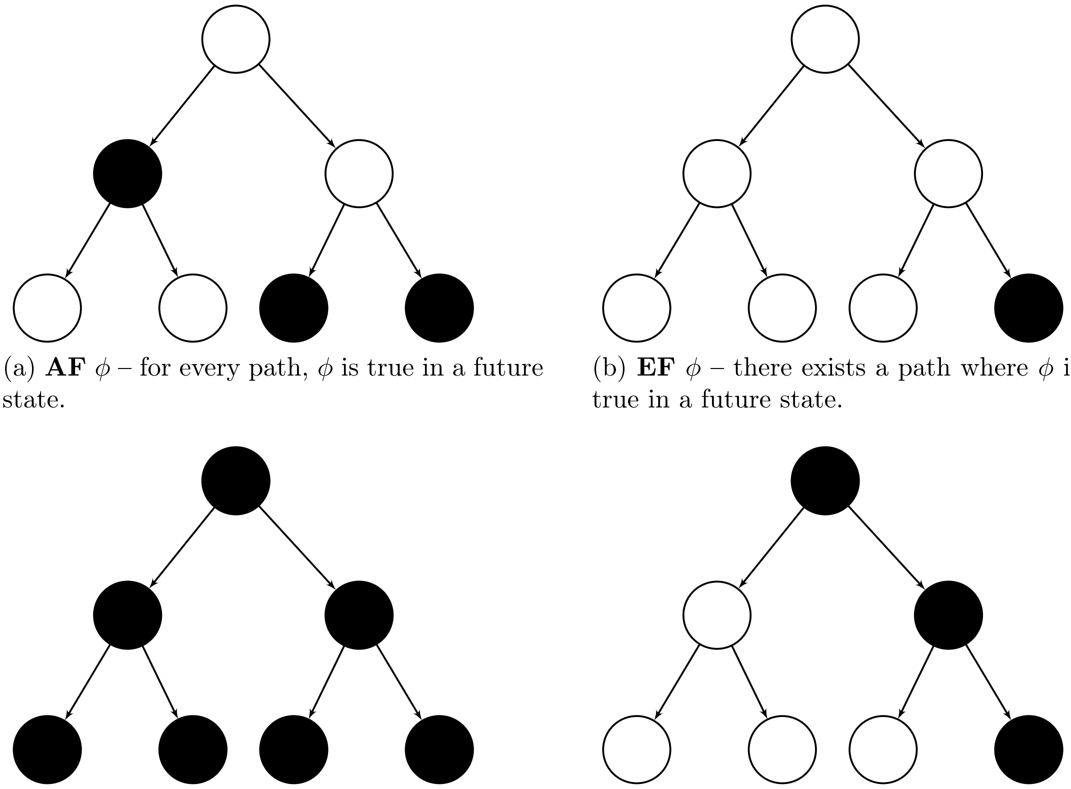


Figure 4.9: The "Until" operator: ϕ_1 **U** ϕ_2.

In CTL, temporal operators must be preceded by a path quantifier, for example, **EG** ϕ, which states that exists a path where ϕ is always true (Figure 4.10d). The most commonly used CTL operators are shown below (Figure 4.10).

Property **AF** ϕ checks if a property ϕ is eventually observed in all paths starting at the current state (Figure 4.10a). Property **EF** ϕ, shown in Figure 4.10b, checks if



(a) **AF** ϕ – for every path, ϕ is true in a future state.

(b) **EF** ϕ – there exists a path where ϕ is true in a future state.

(c) **AG** ϕ – for every path, ϕ is true in all future states.

(d) **EG** ϕ – there exists a path where ϕ is true in all future states.

Figure 4.10: Different types of combinations of temporal operators.

exists a path where the property ϕ is eventually observed. Property **AG** ϕ checks if a property ϕ is always observed in all paths starting at the current state (Figure 4.10c). These properties could be used to check safety aspects of the model, such as situations which should never occur, for example, "the system never crashes!" (**AG** $!crash$).

There are other operators which are not that commonly used, such as weak until (ϕ_1 **W** ϕ_2) and past time operators (**H**, **P** and **S**), i.e., events that happened before the current state. There are several other logics, such as Linear Tree Logic (LTL). In the next Section we will discuss probabilistic model checking, therefore one needs appropriate temporal logics, or probabilistic logics.

4.3 Probabilistic Model Checking

Probabilistic Model Checking is a formal, exhaustive and automatic technique for modeling and analyzing stochastic systems. PMC checks if the model satisfies a set of properties given in special types of logics.

A stochastic system M is usually a Markov chain or a Markov decision process – in fact, our system is a Continuous-time Markov chain (CTMC). This means that the system satisfies the Markov property, i.e., its behavior depends only on its current state and not on the whole system history, and each transition between states occurs in real-time.

Given a property ϕ expressed as a formula in a probabilistic temporal logic, PMC attempts to check whether a model of a stochastic system M satisfies the property ϕ with a probability greater than or equal to a probability threshold $\theta \in [0, 1]$.

Tools called models checkers such as PRISM [Kwiatkowska et al., 2011] attempt to solve this problem. It requires two inputs: a modeling description of the system, which defines its behavior (for example, through the PRISM language), and a probabilistic temporal logic specification of a set of desired properties (ϕ).

The model checker builds a representation of the system M, usually as a graph-based data structure called Binary Decision Diagrams (BDDs), which can be used to represent boolean functions. States represent possible configurations, while transitions are changes from one configuration to another. Probabilities are assigned to the transitions between states, representing rates of negative exponential distributions.

Let $\mathbb{R}_{\geq 0}$ be the set of positive reals and AP be a fixed, finite set of atomic propositions used to label states with properties of interest. A labeled CTMC $\mathcal{C}$ is a tuple (S, $\bar{s}$, $\mathbf{R}$, L) where:

- S is a finite set of states;
- $\bar{s} \in S$ is the initial state;
- $\mathbf{R} : S \times S \rightarrow \mathbb{R}_{\geq 0}$ is the transition rate matrix, which assigns rates between each pair of states;
- $L : S \rightarrow 2^{\mathrm{AP}}$ is a labeling function which labels each state $s \in S$ the set $L(s)$ of atomic propositions that are true in the state.

The probability of a transition between states s and s' being triggered within t time-units is $1 - e^{-\mathbf{R}(s,s') \cdot t}$. The elapsed time in state s, before a transition occurs, is exponentially distributed with the *exit rate* given by $E(s) = \sum_{s' \in S} R(s, s')$. The probability of changing from state s to s' is given by $P(s, s') = \frac{\mathbf{R}(s,s')}{E(s)}$ [Parker, 2002]. Finally, the probability of changing from s to s' in t time units is given by [Clarke et al., 2008]:

$$P(s, s', t) = \frac{\mathbf{R}(s, s')}{E(s)} \times (1 - e^{-\mathbf{R}(s,s')\cdot t})$$

A computational path of a CTMC model starting at state s_0 is an infinite sequence $\pi = s_0 t_0 s_1 t_1 ...$, where $s_i \in S$, $t_i \in \mathbb{R}_{\geq 0}$ is the time spent at state s_i and $R(s_i, s_{i+1}) > 0$ for all $i \geq 0$. The Figure 4.11 shows the graph of the CTMC model below:

- $S = \{S_0, S_1, S_2\}$;
- $S_0 = \{S_0\}$;
- $R(S_0, S_1) = \lambda_1$, $R(S_1, S_2) = \lambda_2$, $R(S_2, S_0) = \lambda_3$ and $R(S_1, S_0) = \lambda_4$;
- $L(S_0) = \{A, B\}$, $L(S_1) = \{A, \neg B\}$ and $L(S_2) = \{\neg A, B\}$.

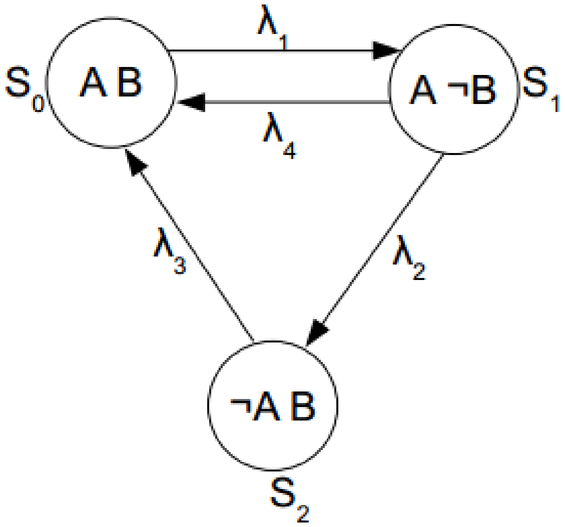


Figure 4.11: An example of a CTMC model.

4.3.1 Probabilistic Logics

Properties can be expressed quantitatively as "What is the probability of ATP binding to the pump?" or qualitatively as "ATP eventually depletes", offering valuable insight over the system behavior.

Properties are specified using the Continuous Stochastic Logic (CSL) [Kwiatkowska et al., 2008], which is based on the Computational Tree Logic (CTL) and the Probabilistic Computation Tree Logic (PCTL). The syntax of

CSL formulas is the following:

$$\Phi \ ::= \ true \mid a \mid \neg\Phi \mid \Phi \wedge \Phi \mid \mathcal{P}_{\trianglelefteq p}[\phi] \mid \mathcal{S}_{\trianglelefteq p}[\phi]$$

$$\phi \ ::= \ \mathbf{X}\ \Phi \mid \Phi\ \mathbf{U}^{I}\ \Phi$$

where a is an atomic proposition, $\trianglelefteq \in \{>, <, \geq, \leq\}$, $p \in [0, 1]$ and I is an interval of $\mathbb{R}_{\geq 0}$.

There are two types of CSL properties: transient ($\mathcal{P}_{\trianglelefteq p}$) and steady-state ($\mathcal{S}_{\trianglelefteq p}$). In this work we are interested in transient or time related properties. A formula $\mathcal{P}_{\trianglelefteq p}\,[\phi]$ states that the probability of the formula ϕ being satisfied from a state respects the bound $\trianglelefteq p$. Path formulas use the $\mathbf{X}$ (next) and the $\mathbf{U}^{I}$ (time-bounded until) operators. For example, formula $\mathbf{X}\,\Phi$ is true if Φ is satisfied in the next state.

This can be applied to check if one state leads to another with a probability p, for example, state "open-in" is followed by state "open-out" with at least 10% chance: $\mathcal{P}_{\geq 0.1}[\text{``}open - in'' \implies \mathbf{X}\ \text{``}open - out'']$.

4.3.2 Rewards

PRISM also allows including **rewards** in the model, which are structures used to quantify states and transitions by associating real values to them. The state rewards are counted proportionately to the elapsed time in the state, while transition rewards are counted each time the transition occurs. In PRISM, rewards are described using the following syntax:

rewards "*reward name*"

...

endrewards

Each reward is specified using multiple reward commands which follow the syntax below.

$$[sync]\, guard\ :\ reward\,;$$

Reward commands describe state rewards and transition rewards, respectively. The predicate which must be observed is the *guard*. The *sync* is a label used to synchronize a set of commands into a single transition in the system. Finally, *reward* is an expression, which can contain variables and constants from the model, and when evaluated it counts for the reward.

rewards “$E1$”
$[r1]\, E1 = 1\ :\ 1\,;$
endrewards

Figure 4.12: An example of a state reward.

One example of a state reward is the pump open to the intracellular side (`E1`, in our model), described in Figure 4.12. The *reward* is one, since it is essentially counting the number of times when that particular reward was observed. The *guard* is the condition which must be observed – `E1=1`, that state must be present. The *sync* `r1` is used to synchronize with another module which defines the rate of that reaction. Therefore, the cumulative reward represents how many times the pump was open to the intracellular side.

Reward properties can be applied to states and transitions. For example, “What is the expected reward for the phosphorylated pump open to the external side of the cell at time T?”.

This reward can be instantaneous, obtaining its value at the given time through the property $\mathcal{R}_{=?}[\mathcal{I}^{=t}]$, or accumulated, calculating its value until the given time, using the property $\mathcal{R}_{=?}[\mathcal{C}^{<=t}]$. In this work, we have used cumulative rewards because they show the reward history, giving a better intuition on its role on the model, while instantaneous rewards yield only local information.

One can obtain the probability of a state reward by dividing it to the sum of all state rewards. The same procedure can be applied to transitions (or chemical reactions, in our model).

Rewards of paths in a Continuous-time Markov chain are summations of state rewards along the path and transition rewards for each transition between these states. State rewards are interpreted as the rate at which rewards are accumulated, essentially counting them, i.e. if t time units are spent in a state with state-reward r, the accumulated reward in that state is $r \times t$.

4.4 Model Checkers for Markov Chains

There are several model checkers for each stochastic process. Each tool presents its own features, some more complete than others, such as the case of PRISM which covers several distinct types of models. Others implement specific techniques, for example, the parametric exploration of PARAM.

Some of the main model checkers for Markov chains, either discrete-time (DTMC) or continuous-time (CTMC), or even extensions of these types, include: PRISM [Kwiatkowska et al., 2011]; MRMC [Katoen et al., 2011]; the PEPA Eclipse plug-in [Tribastone, 2007]; PARAM [Hahn et al., 2010]; INFAMY [Hahn et al., 2009] and CASPA [Riedl et al., 2008].

The MRMC (Markov Reward Model Checker) uses explicit-state (and approximate) model checking for DTMCs, CTMCs and CTMDPs (continuous-time Markov decision processes) with rewards against PC(R)TL (Probabilistic Computation Reward Tree Logic) and CS(R)L (Continuous Stochastic Reward Logic). It also supports bisimulation [Katoen et al., 2011].

The PEPA Eclipse Plug-in project supports CSL (Continuous Stochastic Logic) model checking (plus steady-state/ODE analysis and abstraction techniques) for the stochastic process algebra PEPA [Tribastone, 2007]. PARAM uses parametric probabilistic model checking of DTMCs [Hahn et al., 2010]. INFAMY performs model checking of infinite-state CTMCs [Hahn et al., 2009]. CASPA employs symbolic (MTBDD-based) model checking of (extended) stochastic labeled transition systems against Stochastic Propositional Dynamic Logic (SPDL) [Riedl et al., 2008].

4.4.1 PRISM

PRISM supports diferent types of models, properties and simulators [Kwiatkowska et al., 2011]. It has been largely used in distinct fields, e.g. communication and media protocols, security and power management systems. We have used PRISM in this work for several reasons, which include: exact PMC in order to obtain accurate results; Continuous-time Markov Chain (CTMC) models, suited for our field of study; rich modeling language that allowed us to build our model; and finally property specification using Continuous Stochastic Logic (CSL), which is able to express qualitative and quantitative properties.

4.4.1.1 Modeling

One way to model the set of chemical reactions described in Figure 4.13 in the PRISM tool is presented in the Figure 4.14.

A description of a CTMC model in PRISM must begin with the keyword `ctmc`. It is composed of modules, which have their states represented by a set of variables which assume a finite set of values. In the example, there is a module for each ligand

Reactions:
1. A + B $\rightleftharpoons$ A : B (complex formation)
2. A $\rightharpoonup$ (degradation)

Reaction rates

- complex formation	A : B : r1
- complex release	A : B : r2
- ligand degradation	: r3

Figure 4.13: Example of a chemical reactions system.

in the reactions: `m_A`, `m_B` and `m_AB`. In each module there is a variable which describes if that ligand is present or not, indicated by 1 or 0, respectively. In this example there can be a maximum of one molecule for each of the three ligands. Initially, only the molecules A and B are present.

The module `start` is responsible for defining the rate constants for each of the three possible reactions of the system: `bind` (binding `A` and `B`), `rel` (release of the molecules in the `AB`) and `deg` (degradation of `A`).

```
ctmc

const double r1=1;
const double r2=1;
const double r3=0.1;

module m_A
  a: [0..1] init 1;
  // 0 - degraded or bound, 1 - livre

  [bind] a=1 -> 1 : (a’=0);
  [rel] a=0 -> 1 : (a’=1);
  [deg] a=1 -> 1 : (a’=0);
endmodule
```

```
module m_B
  b: [0..1] init 1; // 0 - bound, 1 - free
  [bind] b=1 -> 1 : (b’=0);
  [rel] b=0 -> 1 : (b’=1);
endmodule

module m_AB
  ab: [0..1] init 0;
  // 0 - complex absent, 1 - complex present
  [bind] ab=0 -> 1 : (ab’=1);
  [rel] ab=1 -> 1 : (ab’=0);
endmodule

module start
  [bind] true -> r1 : true;
  [rel] true -> r2 : true;
  [deg] true -> r3 : true;
endmodule
```

Figure 4.14: PRISM model of the chemical reactions systems presented in Figure 4.13.

The behavior of the modules is defined by the transitions between states. These transitions are defined by commands expressed as $[action]\ g \rightarrow r : u$. This command indicates that if the predicate `g` (also known as conditions or guard) is observed (true), then the system will be updated by `u`, which is composed of one or more declarations expressed as $x' = \cdots$, indicating that the value of x is updated (x' is the value of variable x in the next state). The value of the rate at which the update will occur is defined by `r`.

PRISM also allows the synchronization between modules, through labels which must be in brackets at the start of the synchronized commands. Transitions in different modules using the same label happen simultaneously. The resulting rate is equals to the product of the individual command rates of each synchronized module. In the example, the formation of the complex AB (given by the reaction A + B $\xrightarrow{r1}$ AB) is represented by the commands labeled with bind. In order for that reaction to occur, the molecules A and B must be present (each equals to 1) and the complex AB must be absent (equals to 0). The final rate when this reaction occurs is $r1 \times 1 \times 1 \times 1$ and it is given by the product of the rates of the four commands labeled with bind. Similarly, the release of the complex AB, releasing the molecules A and B, is represented by commands labeled with rel and the degradation of molecule A is represented by the command labeled with deg.

Furthermore, PRISM allows extending CTMC models with **rewards**, structures which allow marking states or transitions with real values which can be used in quantitative properties. There are two types of rewards: state and transition rewards. State rewards are accumulated proportionally to the time spent in each state. Transition rewards are accumulated each time a transition occurs. Rewards allow estimating the expected time for the occurence of a given event. Each reward is associated with a name, which is used in the property specification, and uses the following syntax:

rewards "*reward_name*"
reward_body
endrewards

where reward_body of a state reward can be expressed as $g : v$, while a transition reward can be expressed as $[action]\ g : v$. In the state reward case, each PRISM state which satisfies the predicate g is marked with the real value v. In the transition reward case, each transition labeled with action, from PRISM states which satisfy the predicate g, is marked with the value v.

```
rewards 'time'
  true : 1;
endrewards
```

```
rewards 'freeA'
  a=1 : 1;
endrewards
```

```
rewards 'bind'
  [bind] true : 1;
endrewards
```

Figure 4.15: Rewards for the PRISM model of Figure 4.14.

The code of Figure 4.15 presents the definition of three rewards for our example. From left to right, the first one (named time) associates the value 1 to every system state, essentially counting the elapsed time. The second one (freeA) associates the value 1 only to states at which the molecule A is free (or $a = 1$). Finally, the reward bind associates the value 1 to each transition labeled with bind, i.e., every time the molecule A binds to B.

4.4.1.2 Property Specification

The properties to check CTMC models in PRISM must be specified in the Continuous Stochastic Logic (CSL), which is a temporal logic based on Computation Tree Logic (CTL), Probabilistic CTL (PCTL) and reward-based extensions [Kwiatkowska et al., 2008]. The CSL formulas use the following syntax:

$$\Phi \ ::= \ \mathit{true} \mid a \mid \neg\Phi \mid \Phi \wedge \Phi \mid \mathcal{P}_{\trianglelefteq p}[\phi] \mid \mathcal{S}_{\trianglelefteq p}[\phi]$$

$$\phi \ ::= \ \mathbf{X}\ \Phi \mid \Phi\ \mathbf{U}^{I}\ \Phi$$

where a is an atomic proposition, p $\in [0,\ 1]$ is a probability and I is an interval of R ≥ 0 (real non-negative numbers) at which the property must be met. The operators $\neg$ and $\wedge$ are logical ones, while **X** and **U** are temporal operators. The symbol $\trianglelefteq \in \{>, <, \geq, \leq\}$ represents the type of bound which the property must satisfy. For example, if $\trianglelefteq$ is $>$, the probability of the property must be higher than p.

There are two basic types of CSL properties: transient ($\mathcal{P}_{\trianglelefteq p}$) and steady-state ($\mathcal{S}_{\trianglelefteq p}$). Although we have specified a few steady-state properties, this work focused on transient and reward-based properties, therefore we will describe the semantics of only transient properties.

The formula $\mathcal{P}_{\trianglelefteq p}[\phi]$ is true in state s if the probability that ϕ is satisfied by a path starting at state s matches the bound $\trianglelefteq p$. Path formulas are built using the operators **X** (*next*) and $\mathbf{U}^{I}$ (*time-bounded until*). The path formula $\mathbf{X}\ \Phi$ is true if Φ is satisfied in the next state, while $\Phi^1\ \mathbf{U}^{I}\ \Phi^2$ is true if Φ^2 is satisfied at some time unit in the interval I and in all previous time units Φ^1 is satisfied.

Other operators can be created from this minimum set of CSL operators, such as the $\mathcal{G}$ (always) operator. The interval I can be omitted from the operators **U** and **F**, which means that I $= [0,\ \infty)$. Finally, one can also quantify the probability of a property ϕ by using the expression $=?$ instead of the bound $\trianglelefteq p$ ($\mathcal{P}_{=?}\ [\phi]$).

A few examples of transient properties are presented below considering the PRISM model previously discussed (Figure 4.13).

- $\mathcal{P}_{=?}\ [\ \mathbf{F}^{[0,T]}$ ab $= 1\ ]$: the probability of the molecule A binding to B in the first T time units. This is a $\mathbf{F}^{I}\Phi$ (eventually) property, where Φ is the atomic proposition $\texttt{ab} = 1$;

- $\mathcal{P}_{=?}$ [$\texttt{ab} = 0$ **U** ($\texttt{a} = 0 \wedge \texttt{ab} = 0$)] : the probability of the molecule A degrading before binding to the molecule B. This is a Φ_1 $\mathbf{U}^I$ Φ_2 (time-bounded until) property, where $\texttt{I} = [0, \infty)$ and Φ_1 and Φ_2 are the atomic propositions $\texttt{ab} = 0$ and ($\texttt{a} = 0 \wedge \texttt{ab} = 0$), respectively.

Furthermore, PRISM also allows to check the expected value of model rewards. Some of the properties of this type which will be used throughout this work have the forms $\mathcal{R}_{\trianglelefteq r}$ [I = t], $\mathcal{R}_{\trianglelefteq r}$ [**F** Φ] and $\mathcal{R}_{\trianglelefteq r}$ [**C** $\leq$ t], with $\texttt{r}$ and $\texttt{t} \in \mathbb{R}_{\geq 0}$.

The first property ($\mathcal{R}_{\trianglelefteq r}$ [I = t]) is true, starting from a state s, if the state reward at the instant t satisfies the bound $\trianglelefteq r$.

The second property ($\mathcal{R}_{\trianglelefteq r}$ [**F** Φ]) is true, starting from s, if the accumulated reward along the path until the point where Φ is true satisfies the limit $\trianglelefteq r$.

Finally, the third property ($\mathcal{R}_{\trianglelefteq r}$ [**C** $\leq$ t]) is true, starting at s, if the accumulated reward along the path at instant t satisfies the bound $\trianglelefteq r$.

Given the definition of a reward, its accumulated value along the path in a CTMC model is the sum of the state rewards along the path, plus the sum of the transition rewards between these states, both defined in the body of the same reward structure. The state reward associated with each state is $v \times t$, where t is the time spent at the state and v is the state reward associated with the state. If the bound is not specified, using the expression $=?$ one obtains the expected value of that reward.

A few examples of properties which obtain the reward values for the considered example are presented below, given the rewards defined in the Figure 4.15.

- $\mathcal{R}\{'freeA'\}_{=?}$ [**C** $\leq$ T] : the expected time that the molecule A is free (i.e., it is not associated with the B or has not degraded) during the first T time units;

- $\mathcal{R}\{'bind'\}_{=?}$ [**F** (a = 0 $\wedge$ ab = 0)] : the expected number of times that the molecule A binds to B before degrading;

- $\mathcal{R}\{'time'\}_{=?}$ [**F** (a = 0 $\wedge$ ab = 0)] : the expected time for the molecule A to degrade.

4.4.1.3 PMC Implementation

The techniques which are implemented in PRISM to check the properties of CTMC models with rewards include graph theory algorithms and numerical computation. The first ones are used on the graph structure which represents the Markov chain specified in the tool to determine, for example, the set of reacheable states or to check qualitative properties. In this case, the algorithms are executed on the BDDs as it happens on the non-probabilistic version of the model checking technique.

The numerical computation is required to solve a Markov chain and calculate the probabilities and reward values (quantitative properties). Iterative methods such as Jacobi and Gauss-Seidel are used to solve systems of linear equations and check properties such as $\mathcal{S}_{\trianglelefteq p}$, $\mathcal{P}_{\trianglelefteq p}$ $[\Phi_1 \ \mathbf{U} \ \Phi_2]$ and $\mathcal{R}_{\trianglelefteq r}$ $[\ \mathbf{F} \ \Phi]$. The iterative method known as uniformisation is used to calculate rewards and probabilities for properties which involve a time interval `I` or a specific time `t`: $\mathcal{P}_{\trianglelefteq p}$ $[\Phi_1 \ \mathbf{U}^I \ \Phi_2]$, $\mathcal{R}_{\trianglelefteq r}$ $[\ \mathbf{I} = \mathrm{t}]$ and $\mathcal{R}_{\trianglelefteq r}$ $[\ \mathbf{C} \leq \mathrm{t}]$. Further details on these techniques to solve Markov chains can be found in [Parker, 2002].

Furthermore, to determine the quantitative properties using numerical computation, PRISM allows the use of three data representations:

1. a generalization of BDDs, known as Multi-terminal BDDs (MTBDDs) to represent real-valued functions, given that matrices and real vectors are required. These data structures allows compact representations and efficient manipulation of big models because they explore their regularities. However, the computation is often slow;

2. explicit representation, as a sparse matrix, which allows a faster and direct computation, however, it can not deal with big models;

3. a hybrid approach, which extends the MTBDDs, allowing faster computations than compared to their original representation. In this case, MTBDDs are used to represent only the transition matrix, while the solution vectors of the iterative methods are represented by traditional real-valued vectors.

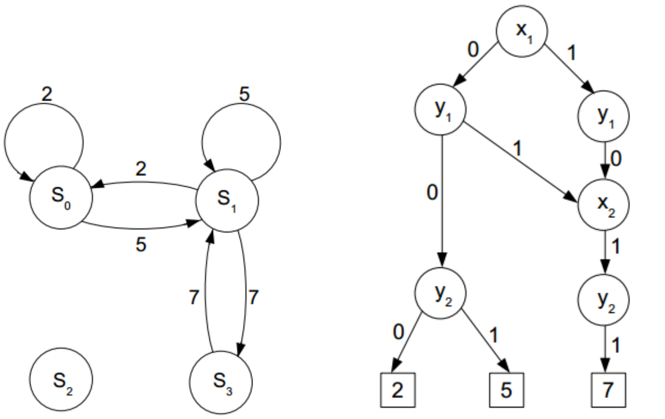
The adjacency matrix below illustrates the explicit representation of the transitions of the CTMC model of the Figure 4.17a.

$$\begin{pmatrix} 2 & 5 & - & - \\ 2 & 5 & - & 7 \\ - & - & - & - \\ - & 7 & - & - \end{pmatrix}$$

Figure 4.16: The adjacency matrix that explicitly represents the CTMC model.

Consider that the first line and first column of the matrix are 0-indexed. The entry $M(l, c)$ of the matrix indicates the value of the line l and column c. For example, the value of the entry $M(0,0)$ is 2, while the value of $M(1,3)$ is 7. The Figure 4.17b shows an example of a MTBDD which represents the transitions of the CTMC model of the Figure 4.17a.

In the representation through MTBDD, the binary variables x_1 e x_2 encode the indexes of the transition matrix lines, while y_1 e y_2 encode the indexes of the columns. For example, for the entry $M(1,3)$ of the transition matrix, the index 1 of the line is encoded through the binary representation (x_1,x_2) = (0,1) and the index 3 of the column is encoded as (y_1,y_2) = (1,1). Thus, by following the path x_1 = 0, y_1 = 1, x_2 = 1 and y_2 = 1 in the MTBDD (note that the order of the variables is $x_1 < y_1 < x_2 < y_2$) the terminal node 7 is reached, which is the value of the transition matrix for the entry $M(1,3)$.



(a) A CTMC model.

(b) The MTBDD for the CTMC model.

Figure 4.17: A CTMC and its MTBDD representation

Further details on the techniques used by the PRISM tool can be found in [Kwiatkowska et al., 2011, 2007, 2004].

Summary. In this chapter we have presented model checking, a technique used to model and analyze systems, verifying if those respect a set of properties. The symbolic model checking is presented, as well as its concepts and related subjects, such as the model representation using the Kripke structure and the data structure binary decision diagram. The properties to be verified are specified using special types of logic, the temporal logics, Computational Tree Logic (CTL and CTL*) and Linear Time Logic (LTL), which are also described. Using these logics, it is possible to specify different types of properties, such as “if a message is sent, then that message is eventually received”, or “a distributed system never enters a deadlock state”.

The probabilistic model checking is also presented, an extension to the previous technique which allows the modeling and analysis of stochastic systems. Different structures are necessary and were described, such as multi-terminal binary decision diagrams, and probabilistic logics, Probabilistic Computational Tree Logic (PCTL), Continuous Stochastic Logic (CSL), and reward-based extensions. Some model checkers, the tools which implement these techniques, are examined, such as PRISM, used in the modeling and verification of our models. The PRISM modeling language and property specification were briefly presented.

Chapter 5

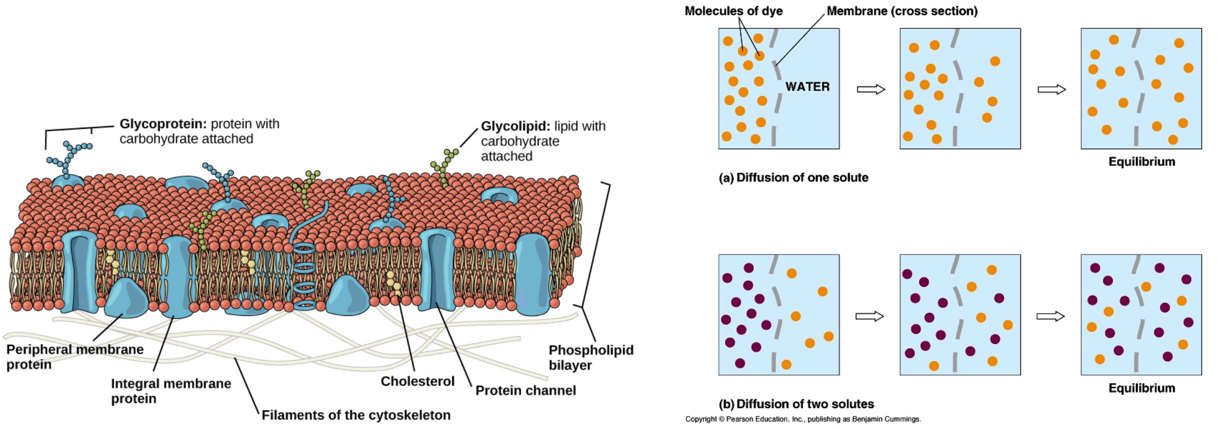
Transmembrane Ionic Transport Systems

Outline. In this chapter we describe transmembrane ionic transport systems, namely ion channels and ionic pumps, cell structures which are responsible for transporting ions between the inside and the outside of the cell. These systems can be affected by diseases, syndromes and toxins, which change their regular behavior and compromise the health of the cell, which has also been covered below.

5.1 Transmembrane Ionic Transport Systems

Animal cells contain structures called **transmembrane ionic transport systems**, which are responsible for transporting ions between the inside and the outside of the cell. Being a **transmembrane** structure means that it is located across the plasma membrane, which is a biological boundary that separates the interior of all cells from the outside environment, as shown in Figure 5.1a. Therefore, these systems function as doors which open and close to control the movement of ions.

Ions are atoms in which the total number of electrons are not equal to the total number of protons, giving the atom a net positive or negative electrical charge. They are necessary for several biological processes. Their concentration levels must be controlled in order to preserve the correct behavior of cells. For example, the concentration of potassium ions inside a cell must be higher than the concentration of sodium ions. On the other hand, the outside of the cell is rich in sodium and poor in potassium. Ions tend to equalize their concentrations across the environment, which


Glycoprotein: protein with carbohydrate attached
Glycolipid: lipid with carbohydrate attached
Peripheral membrane protein
Integral membrane protein
Filaments of the cytoskeleton
Cholesterol
Protein channel
Phospholipid bilayer
Molecules of dye
Membrane (cross section)
WATER
Equilibrium
(a) Diffusion of one solute
Equilibrium
(b) Diffusion of two solutes
Copyright © Pearson Education, Inc., publishing as Benjamin Cummings.

(a) An illustration of a cell membrane. Adapted from [OpenStax College, 2013a].

(b) Ion diffusion. Adapted from [Dickinson, 2013].

Figure 5.1: Transmembrane ionic transport systems are cell structures present in the cell membrane (Figure 5.1a) which control the diffusion of ions (Figure 5.1b).

means that sodium ions tend to enter the cell, while potassium ions tend to leave it (Figure 5.1b). This is called an ion **concentration gradient**.

Transmembrane ionic transport systems control ion movement by opening and closing its doors, to allow or block the passage of ions, respectively. Ions commonly transported are sodium (Na^+), potassium (K^+) and calcium (Ca^{2+}). The difference in charge and concentration between ions in both sides creates an **electrochemical gradient**, which is essential for all animal cells to perform their functions properly (Figure 5.2). Ionic transport systems are responsible for the maintenance of this gradient, and participate in several biological processes, such as cellular volume control, synapses and nerve impulses, coordination of heart muscle contraction, transport of nutrients and release of accumulated calcium in the sarcoplasmic reticulum for performance of muscle contraction [Aidley and Stanfield, 1996].

There are two types of transport systems: ion channels and ionic pumps. Both systems are responsible for transporting ions, however, each one presents different characteristics and properties, described in further detail in the following subsections. These systems can be opened or blocked by toxins and drugs, which can compromise or improve the health of an individual by manipulating its physiological conditions. Some blockers and openers of transport systems are presented further below.

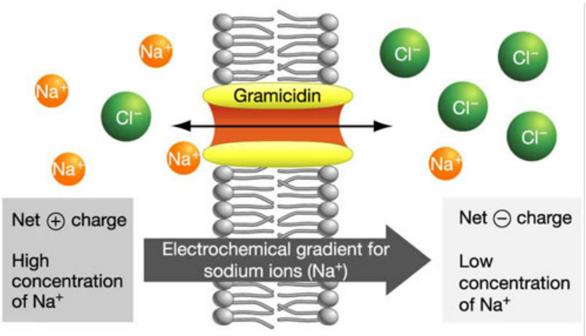
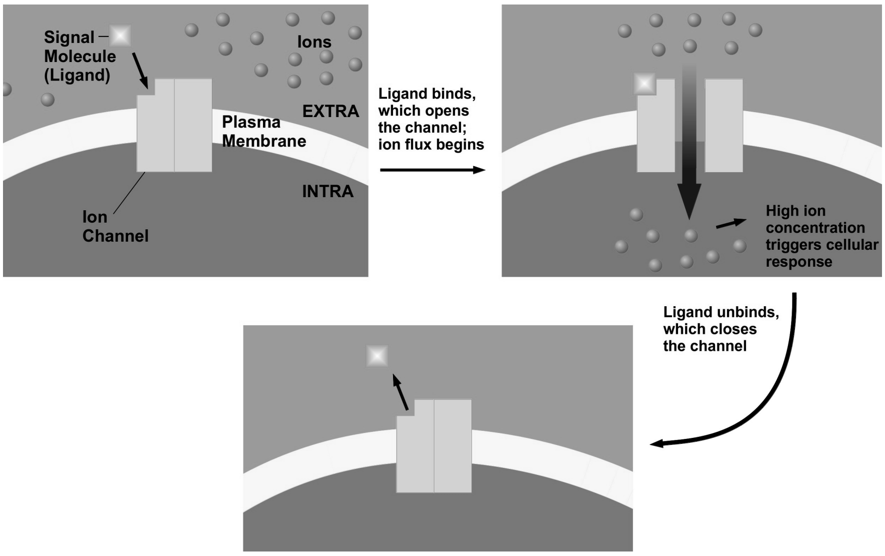


Figure 5.2: The difference of charges and concentrations between ions in both sides of the cell creates an electrochemical gradient, necessary for all animal cells to perform their functions properly. Adapted from [OpenStax College, 2013b].

5.2 Ion Channels

An ion channel is a passive transport system which does not spend energy to promote ion exchange. This means that ions move freely as long as the channel is open. The ions are transported according to their concentration gradient, moving from a high concentration environment to a low concentration one. Once open, ion channels rapidly diffuse ions, allowing abrupt changes in ion concentration. Different factors can open or close the channel, such as chemical or electric signals.



Signal Molecule (Ligand)
Ions
EXTRA
Plasma Membrane
INTRA
Ion Channel
Ligand binds, which opens the channel; ion flux begins
High ion concentration triggers cellular response
Ligand unbinds, which closes the channel

Figure 5.3: Generic Ion Channel. An open ion channel transfers ions down their electrochemical gradient. For example, if the extracellular concentration is higher than the intracellular one, the ions would rapidly move into the cell. A high concentration of ions inside the cell could trigger a cellular response, which closes the channel.

A generic ion channel is shown in Fig. 5.3. The ion channel is initially closed, and there is a high concentration of ions in the extracellular medium. A signaling molecule binds to the ion channel, which opens it. This allows the ions to diffuse rapidly from the extra to the intracellular side (low concentration of ions). The change in ion concentration inside the cell triggers a cellular response. The signaling molecule unbinds the ion channel, which closes it, therefore interrupting ion flux.

Ion channels are prone to several diseases and syndromes, called channelopathies, which disturb its functions and compromise the health of the individual. For example, hyper- and hypokalemic, where high and low potassium blood concentrations, respectively. Other examples include arrhythmias, cystic fibrosis, fibromyalgia, seizures, Timothy syndromes, and long and short QT syndrome [Dworakowska and Dolowy, 2000].

5.3 Ionic Pumps

An ionic pump is an active transport system that uses energy to perform ion exchange. This energy comes in different forms, such as Adenosine Triphosphate (ATP), the cell energy unit, and sunlight. Ionic pumps exchange ions against their concentration gradient, which means that ions are moving from their low concentration side to their high concentration side [Lehninger et al., 2008]. Ionic pumps exchange ions very slowly, permitting only subtle changes in ion concentration. For ionic pumps, the passage of ions can be viewed as two gates, one internal and another external, which open or close based on different factors, such as chemical signals [Aidley and Stanfield, 1996].

An example of a pump is the sodium-potassium pump or Na^+/K^+-ATPase (Fig. 5.4). This pump is responsible for exchanging three sodium ions from the intracellular medium (rich in potassium and poor in sodium) for two potassium ions from the extracellular medium (poor in potassium and rich in sodium).

This pump can be in two major states: open to the inside of the cell, or open to the outside. The pump cycle starts with three sodium ions binding to the pump when its open to the intracellular side. An ATP binds to the pump, which is followed by its hydrolysis. This breaks the ATP into two molecules, one of phosphate (P_i), which remains bound to the pump, and another of Adenosine Diphosphate (ADP), which is released inside the cell. This also causes the pump to release the sodium ions outside. Two potassium ions in the outside bind to the pump, which are released in the intracellular side, as well as the phosphate. The pump now can repeat the process [Aidley and Stanfield, 1996].

Other pumps include the calcium-pump or Ca^{2+}-ATPase, proton pumps,

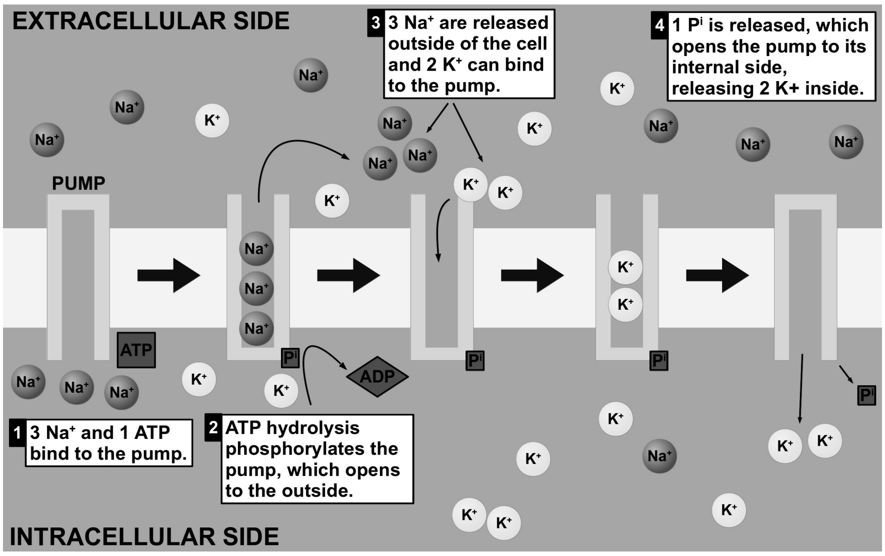

EXTRACELLULAR SIDE
3 3 Na$^+$ are released outside of the cell and 2 K$^+$ can bind to the pump.
4 1 P^i is released, which opens the pump to its internal side, releasing 2 K+ inside.
PUMP
Na$^+$
K$^+$
ATP
P^i
ADP
1 3 Na$^+$ and 1 ATP bind to the pump.
2 ATP hydrolysis phosphorylates the pump, which opens to the outside.
INTRACELLULAR SIDE

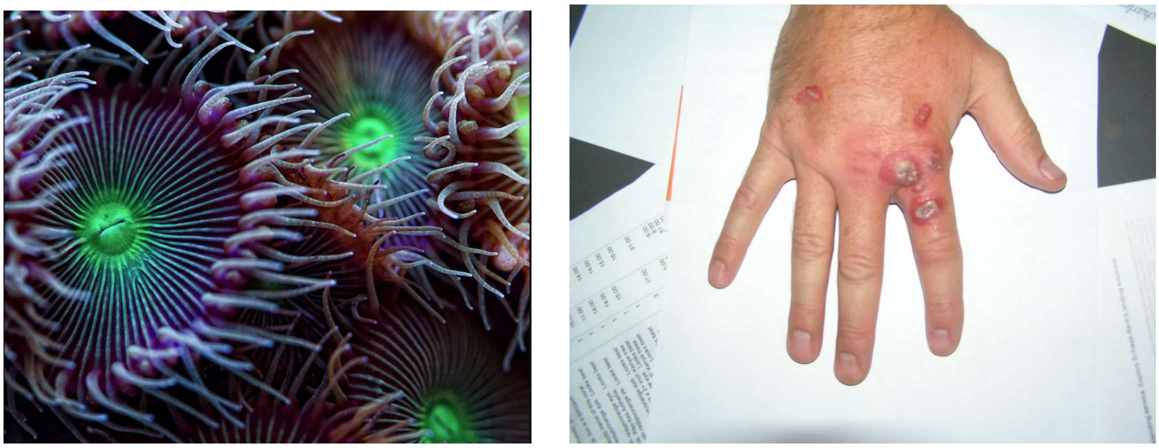
Figure 5.4: The sodium-potassium pump. Na$^+$/K$^+$-ATPase exchanges three sodium ions from the intracellular side of the cell for two potassium ions from the extracellular side. This is an active transport and it hydrolyzes a molecule of ATP to phosphorylate the pump and change its shape.

hydrogen-potassium-pumps or H$^+$/K$^+$-ATPase, and sodium-chloride-symporter or Na$^+$-Cl$^+$ cotransporter. Ionic pump channelopathies include seizures, ataxia, migraine, hyper- and hypokalemic, and hyper- and hyponatremia.

5.4 Blockers and Openers

Transmembrane ionic transport systems are one of the main targets in research for discovery and development of drugs, since they play a major role in several biological processes and their irregular behavior is associated with several diseases. Cardiac glycosides are one type of these drugs, for example digoxin (also known as digitalis) and ouabain, drugs that are used to improve heart performance by increasing its contraction force [Aidley and Stanfield, 1996]. These drugs bind to the transport system, either **blocking** it (closing its doors) and interrupting the passage of ions, or **opening** it and letting the ions move freely across the plasma membrane. This has to be done in a controlled manner, otherwise in order to treat one disease or syndrome, it would disrupt many other biological functions – the side-effects of drugs.

Toxins are another type of blockers and openers of transport systems. They operate in a similar manner to drugs, except that usually they only have negative effects and are not controlled, depending exclusively on the toxin concentration levels. Some examples of these toxins include Tetrodotoxin (TTX), used by puffer fish and some types of newts for defense, which blocks sodium channels; and Iberiotoxin, produced by the Buthus tamulus (Eastern Indian scorpion) and blocks potassium channels.



(a) Marine species *Palythoa toxica*. Adapted from [Johnny Vincent C., 2013].

(b) Palytoxin poisoning. Adapted from [WetWebMedia Forum, 2013].

Figure 5.5: One toxin which affects the sodium-potassium-pump is the Palytoxin (PTX) which is produced by the marine species *Palythoa toxica*. Its effects are opening the pump which lets ions move freely across the plasma membrane.

Due to their role in the nervous system, ion transport systems are affected by neurotoxins [Aidley and Stanfield, 1996]. One of the toxins that affects these structures is the Palytoxin (PTX), a deadly toxin found in corals of the *Palythoa toxica* species. PTX disturbs the Na^+/K^+-ATPase, modifying its behavior to the one of an ion channel, meaning that the pump transfers ions down their electrochemical gradient (from the ions high concentration side to their low concentration side), instead of against it [Artigas and Gadsby, 2004].

Typical symptoms of palytoxin poisoning are angina-like chest pains, asthma-like breathing difficulties, tachycardia, unstable blood pressure, hemolysis (destruction of red blood cells), and an electrocardiogram showing an exaggerated T wave. The onset of symptoms is rapid and death usually follows just minutes after [Artigas and Gadsby, 2002].

Even though transmembrane ionic transport systems have been discovered over 60 years ago, they still are an active research field [Aidley and Stanfield, 1996]. Because of their different behaviors and transfer rates, ion channels and ionic pumps have been seen as different entities. However, discoveries such as the interaction between PTX and the Na^+/K^+-ATPase are forcing new studies about the mechanics of these structures and its perception by the scientific community [Artigas and Gadsby, 2002, 2004].

Summary. In this chapter we have described transmembrane ionic transport systems, namely ion channels and ionic pumps, cell structures which are responsible for transporting ions between the inside and the outside of the cell. We have also covered diseases, syndromes and toxins, which block or open these systems and change their regular behavior and compromising the health of the cell.

Chapter 6

Related Work

Outline. In this chapter we present the related works to this dissertation. Due to its interdisciplinary nature (a computational approach and a biological case study), the related works have been divided into two groups, which reflects in the organization of this chapter.

The first group, **Analysis of Transmembrane Ionic Transport Systems** (Section 6.1), consists of experimental studies of transmembrane ionic transport systems, mostly represented by the works of Artigas and Gadsby, and Rodrigues and co-workers. They have investigated the interactions of the toxin palytoxin with the sodium-potassium pump, which essentially turns the pump behavior into the one of an ion channel, which disrupts several biological processes. These works have been studied in order to improve our understanding of pumps and ion channels from different perspectives, as well as knowing other ways of investigation of these cell structures.

The second group, **Modeling and Formal Analysis of Biological Systems** (Section 6.2), consists of works in the field of Formal Verification and Model Checking which have modeled and analyzed biological systems, not necessarily pumps and ion channels. It also includes useful formal techniques in the modeling and analysis of stochastic systems. These works have been studied in order to be aware of the currently used methodologies in the modeling, analysis and property specification of biological systems using formal methods.

Each of these techniques take place in different contexts, which can be experimental (*in vivo* or *in vitro*) or computational (*in silico*), briefly defined below.

In Vivo: model is inside or in the tissue of a living organism.

In Vitro: model is outside of a living organism.

In Silico: model is represented computationally using simulations and other methods.

6.1 Analysis of Transmembrane Ionic Transport Systems

The authors of [Artigas and Gadsby, 2002; Artigas, 2003b,a; Artigas and Gadsby, 2004] studied a deadly toxin produced by the marine species *Palythoa toxica* called Palytoxin (PTX) and investigated its interactions with the sodium-potassium-pump (Na^+/K^+-ATPase). They have performed several experimental studies using the technique called *Patch-Clamp*, which allows measuring the induced current of ionic pumps. This allowed them to observe the effects of PTX in the induced current. They have discovered that the toxin drastically modifies the nature of the pump, changing its behavior (active transport) to the one of an ion channel (passive transport). This means that ions are moving freely down their electrochemical gradients, instead of being controlled by the pump, which disrupts several biological processes. The authors suggest that PTX, as well as other toxins, could be useful tools in future experiments to discover the control mechanisms for opening and closing the gates of ion pumps.

The effects of PTX on the sodium-potassium-pump is later studied by the authors of [Rodrigues et al., 2008b], this time through mathematical simulations using non-linear ordinary differential equations. In this first work, they have considered only states and reactions related to the phosphorylation process (phosphate binding and unbinding to the pump). This line of research produced other contributions considering different states and reactions. One of these contributions is [Rodrigues et al., 2008a], which considers interactions that are not related to the phosphorylation, in a way complementary to its previous work. Another contribution is [Rodrigues et al., 2009a], which considers only interactions with potassium ions (K^+) in order to understand its role, since potassium has been previously indicated as a physiological inhibitor of PTX.

Rodrigues and co-workers obtained different and reduced models, each one analyzing one aspect of PTX interactions with the sodium-potassium pump cycle, also known as the Albers-Post model [Post et al., 1965, 1972]. Interactions of PTX with the complete model of the Na^+/K^+-ATPase are presented in [Rodrigues et al., 2009b]. This series of studies by Rodrigues and co-workers can be viewed as a simulational approach of the experimental results of Artigas and co-workers. This dissertation can be seen as a probabilistic model checking approach to the same experiments.

The sodium-potassium pump specific to cardiac cells is examined in [Oka et al., 2010] using different types of models based on ordinary differential equations. Initially, a model composed of thirteen states is presented. However, a reduction to a four-state model is obtained, while mantaining the same behavior despite it being simpler. This work also demonstrates the central role that the pump plays in maintaining the calcium ion concentration levels, essential to the heart muscle contraction. Additionally, it presents a model for the pump coupled with cesium, a marker substance used to perform experiments, which interferes in the behavior of the pump similarly to toxins and drugs.

Channels and pumps usually are investigated using experimental results in laboratory benches, which are expensive for both financial and time resources. In order to avoid or minimize these costs, different types of simulations, mathematical and computational methods are employed, among these include sets of ordinary differential equations (ODEs) and Gillespie's algorithm for stochastic simulations [Gillespie, 1977]. Despite their ability to obtain valuable information, simulations do not cover every possible situation, and might never search certain regions of the state space, therefore possible overlooking some events, such as ion depletion, where all ions have been exchanged. These systems are also described as a kinetic model, which presents its states and possible reactions from one state to other states (for example, Figure 7.8).

6.2 Modeling and Formal Analysis of Biological Systems

The main tools used in the formal verification of biological systems are PRISM [Kwiatkowska et al., 2011], BioLab [Clarke et al., 2008], Ymer [Younes, 2005], Bio-PEPA [Ciocchetta and Hillston, 2009] and BIOCHAM [Fages and Soliman, 2008].

PRISM is a probabilistic model checker, which allows modeling and analyzing probabilistic systems. It exhaustively and automatically explores a formal model checking if it respects a set of properties. PRISM supports different types of models, properties and simulators [Kwiatkowska et al., 2011]. It has been largely used in distinct fields, e.g. communication and media protocols, security and power management systems. Due to its probabilistic nature, it switches its internal representation of the models from the traditional binary decision diagrams (BDDs) of symbolic model checking to multi terminal binary decision diagrams (MTBDDs), thus allowing multiple terminal states and non-deterministic transitions [Bryant, 1986; Fujita et al., 1997].

Among its many supported models, there are continuous-time Markov chains (CTMCs). These models are computationally efficient because they present the *Markov property* of stochastic processes, which means that evaluation of future states depend only on the current state, instead of all previous states (or chain history), which would be otherwise computationally expensive [Kwiatkowska et al., 2011].

We have used PRISM in this dissertation due to several reasons, which include:

exact PMC in order to obtain accurate results; Continuous-time Markov Chain (CTMC) models, suited for our field of study; rich modeling language that allowed us to build our model; and property specification using Continuous Stochastic Logic (CSL), which is able to express qualitative and quantitative properties.

The authors of [Clarke et al., 2008] introduce a new algorithm called BioLab. Instead of building all states of a model, the algorithm generates the minimum number of necessary simulations, given error bounds parameterized for acceptance of false positives and false negatives of the properties to be verified. This algorithm is based on the work of [Younes, 2005], author of the approximate model checker Ymer.

In [Younes et al., 2006], the authors compare numerical and statistical methods for PMC, since exact model checking is not always possible due to timewise and computational resources restrictions. Approximate model checking is an alternative when it is acceptable to lose precision in order to obtain approximate results that are obtained in a timely manner. The model checker Ymer uses this technique [Younes, 2005].

A novel computational modeling approach using formal verification with the PRISM model checker is presented in [Calder et al., 2010]. The authors treat their model – signaling pathways – as a distributed system, where its components can interact with each other similarly to computer processes. Rather than using an individual approach where each of the ligands are modeled individually, they treat these components as populations, in order to capture the behavior of a whole set of ligands.

The Biochemical Abstract Machine (BIOCHAM) is an environment for the modeling and verification of biochemical systems [Fages and Soliman, 2008]. It supports its own modeling language as well as other formats such as *Systems Biology Markup Language* (SBML) and *Systems Biology Graphical Notation* (SBGN). BIOCHAM includes several simulators, such as: boolean, which allows observing the presence or absence of molecules; deterministic, which allows observing molecular concentrations; and stochastic, which allows quantifying molecules [Fages, 2006]. Property specification is performed using several logics, including *Computation Tree Logic** (CTL*), *Computation Tree Logic* (CTL), *Linear Temporal Logic* (LTL), *Probabilistic Computation Tree Logic* (PCTL) and *LTL with Constraints Over Reals* (Constraint-LTL).

BIOCHAM is also capable of deducing the values of kinetic parameters (rates of transitions) from a set of restrictions, as well as deducing logical rules from a set of logical properties [Calzone et al., 2006; Gay et al., 2010]. In [Batt et al., 2010], the authors also demonstrate the deduction of model parameters, however they must satisfy real data instead of a set logical properties.

The logics used by these tools are frequently extended, such as the work of [Mateescu et al., 2011], which presents an extension to CTL named *Computation Tree Logic with Regular Expressions* (CTRL). This new logic has the objective of supporting regular expressions in the property specification. In [Monteiro et al., 2008], the authors present a set of common property patterns frequently used in system specification.

The authors of [Monteiro et al., 2009] created and implemented a client-server architecture with task delegation for different computational nodes in order to perform formal verification of gene regulatory networks. The presented infrastructure is sufficiently robust to deal with different model checkers, using server plugins. However, depending on the model, parameters and properties to be checked, the verification can take a considerable amount of time and rapidly occupy the nodes.

The application of PMC to model and analyze different complex biological systems can be seen in [Heath et al., 2008; Kwiatkowska et al., 2010], for example the signaling pathway of Fibroblast Growth Factor (FGF), a family of growth factors involved in healing and embryonic development. The analysis of other signaling pathways such as MAPK and Delta/Notch can be seen in [Kwiatkowska et al., 2009].

The use of PMC is demonstrated also in [Kwiatkowska et al., 2008; Kwiatkowska and Heath, 2009], where the authors examine and obtain a better understanding of mitogen-activated kinase cascades (MAPK cascades) dynamics, biological systems that respond to several extracellular stimuli, e.g. osmotic stress and heat shock, and regulate many cellular activities, such as mitosis and genetic expression.

Several researchers presented in [Sauro et al., 2006] some challenges for systems biology, such as difficulties to integrate communities so different such as computer science and biology, building a computational model of a multi-cellular organism and using compositional reasoning on parallel systems such as cell signalling pathways.

Usually each tool has its own modeling language, which can be cumbersome for someone using multiple tools. Some of them share similar notations which allows faster modeling, such as the case of PRISM and MRMC, however often that is not the case. Nonetheless, the Systems Biology Markup Language (SBML) [SBML, 2013] and Cell Modeling Language (CellML) [CellML, 2013] are XML based markup languages for describing models, and they are supported across several tools. There are other languages as well, such as BioPAX [BioPAX, 2013]. Also, there are some online repositories of biological models, such as the CellML [CellML Models, 2013] one, Reactome [Reactome, 2013] and BioModels [BioModels, 2013].

Summary. In this chapter we have presented the related works of this dissertation, regarding both experimental and computational studies of transmembrane ionic transport systems [Artigas and Gadsby, 2002]. Computational approaches can be cost efficient alternatives to laboratory experiments, for example, using simulations of ordinary differential equations to represent the behavior of the sodium-potassium-pump coupled with the toxin palytoxin [Rodrigues et al., 2008b]. We have also covered formal approaches to study biological systems in general [Kwiatkowska et al., 2008].

Chapter 7

Formal Analysis of Transmembrane Ionic Transport Systems

Outline. In this chapter we present the formal modeling and analysis of one of our electrophysiology models using probabilistic model checking. Our case study is the sodium-potassium pump and its interactions with the toxin palytoxin (PTX), which opens the pump and compromises its regular behavior.

We have built four models, following the works of Rodrigues and co-workerss [Rodrigues et al., 2008b] and Artigas and Gadsby [Artigas and Gadsby, 2002], where they have studied the effects of PTX on the sodium-potassium-pump. One could view the pump as a base model, and the palytoxin interactions as its extension, therefore we have divided this chapter following that view.

	Present ligands			References	
Model name	Sodium	Potassium	ATP	Experimental	Simulational
`ptx2007`			✔	Artigas and Gadsby [2002]	Rodrigues et al. [2008b]
`ptx2008a`	✔	✔		Artigas [2003b]	Rodrigues et al. [2008a]
`ptx2008b`		✔		Artigas [2003a]	Rodrigues et al. [2009a]
`ptx2009`	✔	✔	✔	Artigas and Gadsby [2004]	Rodrigues et al. [2009b]

Table 7.1: We have built four models, following the works of Artigas and Gadsby and Rodrigues and co-workers. Each model focuses in one aspect, such as cell energy reactions or the potassium inhibitory effect on PTX action.

Our models are called `ptx2007` [Rodrigues et al., 2008b; Braz, 2012d], `ptx2008a` [Rodrigues et al., 2008a; Braz, 2012e], `ptx2008b` [Rodrigues et al., 2009a; Braz, 2013a] and `ptx2009` [Rodrigues et al., 2009b; Braz, 2013b], each one focused on a different aspect of the model and associated with the publication in parenthesis, respectively. Our models have been built using the PRISM model checker, and have their characteristics summarized in Table 7.1.

The first model (`ptx2007`) focuses only on the role of cell energy (Adenosine Triphosphate or ATP), which our model suggests that high concentrations of ATP could inhibit PTX action. The second model (`ptx2008a`) focuses on the role of sodium and potassium ions, excluding ATP related states and reactions. This model suggests that sodium enhances PTX action and potassium inhibits it.

The third model (`ptx2008b`) focuses on the role of potassium ions, excluding everything else (sodium and ATP), since potassium has been indicated as a natural inhibitor of PTX action. We have also included in our model the GHK (Goldman-Hodgkin-Katz) flux equation, which allows measuring the induced current of ions.

The fourth and final model (`ptx2009`) is the complete model, and includes all states, reactions and ligands. This model is considerably more complex than the other ones and unfortunately we have not been able to perform thoroughly experiments on it. However, it confirmed some of our previous results and it remains as a future challenge as we intend to use approximate analysis techniques to handle large models.

Finally, we have used only our first model as an example of our formal modeling, otherwise this chapter would be repetitive and impractical. This is because the models are similar in their modeling, although different in their semantics.

7.1 Formal Analysis of the Sodium-Potassium Pump

The models have been written in the PRISM modeling language (used by the PRISM model checker [Kwiatkowska et al., 2011]). The Figure 7.1 below is a high level diagram which describes the structure of our models.

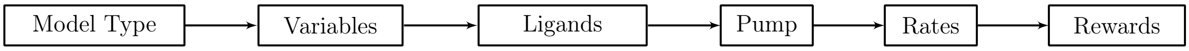


Figure 7.1: The structure of our models.

The PRISM modeling language is explained in Chapter 4. First of all, the model type is a continuous-time Markov chain, which is indicated by the `ctmc` keyword. This is followed by the declaration of global variables (Figure 7.2), explained below in detail.

```
ctmc

const double exp;

const double AV=6.022*pow(10.0,23);
const double V=pow(10.0,-exp);

label "ptxAllBounded" = ptxOut=0;

formula pK = gamma01 * PTXE + gamma02 * PTXATPhighE;
```

Figure 7.2: Model type and declaration of variables.

These variables include useful definitions across the model, such as the Avogadro's constant (the constant `AV`) which is used alongside the pump volume to obtain the number of ligand molecules. There are also parameters, such as the size of the pump volume (the variable `exp`), which allows the parametric study of our models. Labels are used to create meaningful events, such as the PTX depletion (the label `ptxAllBounded`), where the condition of PTX molecules being equal to zero must be met for it to be true. Formulas are used to quantify the model in function of other variables, such as the permeability of potassium ions (the formula `pK`).

The model is divided in several PRISM modules, most of them being ligand modules (the molecules and ions). A ligand module contains a variable to store the current number of ligands, e.g. `atpIn` for Adenosine Triphosphate (ATP), or `naOut` for external sodium ($[Na^+]^o$). These modules are also responsible for controlling their ligand consumption and allowed reactions (Figure 7.3). The other ligands are Adenosine Disphosphate (ADP), Phosphate (P_i), Sodium (Na^+) and Potassium (K^+). The Palytoxin (PTX) extensions include one additional module to represent the toxin.

```
module ptx
  ptxOut : [0..(PTX0)] init PTX0; // number of PTX outside the cell
  // reaction p1: PTXo + E1 <-> PTXE
  [rp1]  ptxOut>=1 -> ptxOut : (ptxOut'=ptxOut-1);
  [rrp1] ptxOut<=(PTX0-1) -> 1 : (ptxOut'=ptxOut+1);
endmodule
```

Figure 7.3: A ligand module.

Each module is composed of PRISM commands (or transitions) that represent reactions, which are responsible for changing the number of molecules. A PRISM command uses the following structure:

```
[sync] conditions → rate :  update
```

where the `conditions` (also known as the *guard* part) must be observed for the `update` to occur at a given `rate` (or probability). The `sync` (also known as label) is used to synchronize multiple commands.

Command conditions usually are lower and upper bounds, i.e. there must be at least one molecule for a binding reaction. The list of reactions can be found in the comments of our model [Braz, 2012d], and in its associated publication (for example, `ptx2007` is [Rodrigues et al., 2008b], see Table 7.1).

```
module pump
  E1 : [0..1] init 1;  // e1 conformational state
  E2 : [0..1] init 0;  // e2 conformational state

  // reaction 1: E2 <-> E1
  [r1]  E2=1 & E1=0 -> 1 : (E2’=0) & (E1’=1);
  [rr1] E2=0 & E1=1 -> 1 : (E2’=1) & (E1’=0);
endmodule
```

Figure 7.4: The pump module.

There are also two special modules, one being the pump, which stores and controls its state by manipulating boolean vectors (Figure 7.4). For example, there are two states of the pump called `E1` and `E2`, which represent the pump open to the internal and external sides of the cell, respectively. The Table 7.2 lists the states of the pump.

	Open		ATP Binding Site		
State name	Intra	Extra	High	Low	P_i
E1	✔				
ATPhighE1	✔		✔		
ATPlowPE1	✔			✔	✔
E2		✔			
PE2		✔			✔
ATPlowPE2		✔		✔	✔
ATPlowE2		✔		✔	

Table 7.2: Albers-Post states characteristics. Albers-Post model states which correspond to the pump cycle. The pump can be open to the intracellular side (2nd column) and open to the extracellular side (3rd column). An ATP can bind to the pump in either its high or low affinity binding sites (4th and 5th columns). The pump can be phosphorylated (6th column).

The other special module is the rates module, which defines the rate or speed of each reaction by binding the reaction rate constant with its associated reaction label (Figure 7.5). For example, the reaction `r1`, which is the pump closing to the external side and opening to the internal side. Its reaction rate is defined by the constant `r1rate`. A PRISM command is used to bind the `r1` label to the `r1rate`.

```
const double r1rate  = 1.00*pow(10,2);
const double rr1rate = 0.01;

module base_rates
  [r1]  true -> r1rate : true;
  [rr1] true ->rr1rate : true;
endmodule
```

Figure 7.5: The rates module.

The model ends with the rewards definitions, which allows the quantification of different aspects of the model, such as states and transitions (Figure 7.6). For example, the state `e1` is quantified by its homonym reward. In order to count towards the reward, its conditions must be met. Transition rewards are slightly different – they are bound to their labels and count only when all commands associated with that label are allowed to execute. Finally, the time reward is always counted.

```
rewards "e1"
  (E1=1) : 1;
endrewards

rewards "r1rate"
  [r1] true: 1;
endrewards

rewards "time"
  true: 1;
endrewards
```

Figure 7.6: The rewards definitions.

A fragment of the `ptx2007` model is presented in Figure 7.7. It does not include PTX because its interactions with the pump are an extension presented in the code fragment of Figure 7.9. The complete version of this model can be viewed online [Braz, 2012d].

The Albers-Post model [Post et al., 1972] represents the Na^+/K^+-ATPase cycle and it can be seen on the left side of Figure 7.8. According to it, the pump can be in different sub-states, which change depending on different reactions involving ATP, ADP and P_i. The right side shows the Palytoxin extension model, which is discussed later. The Table 7.2 summarizes the states of the pump and their characteristics.

The pump can be open or closed to the extra and intracellular sides. An ATP molecule can bind to the pump in its high or low affinity binding sites. An ATP molecule bound to the pump can be hydrolyzed, leaving one P_i molecule bound to the pump and releasing one ADP molecule. The reactions are bidirectional and their rates were obtained from [Rodrigues et al., 2008b].

Na^+/K^+-ATPase PRISM Model

```
module atp
 // number of ATP inside cell
 atpIn : [0..(N)] init ATPI;
 // reaction1: ATPi + E1 <-> ATPhighE1
 [r1] atpIn>=1 -> atpIn : (atpIn'=atpIn-1);
 [rr1] atpIn<=(N-1) -> 1 : (atpIn'=atpIn+1);
endmodule

module pump
 // e1 state: pump open to the intracellular side
 E1 : [0..1] init 1;
 // e1 with atp bound to its high affinity site
 ATPhighE1 : [0..1] init 0;
 //reaction1: ATPi + E1 <-> ATPhighE1
 [r1] E1=1 & ATPhighE1=0 -> 1 : (E1'=0)&(ATPhighE1'=1);
 [rr1] E1=0 & ATPhighE1=1 -> 1 : (E1'=1)&(ATPhighE1'=0);
endmodule

// base rates
const double r1rate = 1.50*pow(10,4)/(0.001*V*AV);
const double rr1rate = 1.64;
// module representing the base rates of reactions
module base_rates
 [r1] true -> r1rate : true;
 [rr1] true ->rr1rate : true;
endmodule
```

Figure 7.7: The Na^+/K^+-ATPase model. The model is a probabilistic transition system composed of modules for the molecules species (for example, ATP, ADP and P_i), which control their flow; the pump, which controls the pump sub-state and the base rates, which define the speed of the reactions.

7.2 Formal Analysis of Palytoxin Interactions with the Pump

The palytoxin model is an extension of the Na^+/K^+-ATPase model, previously described. It corresponds to the right side of the Figure 7.8, and it is based on the description of [Rodrigues et al., 2008b] and [Artigas and Gadsby, 2002]. Once again, a fragment of the model extension is shown in Figure 7.9, and its complete version can be seen online[1].

This extension consists of: one additional molecule module (PTX) which controls its flow; additional reactions in each of the already present modules; and additional sub-states and transitions for the pump module. Initial concentrations for $[PTX]^o$ and stochastic rates for reactions were obtained from [Rodrigues et al., 2008b].

The six additional sub-states correspond to the pump bound to PTX, when the pump is open to both sides behaving like an ion channel. The Table 7.3 summarizes the PTX states and their characteristics.

[1] ptx2007 model. `http://dcc.ufmg.br/~fbraz/ptx2007/`. Access date: October 31, 2013

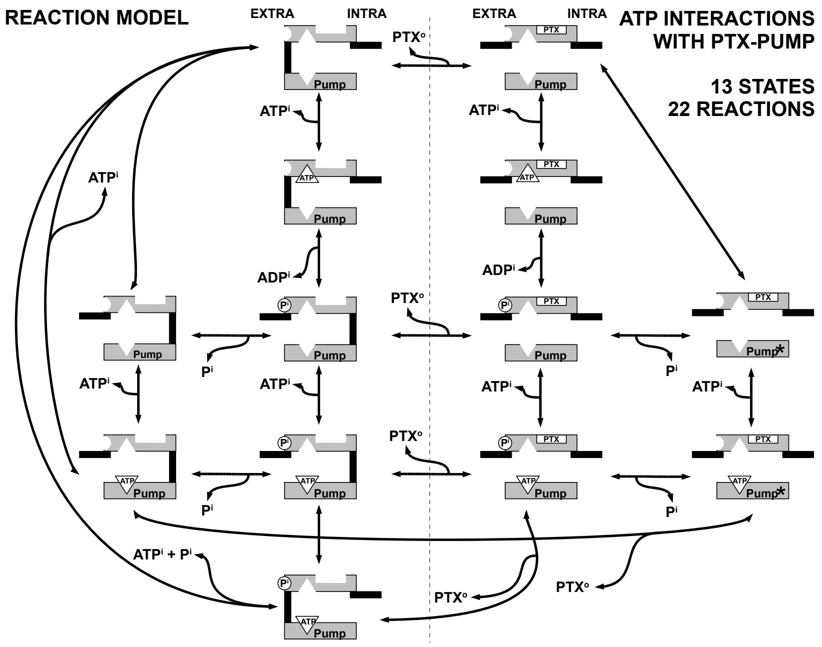


Figure 7.8: The kinetic model for the interactions of PTX with the pump. It describes all the sub states (13) and reactions (22). The left side is the classical Albers-Post model [Post et al., 1972], which describes the regular behavior of the pump, while the right side describes the PTX related states and reactions [Rodrigues et al., 2008b].

Palytoxin PRISM Model

```
module ptx
  // number of PTX outside the cell
  ptxOut : [0..(PTXO)] init PTXO;
  //reaction p1: PTXo + E1 <-> PTXE
  [rp1] ptxOut>=1 -> ptxOut : (ptxOut'=ptxOut-1);
  [rrp1] ptxOut<=(PTXO-1) -> 1 : (ptxOut'=ptxOut+1);
endmodule

module pump
  PTXE : [0..1] init 0;
  //reaction p1: PTXo + E1 <-> PTXE
  [rp1] E1=1 & PTXE=0 -> 1 : (E1'=0) & (PTXE'=1);
  [rrp1] E1=0 & PTXE=1 -> 1 : (E1'=1) & (PTXE'=0);
endmodule

// base rates
const double rp1rate=2.73*pow(10,1)/(0.001*V*AV);
const double rrp1rate=6.0*0.0001;
// module representing the base rates of reactions
module base_rates
  [rp1] true -> rp1rate : true;
  [rrp1] true -> rrp1rate : true;
endmodule
```

Figure 7.9: The palytoxin (PTX) extension of the model requires a new species module to control the number of PTX molecules and several new sub-states and transitions for the pump. Those are created to represent the interactions of PTX with the pump.

	ATP Binding Site			
State name	High	Low	P_i	After P_i
PTXE				
PTXATPhighE	✔			
PTXPE			✔	
PTXATPlowPE		✔	✔	
PTXATPlowE*		✔		✔
PTXE*				✔

Table 7.3: PTX related states characteristics. Sub-states which correspond to the pump bound to PTX, when the pump is open to both sides behaving like an ion channel. An ATP can bind to the pump in either its high or low affinity binding sites (2nd and 3rd columns). The pump can be phosphorylated (4th column). There is a distinction between some states when the pump has been dephosphorylated (5th column).

Summary. In this chapter we have presented the formal modeling and analysis of one of our electrophysiology models. Our case study is the sodium-potassium pump and its interactions with the toxin palytoxin, which opens the pump and disrupts several biological processes.

We have built four different models, following the experimental results of Artigas and Gadsby and simulational results of Rodrigues and co-workers. Each model explores a different aspect of our case study, such as cell energy related reactions and the potassium inhibitory effect on palytoxin action. Our models have been built using the PRISM model checker.

Chapter 8

Discussion and Results

Outline. In this chapter we present the results that we have obtained in this dissertation. We have built four formal models of the toxin palytoxin (PTX) interactions with the sodium-potassium-pump using a probabilistic model checking approach. This toxin disrupts the regular behavior of the pump, compromising several biological processes.

The models are based on different publications of Rodrigues and co-workers, each one focusing on a aspect of the system. For example, our first model covers the role of cell energy, while the second model focuses on the sodium and potassium reactions.

After we have built the models using the PRISM model checker, we have decided to explore their dimensions by performing a parametric study. This consisted of increasing or decreasing ligand concentrations (Section 8.1), which has allowed the investigation of extreme conditions, such as diseases which increase the sodium concentration or fatal exposures to PTX. We have used quantitatives PRISM properties to quantify every aspect of the model, such as state and reaction probabilities.

Each model suggested a different behavior, such as that high concentrations of ATP inhibit PTX action (Section 8.2), sodium enhances PTX (Section 8.3), and potassium inhibits PTX (Section 8.4). We have also quantified the induced current created by ions exchange through the Goldman-Hodgkin-Katz (GHK) flux equation (Section 8.5). Since we have every aspect of the model quantified, we were able to enhance the kinetic model with probabilities, thus creating a heat map (Section 8.6). Finally, we briefly discuss on the matter of alternative models, which might be more appropriate for future works since they are more similar to the currently available experimental procedures (Section 8.7).

8.1 Model Complexity and Parametric Study

We can explore our models by changing their parameters or dimensions[1]. Depending on the model, these dimensions are: $[PTX]^o$ (extracellular PTX concentration), $[ATP]^i$ (intracellular ATP concentration), $[Na^+]^o$ (extracellular sodium concentration), $[K^+]^o$ (extracellular potassium concentration) and pump volume. Each dimension represents a different aspect of the model, and can be changed to modify its behavior.

The values of these parameters influence directly to the complexity of the model representation (number of states, transitions and topology), and the time to build and verify model properties. For example, consider our first model with the parameters $[PTX]^o = 0.001\ \mu M$, $[ATP]^i = 10\ mM$ and pump volume of 10^{-22} L. In these conditions, the model representation has 376 states and 1912 transitions, taking 0.004 s to build and 310.895 s to check a state reward property discussed further below.

Pump Volume	States	Transitions	Check Time
10^{-22}	376	1912	310.895 s
10^{-21}	1274	7140	321.506 s

Table 8.1: Model complexity as function of pump volume.

The Table 8.1 illustrates how these values increase in function of pump volume, for our first model with $[PTX]^o = 0.001\ \mu M$, $[ATP]^i = 10\ mM$ and a state reward property (discussed further below). The machine used to perform the experiments is an Intel(R) Xeon(R) CPU X3323, 2.50GHz with 16 GB of memory RAM.

The volume of an animal cell is 10^{-12} L [Hernández and Chifflet, 2000], which is challenging to be represented using PMC. Otherwise, it would cause the classical problem of state space explosion. Our analysis is restricted to only one pump. Consequently, it would be unrealistic to model a large volume because in a real cell the volume is shared between several pumps and other types of structures.

Our abstraction reduces the cell volume, concentrating our analysis in one pump and its surroundings. We have achieved this by maintaining the proportions between all interacting components. Therefore, the cell volume is called pump volume and it is usually 10^{-22} L. Even though those values are many orders of magnitude smaller than the real ones, they still represent proper cell behavior, and can be interpreted as using a magnifying glass to investigate a portion of the cell membrane.

[1] A ligand name in brackets with a superscript o, for example $[S]^o$, indicates the external concentration of ligand S. On the other hand, a superscript i, for example $[S]^i$, indicates the internal concentration of ligand S.

On the other hand, for some dimensions we have used more values than the literature reference, ranging three orders of magnitude below and above their regular values. For example, for $[\text{PTX}]^o$ and $[\text{ATP}]^i$, we have used between 1 and 10 μM for $[\text{PTX}]^o$ and 1 and 10 mM for $[\text{ATP}]^i$. This allows the simulation of different conditions for pump behavior, including abnormal concentrations of $[\text{ATP}]^i$ due to a disease or syndrome, and different degrees of exposure to $[\text{PTX}]^o$, from mild to fatal exposure.

Each configuration of model parameters is called a scenario. For example, the regular physiological conditions is the Control scenario, while an increased concentration of $[\text{ATP}]^i$, usually ten times greater, is the High $[\text{ATP}]^i$ scenario. Each model has a different set of scenarios, depending on the ligands that are present.

8.2 ATP Interactions with the PTX-Pump Complex

The results in this section have been obtained using our first model (`ptx2007`), which focuses on the role of cell energy reactions [Rodrigues et al., 2008b; Braz et al., 2012b].

We first decided to obtain the probability of PTX inhibiting the pump, i.e., the pump being in PTX-related states. In order to do that, we quantified the probability of every state, either PTX or non-PTX related. All states were labeled and quantified using rewards. Basically, rewards are incremented each time their conditions are true. In this case, they are being used to count how many times the pump is in a particular state. The fragment of the model in Figure 8.1 shows the reward of the state `PTXE`, where the pump is bound to PTX and open to both sides of the cell.

State Rewards

```
rewards "ptxe"
  (PTXE=1) : 1;
endrewards
```

Accumulated State Reward Property

$\mathcal{R}\{\text{“ptxe”}\}_{=?}$ [$\mathbf{C} \leq \text{T}$]

What is the expected accumulated reward of the state `ptxe` at time T?

Figure 8.1: Rewards are used to quantify aspects of the model, such as states (for example, `ptxe`) and reactions. The accumulated reward property is used to obtain the expected accumulated (**C** operator) value of a particular reward ($\mathcal{R}$ operator) at a given time (T variable).

A model with rewards for each state can be completely quantified, and allows, for example, counting the expected accumulated reward associated with each state over time. In order to do that, we have used properties such as the one shown in Figure 8.1. The operator $\mathcal{R}$ allows the quantification a given reward, for example the number of times that the model was in the state PTXE. The operator **C** accumulates the reward value until a given time T, therefore we are able to observe the reward over time.

Consider the following conditions: a single pump, a pump volume of 10^{-22} L, $[\text{ATP}]^i = 10$ mM and $[\text{PTX}]^o = 10$ µM, at instant T=100s. The expected accumulated rewards for the same sub-state PTXE and the sub-state PE2, where the pump is open to the external side and bound to a phosphate, are respectively 1.0100 and 33.3544. The state probability can be obtained by dividing the state reward by the sum of all state rewards. In other words, in 100 seconds, the pump is expected to be open to the extracellular side and phosphorylated approximately 34.1952% of the time, and the pump is expected to be bound exclusively to PTX only 1.0355% of the time.

Table 8.2: PTX-pump state probability for different scenarios.

Pump State	Control	High $[\text{ATP}]^i$	Difference
E1	0.0072%	0.0045%	-37.7249%
ATPhighE1	0.0312%	0.0185%	-40.5495%
ATPlowPE1	0.0314%	0.0187%	-40.5245%
E2	34.1902%	39.2551%	+14.8137%
PE2	34.1952%	39.2389%	+14.7496%
ATPlowPE2	7.3272%	6.5347%	-10.8154%
ATPlowE2	0.0000%	0.0000%	+2.4527%
Non-PTX related	75.7824%	85.0704%	+12.2560%
PTXE	1.0355%	0.6201%	-40.1175%
PTXATPhighE	3.4466%	2.3806%	-30.9279%
PTXPE	8.3250%	5.8626%	-29.5785%
PTXATPlowPE	0.0308%	0.0177%	-42.4199%
PTXATPlowE*	0.7262%	0.1969%	-72.8886%
PTXE*	10.6535%	5.8517%	-45.0723%
PTX related	24.2176%	14.9296%	-38.3520%

Probabilities for states of the sodium-potassium pump interacting with palytoxin at time T=100s. Two states are more present than others: E2, where the pump is open to its external side; and PE2, that same state although the pump is phosphorylated. As [ATP]^i increases, the probability of PTX related states decreases, which suggests that ATP inhibits PTX action.

Using a broad spectrum of different $[ATP]^i$ and $[PTX]^o$, for the pump volume of 10^{-22} L, we have found that there are only two sets of values for sub-state rewards. One set is associated with $[ATP]^i$ equals or below to 10 mM, while the other set is associated with $[ATP]^i$ above 10 mM.

For example, when $[ATP]^i = 100$ mM, the expected rewards associated with the two sub-states `PE2` and `PTXE` change to respectively 37.3577 and 0.5904, or 39.2389% and 0.6201% of the time. Therefore, as we increased $[ATP]^i$, the likelihood of the pump being open to the extracellular side and phosphorylated increased 14.7496%, and for the pump to be bound exclusively to PTX decreased 40.1175%.

Species Depletion Events and Time Reward

```
label "ptxAllBounded" = ptxOut=0;

rewards "time"
  true: 1;
endrewards
```

Species Depletion Properties

$\mathcal{P}_{>=1}$ [**F** “ptxAllBounded”]
The event “ptxAllBounded” eventually always happens.

$\mathcal{R}\{$“time”$\}_{=?}$ [**F** “ptxAllBounded”]
How long it takes for the event “ptxAllBounded” to happen?

Figure 8.2: The depletion of a species (ATP and PTX) happens when the variable which stores its number reaches zero. These events can be observed using labels, one of the features of PRISM. One can check if a given event happens using reachability properties (**F** operator). A time reward allows quantifying when an event occurs.

The rewards can be divided in two groups: PTX related sub-states and Albers-Post (non-PTX) sub-states. Summing all the rewards of each group, and dividing each by the total, one can obtain the probability of the pump being inhibited by PTX. For $[ATP]^i = 10$ mM and T=100s, PTX related states correspond to 24.2176%. As we increased $[ATP]^i$ to 100 mM, PTX related states correspond to only 14.9296%, suffering a 38.3520% reduction.

Therefore, this reduction suggests that as $[ATP]^i$ increases, the probability of being in PTX related sub-states decreases, suggesting that ATP is an inhibitor of PTX. As consequence, people with ATP depletion would be more vulnerable to this toxin. ATP deficiency appears in different forms, e.g. brain disorders, for example, stroke and encephalopathies [Yamada and Inagaki, 2002]. The Table 8.2 summarizes the state reward values and percentages for the Control and High $[ATP]^i$ scenarios, as well as the difference between them.

Similar reward structures to the ones of Figure 8.1 were created for transitions (in our model, chemical reactions). For $[ATP]^i = 10$ mM we found that during the first

100 seconds PTX related reactions correspond only to approximately 8.01%. Once we change $[\mathrm{ATP}]^i$ to 100 mM the role of PTX related reactions decreases by approximately 42.97%, which reinforces our discovery that high doses of ATP inhibit PTX action. The most active reactions are dephosphorylation, changes in the pump conformational state, and coupling and releasing of ATP. Other pump volumes have one set of values for each $[\mathrm{ATP}]^i$, even though the same behavior remains.

The experimental conditions used to study the major effects of various ligands including ATP on PTX-modified $\mathrm{Na}^+/\mathrm{K}^+$-ATPase [Artigas and Gadsby, 2004] are rather different and this poses a problem in terms of comparison with our results. The inhibitory effect elicited by ATP as predicted by our model has been not verified experimentally and it was unexpected. This result raises an important point that may be worth being experimentally validated.

Our results suggest that in the presence of palytoxin, the extent of phosphorylation from ATP is greatly reduced probably by a PTX-promoted rapid dephosphorylation step that could, at high concentrations of ATP, lead to inhibition of ATP binding. This reinforces the notion that the phosphorylated intermediates formed from ATP are different and this may change PTX affinity and the overall behavior of the pump. There are some reports in the literature that could support this result [Tosteson et al., 2003]. These results have been obtained from a parametric study of the state and transitions rewards of our model.

We have also investigated properties related to species (ions or molecules) depletion, i.e. when there are no species in one side of the cell. For example, the events "atpAllBounded" and "ptxAllBounded", where all ATP and palytoxin molecules are bound to the pump, respectively. These events can be created in PRISM using *labels*, one of its features (Figure 8.2).

Species depletion properties state that these events eventually (**F** operator) will always happen ($\mathcal{P}_{>=1}$ operator). For example, in every scenario the event "ptxAllBounded" always eventually happens.

One could check how long it takes for those events to happen. For that we have to use a time reward, and reward properties, such as the one shown in Figure 8.2. The event "ptxAllBounded" is expected to happen in 30.4379 seconds in the $[\mathrm{ATP}]^i = 10$ mM scenario. This event is sensitive to the parameter $[\mathrm{ATP}]^i$– in the 100 mM scenario, since it happens in 49.4342 seconds.

8.3 Sodium Enhances the PTX Inhibitory Effect on the Pump

These results have been obtained using our second model (`ptx2008a`), which focuses on the role of sodium and potassium reactions.

Considering a single pump, a pump volume of 10^{-22} L, a Control scenario, at instant T=100, the expected rewards associated with the state PTXE is 36.2195. In other words, in 100 seconds, the PTX inhibits the pump 36.11% of the time. In a High $[Na^+]^o$ scenario, the expected reward associated with PTXE changes to respectively 45.3599, 42.42% of the time. Therefore, as we increased $[Na^+]^o$, the likelihood of the pump to be bound exclusively to PTX increased 17.46%. This result can be seen in Figure 8.3, which represents the probability of PTX inhibiting the pump for our three different scenarios, and also its time series version in Figure 8.4.

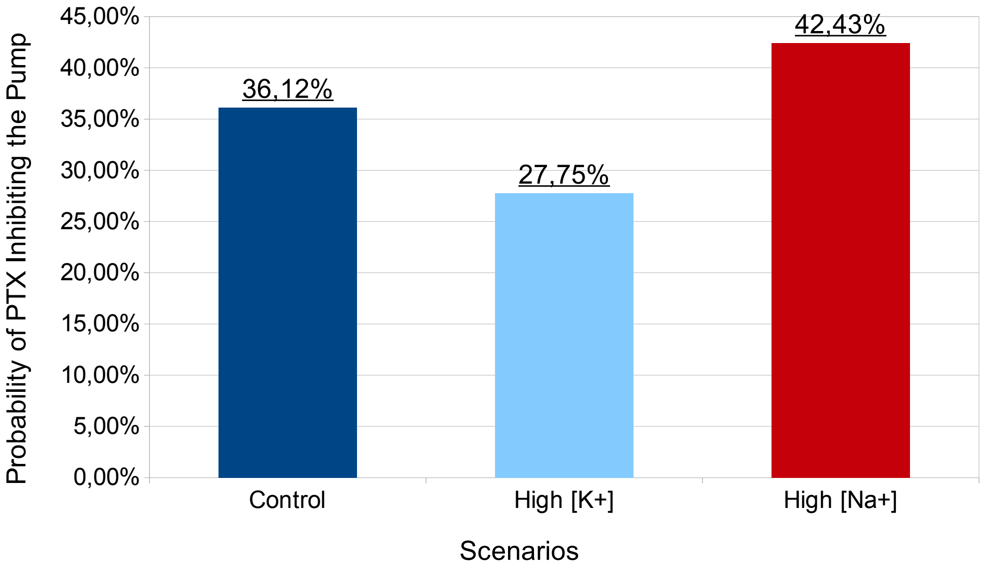


Figure 8.3: Probability of PTX Inhibiting the Pump for Different Scenarios in 100 seconds: Control (normal ion concentration); High $[K^+]^o$ scenario (10 times more potassium than normal) which reduces PTX effect by 23.17%; and High $[Na^+]^o$ scenario (10 times more sodium than normal) which enhances PTX effect by 17.46%.

This result suggests that sodium enhances PTX action, and as consequence people with electrolyte disturbances would be more vulnerable to this toxin. Sodium disturbances appears in different forms (i.e. hypernatremia) and have different causes, such as diabetes insipidus, Conn's syndrome and Cushing's disease [Yamada and Inagaki, 2002]. Sodium concentration could be reduced in order to reduce PTX action. However, this is a solution to be taken with caution since sodium is necessary for survival and its absence would shut down the pump. This is particularly interesting since PTX is found in marine species, which inhabit an environment with a high sodium concentration.

8.4 Potassium Inhibits the PTX Action on the Pump

These results have been obtained using our second model (`ptx2008a`), which focuses on the role of sodium and potassium reactions.

As the potassium concentration increases, an event opposite to the one discussed the previous section is observed. In a High $[K^+]^o$ scenario, the expected reward associated with PTXE changes to respectively 29.2241, 27.74% of the time. Therefore, as we increased $[K^+]^o$, the likelihood of the pump to be bound exclusively to PTX decreased 23.17%. This result can be seen in Figures 8.3 and 8.4.

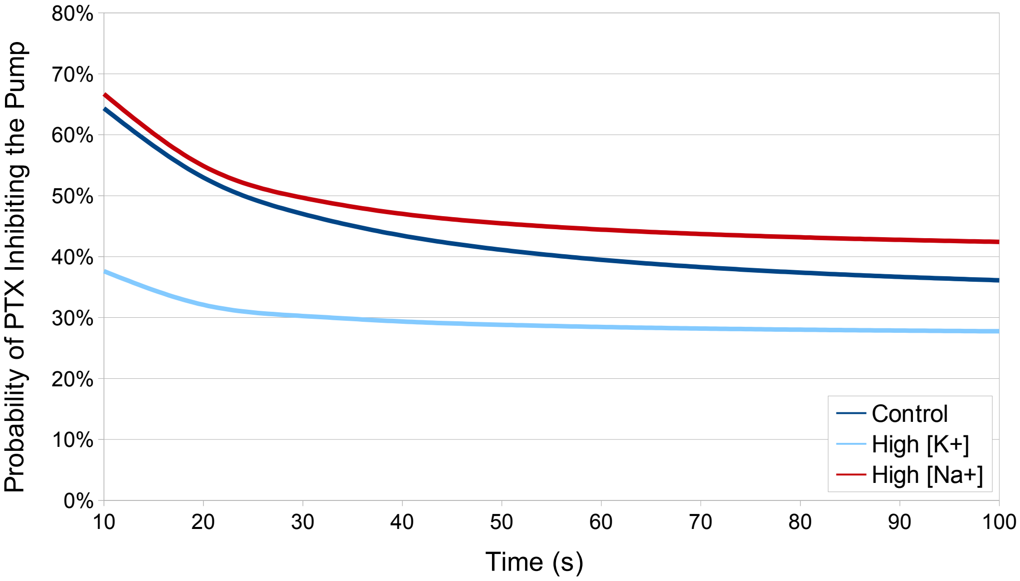


Figure 8.4: Probability of PTX Inhibiting the Pump Time Series for Different Scenarios: Control (regular ions concentrations); High $[K^+]^o$ scenario (10 times more potassium than regular concentration) which reduces PTX effect; and High $[Na^+]^o$ scenario (10 times more sodium than regular concentrations) which enhances PTX effect.

This result suggests that potassium inhibits PTX action. Therefore individuals with diets low in potassium, or with a pathology which decreases the potassium concentration in their metabolism could be more vulnerable to PTX. Potassium concentration could be increased to inhibit PTX action. In a similar way to sodium, there is another fine line here since a maximum amount of potassium is tolerated for one individual. There are a number of causes associated with a high potassium concentration (hyperkaulemia), such as renal insufficiency, Addison's disease, Gordon's syndrome and Rhabdomyolysis. Both results have been obtained from a parametric study of the state and transitions rewards of our model.

Regarding depletion events, in every scenario the event "ptxAllBounded" eventually always happens. That is not the case for the event "naOutDepletion". The event "kInDepletion" is sensitive to the parameter $[K^+]^o$ – in the Control scenario, its property is true, while in the High $[K^+]^o$ scenario, the property becomes false, because it is more difficult to deplete internal potassium since there is 10 times more potassium. One could check how long it takes for those events to happen. For that we have used a time reward, and reward properties, such as the one shown in Figure 8.2. The event "ptxAllBounded" is expected to happen in 1.7513E-5 seconds.

8.5 Induced Electric Current Measurements

These results have been obtained using our third model (`ptx2008b`), which focuses on the known inhibitory effect of potassium on PTX action.

The induced current carried by potassium ions across the membrane can be measured using the Goldman-Hodgkin-Katz (GHK) flux equation (divided in Equations 8.1 and 8.2 shown below), which describes the ionic flux as a function of transmembrane potential and ion concentrations inside and outside the cell [Hille, 2001].

This equation allows studying quantitatively the behavior of the induced current under different conditions, such as the scenarios previously mentioned.

$$J_{ion} \;=\; P_{ion}\; z_{ion}^2 \;\frac{F^2\; V_m}{R\; T}\; \frac{[ion]^i \; e^{\frac{z_{ion}\;F\;V_m}{R\;T}} \;-\; [ion]^o}{e^{\frac{z_{ion}\;F\;V_m}{R\;T}} \;-\; 1} \tag{8.1}$$

The P_{ion} part of Equation 8.1 is another formula, which represents the permeability of the membrane for that ion and it is shown in Equation 8.2 below.

$$P_{ion} \;=\; \gamma_1[PTXE] \;+\; \gamma_2[PTXATPhighE] \;+\; \gamma_3[PTXATPlowE] \tag{8.2}$$

Descriptions, values and units for constants, such as V_m (transmembrane potential) and F (Faraday constant) are shown in Table 4.

Unlike simulations, which could run into local minima problems and demand a prohibitive number of traces to obtain an approximation of expected values, our PMC approach always obtains the real expected value for our model due to its exhaustive exploration of all states.

This is particularly important for electric current measurements, because it depends on the probability of states `PTXE`, `PTXATPhighE` and `PTXATPlowE`, as well as internal and external ion concentrations. These state probabilities and ion concentrations change drastically for different scenarios, as we have in the previous subsections.

Electric Current Rewards

```
rewards "pos_jK"
 (jK>=0): jK;
endrewards

rewards "neg_jK"
 (jK<0): (-1)*jK;
endrewards
```

Instantaenous Current Reward Property

$\mathbf{R}\{\text{“pos\_jK”}\}_{=?}\ [\ \mathbf{I}{=}\mathtt{T}]$

What is the expected instantaneous reward for the positive current `pos_jK` at time T?

PRISM allows the calculation of the correct expected electric current through rewards. Unfortunately, PRISM does not accept negative rewards, therefore we have to separate positive (`pos_jK`) and negative (`neg_jK`) rewards. The negative reward is multiplied by (-1), in order to allow its computation.

Measurements are performed through instantaneous reward properties, which obtain the reward value precisely at `T` time. A negative induced current indicates normal potassium ion flux (two potassium ions go inside), while a positive one indicates an abnormal flux (two potassium ions go outside).

In the Control scenario, at T = 100, the induced current is positive $(52,547\frac{A}{m^2})$, which means that potassium ions are moving in favor of their gradient. In the High $[K^+]^o$ scenario, the induced current becomes negative $(-426,790\frac{A}{m^2})$ – potassium ion exchange is completely changed, having its direction reversed. Both states `PTXATPhighE` and `PTXATPlowE` play a major role in that equation, except that their coefficients are different ($\gamma_1 = 0.15$ and $\gamma_3 = 0.9$, respectively). This reversion of ion exchange suggests that high concentrations of $[K^+]^o$ could fight PTX action.

8.6 A Probabilistic and Quantified Kinetic Model

These results have been obtained using our first model (`ptx2007`), which focuses on the role of cell energy reactions.

The Albers-Post model of the Na^+/K^+-ATPase was first proposed in [Post et al., 1972]. It is a kinetic model (and also a directed graph) which describes the set of states of the pump, and the chemical reactions that lead from one state to another, consuming or producing ligands.

$$P(r_i) = \frac{r_i}{\sum\limits_{i=0}^{N} r_i} \tag{8.3}$$

We are able to quantify this kinetic model using PMC through state and transition (rate) rewards. These rewards have been presented in previous chapter and essentially count the expected number of times that a state (or transition) is observed. We can calculate a state probability by dividing its reward by the sum of all state rewards (Equation 8.3). This is also applied to reactions and ligands, such as ATP.

Once we have all the states and reactions probabilities, we associate colors to each of them, allowing their visual representation. The kinetic model can be colored using a jet palette, which is often associated with temperatures. In this pallete, probabilities transit from red to blue, or from likely to unlikely, respectively.

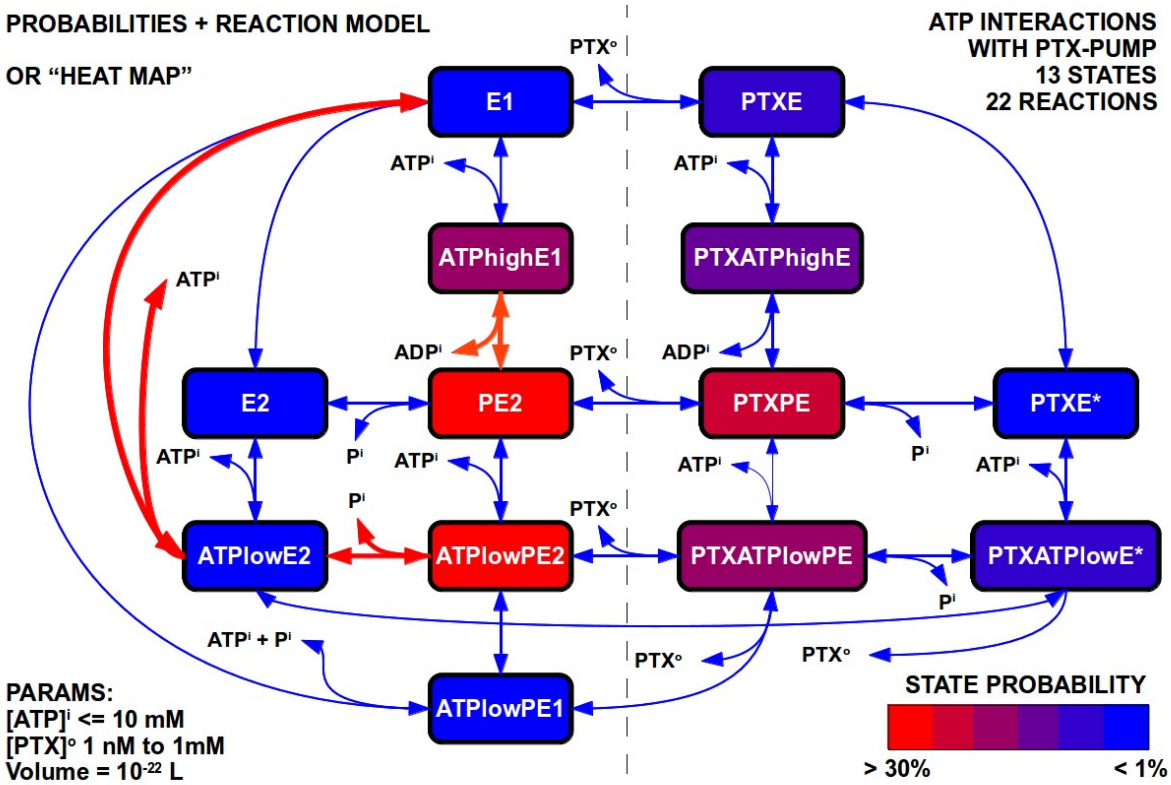


Figure 8.5: Heat Map: the kinetic model of the Na$^+$/K$^+$-ATPase with state and rate probabilities represented as colors. Each state and rate is colored based on its probability. Red states/rates are likely while blue states/rates are improbable. This could be a visual tool for biologists as it shows model dynamics and overlooked conditions.

This modified kinetic model is called a heat map. Red states and reactions are more probable or hot, while blue states and reactions are unlikely or cold. An example of the heat map for our first model (called `ptx2007`) can be seen in Figure 8.5, where the states `PE2` and `ATPlowPE2` (phosphorylated pump open to the external side and that same state with ATP bound to its low affinity binding site, respectively) are more probable, and reactions between `E1` and `ATPlowE2` (pump open to the internal side and pump open to the external side with ATP bound to its low affinity binding site, respectively) occur more often.

For example, the reaction between the states `E1` and `ATPlowE2` is one of the most active reactions, while the states themselves are one of the most inactive states. This could suggest that either these states are temporary or there might be an intermediary and unknown state between these two states.

After contacting Rodrigues and co-workers, they have suggested that this intermediary state is the pump transitioning between open to the external side and open to the internal side, which is exactly the effect of the reaction between those two states.

This odd behavior could reflect an imprecision on the kinetic model description itself, which might not include all the existing states and reactions of the pump interacting with palytoxin. Novel experiments could be performed in order to validate this behavior and further improve the current description of PTX-pump interactions.

The heat map could be a valuable tool for biologists as it shows model dynamics and it could be used to suggest overlooked experiments. Since the kinetic model is an abstraction suggested by experimental data, it could be incomplete, which the heat map would assist towards its completion. It raises several questions, especially about likely reactions involved with improbable states.

8.7 Alternative Models

Alternative models can be produced using different approaches and abstractions. One could favor one aspect of the system over another. Also, even different tools, based on other formal methods, could have been used.

Our models are focused on the current understanding of the pump ion exchange and its interactions with palytoxin. However, the ion exchange itself is an event very difficult to observe using currently available experimental procedures. Techniques such as Patch-Clamp analysis observe the behavior of the induced electric current by several ions. However, these fluctuations occur due to millions of ion exchanges, therefore the granularity of our model is incompatible with such technique, which makes it difficult to validate our results experimentally.

An alternative model that would reflect more accurately current experimental procedures would not focus on the ion exchange and the discretization of the species' concentrations. The reasoning behind this is that the volume of ions in either side of the cell is so big that two or three ions moving from one side to the other does not have an impact on the cell behavior. From the perspective of the pump, one ion which has been exchanged is rapidly replaced by another ion. It is similar to an infinity flux or quantity of ions in either side. The important information is the concentration: how the species' concentration of one side compares to the other side.

In an effort to bridge the gap between our models and the experimental procedures, we have built an initial model taking these possibilities into consideration. We have removed the cell volume and the discretization of species' concentrations. The species' modules no longer have variables storing the number of ions in either side of the cell. Naturally, the species' transitions no longer have to update the species' variables.

This change made the models significantly smaller than the previous models. For example, in the case of the `ptx2007` model, the number of states reduced from 376 states to eight states, and the number of transitions from 1912 transitions to 26 transitions. Even for PMC, which has to perform floating point operations, this is a simple model. However, it is still relevant since it more similar to currently available experimental tests and can analyze several relevant state-of-the-art experimental studies.

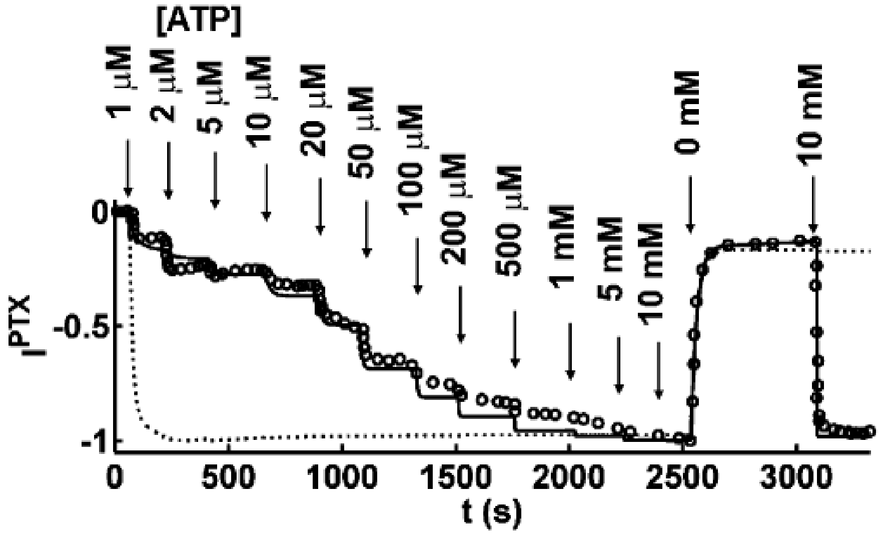


Figure 8.6: The induced electric current by the ions over time. Each segment of the time series is a different concentration of ATP. Adapted from Rodrigues et al. [2008b].

In order to compare this new and smaller model with the experimental procedures, we have selected an experiment devised by Artigas et al., where the ATP concentration is modified over time, while the induced electric current by the ions is observed. This experiment is shown in Figure 8.6, adapted from Rodrigues et al. [2008b]. In different time periods, ranging from 50 to 600 seconds, the experiment is subject to different ATP concentrations. It starts with 1µM, then it is increased to 2µM, followed by an increase to 5µM, and so on (as denoted by the arrows in the graphic). Finally, at 2400s, when the ATP concentration is 10 mM, it is completely removed from experiment. At 3000s, another 10 mM of ATP are added to the experiment. This has been performed in [Artigas and Gadsby, 2002], where they have made their breakthrough, confirming that PTX modulates the pump, which can behave like and ion channel, and observing the role fo ATP in such system.

This modification of ATP concentration over time presents in itself a challenge for our modeling approach and in general to the model checking field. Since the model is built *a priori*, we can not verify a property, change the model structure as we change one parameter *a posteriori*, and continue the verification. We have to build the model again. Therefore, we have several independent verifications, each one for a different ATP concentration, which can not be chained one after another using PRISM and our modeling approach. Nonetheless, we have performed the several experiments and chained them in a single time series in an attempt to visualize how it compares with the original experimental procedure.

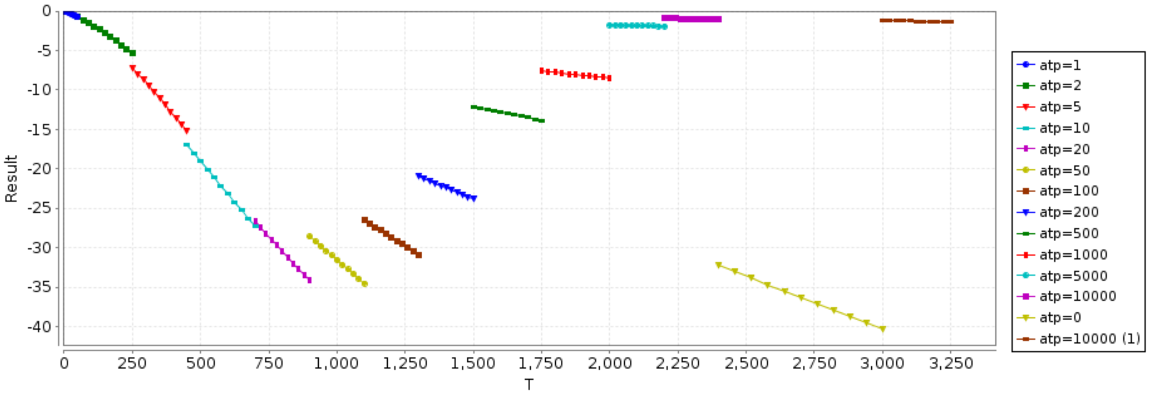


Figure 8.7: Our results for the induced electric current by the ions over time. Each segment of the time series is a different concentration of ATP.

Our results have been shown in Figure 8.7. We have performed several different verifications using this alternative model and PRISM, where each verification uses a different concentration of ATP, corresponding to the ones used by Artigas et al.. For example, in the first verification, the ATP concentration is set to 1µM, which is followed by another verification using an ATP concentration of 2µM. In each verification, the electric current reward is measured for ten time steps of the given period, which is different for each test and have been roughly estimated since the paper does not provide this information. Table 8.3 shows the parameters which have been used to perform the experiment shown in Figure 8.7.

[ATP]	t_i (s)	Interval	t_e (s)
1 µM	0	50	50
2 µM	50	200	250
5 µM	250	200	450
10 µM	450	250	700
20 µM	700	200	900
50 µM	900	200	1100
100 µM	1100	200	1300
200 µM	1300	200	1500
500 µM	1500	250	1750
1 mM	1750	250	2000
5 mM	2000	200	2200
10 mM	2200	200	2400
0 mM	2400	600	3000
10 mM	3000	250	3250

Table 8.3: The parameters which have been used in the experiment shown in Figure 8.7 which attempted to reproduce the results of Artigas and Gadsby [2002].

Although the behavior is not the same as the Artigas et al. experiment shown in Figure 8.6, we can see an overall trend on the rate of change or the inclination of the time series, specially in the initial segments, where the model behaves similarly to the original experimental procedure. From one segment to another, there is a difference which does not chain one experiment with the other. This is because each verification is independent from another. As the experiment progresses (once the ATP concentration reaches 0 mM, roughly between 2500 s and 3000s), that trend is lost. The rate of change should zero, as there is no induced electric current. This difference should be further studied in future works, in an attempt to explain why it happens. However, after we change the ATP concentration back to 10 mM again, there is a rate of change decreasing, although not as abrupt as in the experimental test.

The lack of precise information about the original experiment performed by Artigas et al. is a problem in order to reproduce it, therefore our parameters could be slightly different than the original ones. Nonetheless, our results have shown a similar trend on the rate of change of the induced electric current. In order to chain one experimental scenario with another, one modeling approach or formal method would have to be developed which would change the model structure in real time, while the verification is being performed. Another approach would be creating a modeling which chains the output of a verification with the input of another verification. The fact that the trends have been reproduced is an indication that this alternative modeling can precisely reflect the original experimental tests. Naturally, further studies are necessary, however as the model created is very efficient, this approach is promising.

Summary. In this chapter we have presented the results that we have obtained in this dissertation. We have built four formal models of the toxin palytoxin (PTX) interactions with the sodium-potassium-pump using a probabilistic model checking approach through the PRISM model checker. This toxin disrupts the regular behavior of the pump, compromising several biological processes. Each model focuses in a different aspect of the system. For example, our first model focuses on the role of cell energy, while the second model covers the sodium and potassim reactions.

After that we have used parametric study to the dimensions of our models, which allowed the investigation of extreme conditions, such as diseases which increase ligand concentrations or fatal exposures to PTX. We have used quantitatives properties to quantify every aspect of the model, such as state and reaction probabilities.

Each model provided suggestions for the behavior of the system, such as that high concentrations of ATP inhibit PTX action, sodium enhances PTX, and potassium inhibits PTX. We have also quantified the induced current created by ions exchange through the Goldman-Hodgkin-Katz (GHK) flux equation. The complete model reinforced our previous results.

Chapter 9

Additional Contributions

Outline. In this chapter we present additional contributions of this dissertation. We have used probabilistic model checking results to visually annotate electrophysiology models, obtaining heat maps which represent the most common states and reactions, as well as model dynamics. We have built a tool called *dot2heatmap*, which helps producing heat maps from graphs described in the DOT language using PMC results. We have also created a tool called *MCHelper*, a prototype of a model checking support environment, which helps manage and treat the information and results obtained through PMC. This tool has been further developed by João Sales Amaral in his undergraduate project. Finally, we have also developed a tool called *PrismRecipes*, which helps writing PRISM models and properties, which often exhibit several regularities and repetitions.

9.1 dot2heatmap Tool

We have created a tool called *dot2heatmap* which automatically generates heat maps. It receives two inputs:

- A graph description of the model encoded in the DOT language.
- A file containing the values associated with each node and edge of the graph.

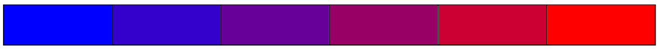
It outputs a customized version of the graph description, which is annotated with appropriate colors for each node and edge based on the values provided for them. A few simple parameters must also be included:

- **Start color:** associated with lower values. For example: blue, or #0000FF.

- **End color:** associated with higher values. For example: red, #FF0000.

- **Number of classes:** number of different colors, from start color to end color.

Suppose the start color blue, end color red and six different classes. Lowest-valued nodes and edges are associated with the start color, highest-valued ones with the end color, and intermediary ones with intermediary colors, following the color schema created by the tool and shown below:

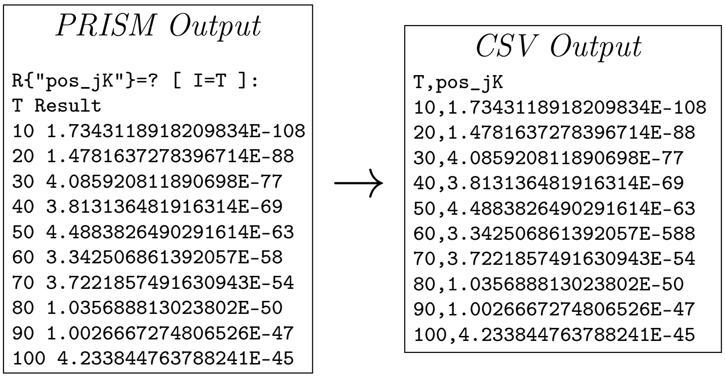


This tool is actually general purpose and could be used with any graph-based model and also any approach to quantify the nodes and edges, despite being completely unrelated to PMC. It is freely available online [Braz, 2012a].

The DOT language was chosen because it is one of the main languages used to formally describe graphs [GraphViz, 2013]. However, the tool could be easily extended to other graph descriptions, such as GEXF [GEXF, 2013] and GraphML [GraphML, 2013].

9.2 MCHelper Tool

We have created a tool called *MCHelper*, a support environment for formal verification, with the objective of helping the user to deal with several aspects of model checking, such as parsing the results and logs, task scheduling and outputting the model graph. It is freely available online [Braz, 2012b].



PRISM Output

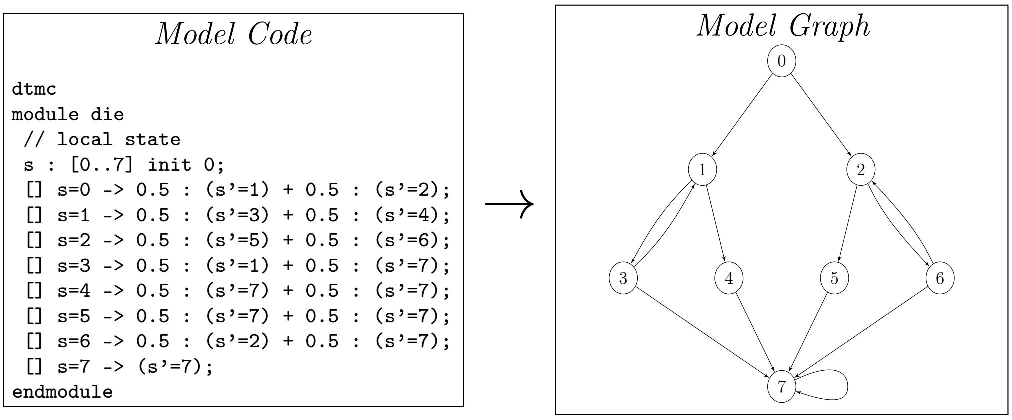
```
R{"pos_jK"}=? [ I=T ]:
T Result
10 1.7343118918209834E-108
20 1.4781637278396714E-88
30 4.085920811890698E-77
40 3.813136481916314E-69
50 4.4883826490291614E-63
60 3.342506861392057E-58
70 3.7221857491630943E-54
80 1.035688813023802E-50
90 1.0026667274806526E-47
100 4.233844763788241E-45
```

CSV Output

```
T,pos_jK
10,1.7343118918209834E-108
20,1.4781637278396714E-88
30,4.085920811890698E-77
40,3.813136481916314E-69
50,4.4883826490291614E-63
60,3.342506861392057E-588
70,3.7221857491630943E-54
80,1.035688813023802E-50
90,1.0026667274806526E-47
100,4.233844763788241E-45
```

Figure 9.1: The *MCHelper* was initially developed to parse the considerable amount of results which are produced when you explore several parameters of the model. For example, one output per property, per parameter scenario.

This tool was initially created to parse the considerable amount of results which are produced when you explore several parameters of the model, as well as checking properties individually (for example, one output per property, per parameter scenario). The output of the tool is a CSV (Comma-separated values) file (Figure 9.1), which is very practical for using in spreadsheets, and other mathematical and statistical environments, such as MATLAB, Octave and the R language. Eventually, a novel version of PRISM (4.0.3) was developed which included outputting CSV files. Nonetheless, this tool still had potential to be further extended.



Model Code

```
dtmc
module die
 // local state
 s : [0..7] init 0;
 [] s=0 -> 0.5 : (s'=1) + 0.5 : (s'=2);
 [] s=1 -> 0.5 : (s'=3) + 0.5 : (s'=4);
 [] s=2 -> 0.5 : (s'=5) + 0.5 : (s'=6);
 [] s=3 -> 0.5 : (s'=1) + 0.5 : (s'=7);
 [] s=4 -> 0.5 : (s'=7) + 0.5 : (s'=7);
 [] s=5 -> 0.5 : (s'=7) + 0.5 : (s'=7);
 [] s=6 -> 0.5 : (s'=2) + 0.5 : (s'=7);
 [] s=7 -> (s'=7);
endmodule
```

Figure 9.2: Further development of the *MCHelper* allowed generating graphical representations of the model using the DOT language.

This is an initial implementation, and could be further extended to several others functions, such as managing computer nodes, allocation of model checking tasks, rescheduling of tasks due to error recovery. Errors include, for example, insufficient memory and model checking engine parameters. As his undergraduate monograph, João Lucas Ademir Sales Amaral further developed the tool, including the parsing of logs, task scheduling and outputting model graphs (Figure 9.2).

9.3 PrismRecipes Tool

We have also created a tool called *PrismRecipes* to help writing PRISM models and properties. There are several parts of PRISM models in general which are repetitive, such as Reward structures and quantitative properties. Specific modeling approaches, such as the one we have used for electrophysiology models, also have their own regularities. For example, chemical reactions are usually translated into PRISM commands in a similar manner.

The tool is currently capable of:

- Creating rewards for variables and labels (f in Figure 9.3).
- Specifying instantaneous and cumulative reward properties (g in Figure 9.3).
- Translating chemical reactions to PRISM commands.

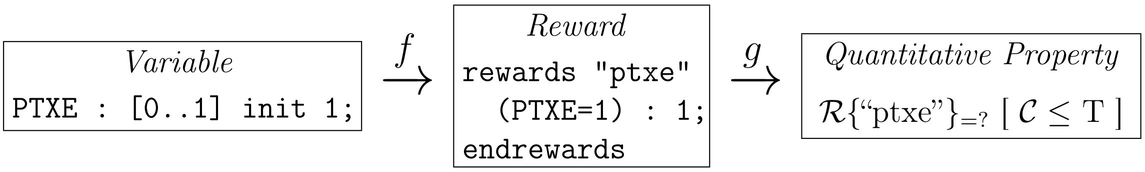


Figure 9.3: The *PrismRecipes* tool can, for example, create rewards for variables (f) and quantitative properties for rewards (g).

Eventually *PrismRecipes* could be integrated into *MCHelper*. It could also be generalized to other model checkers and modeling approaches, such as MRMC [Katoen et al., 2011] and Uppaal [Bengtsson et al., 1996]. It is freely available online [Braz, 2012c].

Summary. In this chapter we have presented additional contributions of this dissertation, such as the tool *dot2heatmap*, which helps producing heat maps from graphs described in the DOT language using PMC results. We have also presented the tool *MCHelper*, an initial implementation of a model checking support environment, which helps manage and treat the information and results obtained through PMC. Finally, we have also shown the tool *PrismRecipes*, which helps writing PRISM models and properties, which often exhibit several regularities and repetitions.

Chapter 10

Conclusions

This dissertation presented the modeling and analysis of transmembrane ionic transport systems and its interactions with toxin palytoxin using the formal verification technique of probabilistic model checking.

The sodium-potassium pump (Na^+/K^+-ATPase) is a cellular structure which is responsible for exchanging sodium and potassium ions through the plasma membrane against their concentration gradients. This exchange costs cell energy (Adenosine Triphosphate or ATP). The pump can be affected by diseases and toxins, which disrupt its behavior. Its correct performance is necessary for all animal cells, otherwise several biological processes such as cellular volume control and heart muscle contraction could be compromised, along with the health of an individual.

One of these toxins is the palytoxin (PTX), which binds to the pump, opening both of its doors. This disturbs the control of ion flux, which are now free to move in favor of their gradients. Artigas and Gadsby [Artigas and Gadsby, 2002; Artigas, 2003b,a; Artigas and Gadsby, 2004], followed by Rodrigues and co-workers [Rodrigues et al., 2008b,a, 2009a,b], have studied these effects thouroughly after several experiments and computational simulations, respectively.

Each of the publications of Rodrigues and co-workers analyzes a different aspect of the PTX interactions with the pump. We have followed these works, building four different models, one for each publication. We have used a probabilistic model checking (PMC) approach to model and analyze each of these models, which has allowed their formal, exaustive and automatic exploration, checking if they respect a set of given properties in a special type of probabilistic logics.

We have used the PRISM tool, a probabilistic model checker which is used to formally explore stochastic systems such as the PTX-pump complex. PMC has allowed us to investigate the models using properties about biological events, which have been expressed in probabilistic logics, e.g. “What is the probability of being in PTX related sub-states?” or “What is the expected time for the occurrence of ATP depletion?”.

After building the models, we have explored them using a parametric approach, where we have changed the concentrations for each of the ligands (ATP, Adenosine Diphosphate or ADP, Phosphate or P_i, Sodium or Na^+, Potassium or K^+, and PTX). This has allowed simulating extreme situations such as a disease where the sodium levels are decreased or a fatal exposure to PTX. These different conditions are called scenarios, for example, the Control scenario are the normal physiological conditions, while the High $[Na^+]^i$ scenario has an increased concentration of intracellular sodium.

We have obtained three different results worthwhile experimental validation. Our first model, called `ptx2007`, which focuses on the role of ATP [Rodrigues et al., 2008b], indicated that as $[ATP]^i$ increases, the probability of PTX related sub-states decreases. For example, from our Control scenario to our High $[ATP]^i$ scenario, that probability is reduced by 38.3520%. This suggests that high $[ATP]^i$ could inhibit PTX action, which implies that individuals with ATP depletion are more susceptible to PTX effects. ATP deficiency appears in different forms, such as in brain disorders, for example, stroke. The study of the role and ability of ATP to change our Na^+/K^+-ATPase model behavior is even more important, because its production can not be stimulated directly.

Our second model, called `ptx2008a`, which focuses on sodium and potassium ions [Rodrigues et al., 2008a], suggested that high $[Na^+]^o$ could enhance PTX effects. For example, from the Control scenario to the High $[Na^+]^o$ scenario, the probability of PTX inhibiting the pump increases 17.46%. This suggests that electrolyte (sodium and potassium ions) disturbances could render an individual susceptible to the toxin. Since PTX is found in an environment with a high $[Na^+]^o$, this does not seem to be a coincidence. Disturbances which present high $[Na^+]^o$ levels in the blood have different causes, such as diabetes insipidus [Aidley and Stanfield, 1996]. An opposite behavior is observed for high $[K^+]^o$. From a Control scenario to a High $[K^+]^o$ scenario, PTX effects are reduced by 23.17%. Both results suggest that electrolytes could be changed to reduce PTX effects by decreasing $[Na^+]^o$ and increasing $[K^+]^o$. Since electrolyte levels in the blood can be manipulated up to a certain degree, the study of their role and capability to change the behavior of our pump model is even more important.

Our third model, called `ptx2008b`, which focuses on the known inhibitory effect of potassium on PTX action [Rodrigues et al., 2009a], suggested that in the Control scenario, the most active state of the pump is the PTX-pump complex with an ATP molecule bound to its high affinity binding site (`PTXATPhighE`). In the High $[K^+]^o$ scenario, the most active state shifts to a similar one, except that now ATP is bound to its low affinity binding site (`PTXATPlowE`).

We have used the Goldman-Hodgkin-Katiz (GHK) flux equation to measure the induced current caused by ion exchange. In the Control scenario, the induced current is positive $(52,547\frac{A}{m^2})$, which means that potassium ions are moving in favor of their gradient. In the High $[K^+]^o$ scenario, the induced current becomes negative $(-426,790\frac{A}{m^2})$ – potassium ion exchange is completely changed, having its direction reversed. Both states PTXATPhighE and PTXATPlowE play a major role in that equation, except that their coefficients are different ($\gamma_1 = 0.15$ and $\gamma_3 = 0.9$, respectively). This reversion of ion exchange suggests that high concentrations of $[K^+]^o$ could fight PTX action.

Our fourth and final model, called ptx2009, which represents the complete Albers-Post model (the sodium-potassium-pump cycle) and its interactions with PTX [Rodrigues et al., 2009b], reinforced the results observed in the previous models.

We have also enhanced the kinetic model of the pump, which is used to describe the states and reactions of the system, with probabilities, creating a heat map. It reveals unexpected situations, such as a frequent reaction between unlikely states, which suggests that either these states are temporary; or there is an unknown state between those two. This odd behavior could reflect an imprecision on the kinetic model description itself, which might not include all the states and reactions of the pump interacting with palytoxin. Novel experiments could be performed in order to validate this behavior and further improve the current description of PTX-pump interactions.

We have also started building a set of initial tools which we expect that in the long run will be helpful to probabilistic model checking research. The PrismRecipes tool helps writing PRISM models which often show regularities and repetitions, while the MCHelper tool helps parsing results and logs which are mass produced during a parametric study such as the one we have used.

We have shown in this dissertation that PMC can be used to obtain valuable insight of transmembrane ionic transport systems in a simple and complete way. This type of analysis can provide a better understanding of how cell transport systems behave, giving a better comprehension of these systems. These systems often are targets of toxins and other substances such as drugs, and their study could lead to the discovery and development of drugs and novel ways to fight toxins.

10.1 Future Works

There are several open questions and frontiers for this work, which could be explored in future works. We are currently working towards some of them, such as using a qualitative approximate analysis of our models and building an integrated and hierarchical environment or approach to cope with PMC and other techniques, such as simulations. We have briefly discussed each one of these open questions below, and we hope that they will be explored in the future.

Validation. We have obtained several results using formal computational models, such as the indication that cell energy and potassium inhibits palytoxin action, or the other way around, that sodium enhances palytoxin action. However, as good as a computational model can be, its results must be confronted with supporting experimental evidence obtained through wet lab experiments. Therefore, one could attempt to reproduce the conditions we have explored computationally in an experimental environment.

Expansion. We have built four formal models, following the works of Artigas and Gadsby, and Rodrigues and co-workers. Each one of these models is focused in one aspect of the sodium-potassium pump cycle interacting with the palytoxin. One model focuses in the cell energy role, while another one focuses in the potassium inhibitory effects of palytoxin. However, these models follow the current understanding of the pump behavior based on experimental evidence and currently available experimental procedures, which in turn could be incomplete, including unnecessary states or omitting unknown or not fully understood ones. The reaction rates could be slightly wrong, as novel experimental techniques could offer improved precision in their measurements. Our models could be easily further expanded to new states.

Exploration. We have explored the different dimensions or aspects of our models, namely the molecules, ions and pump volume, in order to observe their impact in the model behavior. We have used several orders of magnitude for each dimension above or below their reference values (for example, the concentration of phosphate is 4.95 M^{-3} and we have studied 4.95 M^{-2} and 4.95 M^{-4}). These changes could be interpreted as simulating a disease, syndrome or conditions. Although this exploration has revealed several interesting results, such as increasing cell energy reduces palytoxin action, it has been done so far in a naive way. We would like to know the precise point where model behavior changes. Also, some of these dimensions still are challenging, such as pump volume, which drastically increase model complexity.

Adaptation. We have studied the effects of the toxin palytoxin in the sodium-potassium-pump, which essentially open the pump more than it should be and disrupt several cell processes and physiological concentrations of ions. Our models could be adapted to other toxins (for example, ouabain), deadlier or more common. This adaptation requires further study of these toxins, state-transition models for pump and its interactions with the toxin, ligands concentrations and reactions rates. Furthermore, these models could be adapted to drugs (e.g. digitalis) which behave similarly to toxins, by changing physiological conditions to trigger a cell response, in order to study drug effects at the cellular level. Finally, these toxins or drugs might block the pump in a particular state, instead of opening it.

Approximate Analysis. We have used exact probabilistic model checking in order to obtain accurate results. However, as our models grew in size and complexity, properties began to take too long to be verified, specially quantitative ones. We have performed an initial study on approximate analysis of formal models, using methods available in the PRISM model checker such as confidence intervals. However, the obtained results were far off from the exact ones and also did not follow the same behavior, which lead us to halt progress in this line of research. Further studies could reveal the appropriate methods and their parameters to use in electrophysiology models. This would allow studying larger pump volumes and macroscopic studies using several pumps.

Hierarchical Approach. We have used a different computational approach to study electrophysiology models – the probabilistic model checking. Usually these models are studied using simulations, such as the works of Rodrigues and co-workers which have used a set of ordinary differential equations to explore the interactions of palytoxin with the sodium-potassium-pump. There are also several other approaches, such as the Gillespie's method for stochastic simulations, not to mention the experimental results itself, which are electric current time series obtained using *Patch-Clamp* techniques. However, each approach explores different aspects of the same models, which difficults their comparison and correlation. We would like to build tools which allowed us to obtain and compare results using different approaches, using experimental studies, simulation techniques and formal approaches.

Mixed or Hybrid Approach. We have studied discrete and stochastic formal models for small pump volumes. However, for larger volumes this discretization of the environment could be unnecessary, and some models might exhibit deterministic behavior. We would like to develop novel mixed or hybrid approaches, which could deal with both discrete and continuous values, as well as deterministic and stochastic behavior. For example, in the sodium-potassium pump, the external sodium concentration is larger than the internal one – the first could be continuous while the second could be discrete. Larger volumes such as the external environment could be represented as ordinary differential equations, while smaller environments such as the pump volume, could be represented as Markov chains. This direction would be a major challenge and if accomplished could be extremely inefficient due to dual nature of the model, however, we believe it would be more close to the real cell behavior.

Other Models. Finally, different types of pumps could be modeled, such as the calcium pump which plays a major role in the heart muscle contraction, or even other types of transmembrane ionic transport systems, such as ion channels, which play a major role in synapses and other nerve impulse related cell processes. There are other biological systems, such as gene regulatory networks and cell signalling pathways, which have already been modeled using formal methods, and could be studied as well.

Bibliography

Aidley, D. J. and Stanfield, P. R. (1996). *Ion channels : molecules in action*. Cambridge University Press. ISBN 0521498821.

Alon, U. (2006). *An Introduction to Systems Biology: Design Principles of Biological Circuits (Chapman & Hall/CRC Mathematical & Computational Biology)*. Chapman and Hall/CRC, 1 edition. ISBN 1584886420.

Artigas, P.; Gadsby, D. C. (2003a). Ion occlusion/deocclusion partial reactions in individual palytoxin-modified Na/K pumps. *Ann N Y Acad Sci*, 986:116–26. ISSN 0077-8923.

Artigas, P.; Gadsby, D. C. (2003b). Na^+/K^+-pump ligands modulate gating of palytoxin-induced ion channels. *Proc Natl Acad Sci U S A*, 100(2):501–5. ISSN 0027-8424.

Artigas, P. and Gadsby, D. C. (2002). Ion channel-like properties of the Na^+/K^+-pump. *Annals of the New York Academy of Sciences*, 976(1):31--40. ISSN 1749-6632.

Artigas, P. and Gadsby, D. C. (2004). Large diameter of palytoxin-induced Na/K pump channels and modulation of palytoxin interaction by Na/K pump ligands. *J. Gen. Physiol.*, 123(4):357--376.

Batt, G., Page, M., Cantone, I., Goessler, G., Monteiro, P., and de Jong, H. (2010). Efficient parameter search for qualitative models of regulatory networks using symbolic model checking. *Bioinformatics*, 26(18):i603–i610.

Bayesia (2013). To leave uncertainty behind you, enter the bayesian network era. `http://www.bayesia.com/en/technology/bayesian-network.php/`.

Bengtsson, J., Larsen, K., Larsson, F., Pettersson, P., and Yi, W. (1996). UPPAAL – a tool suite for automatic verification of real-time systems. In *Proceedings of the DIMACS/SYCON workshop on Hybrid systems III : verification and control:*

verification and control, pages 232--243, Secaucus, NJ, USA. Springer-Verlag New York, Inc.

Berezin, S., Campos, S. V. A., and Clarke, E. M. (1998). Compositional reasoning in model checking. In *Revised Lectures from the International Symposium on Compositionality: The Significant Difference*, COMPOS'97, pages 81--102, London, UK, UK. Springer-Verlag.

BioModels (2013). Biomodels. `http://ebi.ac.uk/biomodels-main`.

BioPAX (2013). Biological pathway exchange. `http://biopax.org/`.

Bollig, B. and Wegener, I. (1996). Improving the variable ordering of OBDDs is NP-complete. *Computers, IEEE Transactions on*, 45(9):993 –1002. ISSN 0018-9340.

Boyce, W. (2008). *Elementary Differential Equations, Ninth Edition Binder Ready Version*. John Wiley & Sons. ISBN 9780470404041.

Braz, F., Cruz, J., Faria-Campos, A., and Campos, S. (2012a). Palytoxin inhibits the sodium-potassium pump - an investigation of an electrophysiological model using probabilistic model checking. In Gheyi, R. and Naumann, D., editors, *Formal Methods: Foundations and Applications*, volume 7498 of *Lecture Notes in Computer Science*, pages 35–50. Springer Berlin Heidelberg.

Braz, F. A., Cruz, J. S., Faria-Campos, A. C., and Campos, S. V. (2012b). A probabilistic model checking approach to investigate the palytoxin effects on the Na^+/K^+-ATPase. In Souto, M. and Kann, M., editors, *Advances in Bioinformatics and Computational Biology*, volume 7409 of *Lecture Notes in Computer Science*, pages 84–96. Springer Berlin Heidelberg.

Braz, F. A. F. (2012a). dot2heatmap tool website. `http://code.google.com/p/dot2heatmap/`.

Braz, F. A. F. (2012b). MCHelper tool website. `http://code.google.com/p/mchelper/`.

Braz, F. A. F. (2012c). PrismRecipes tool website. `http://code.google.com/p/mchelper/`.

Braz, F. A. F. (2012d). ptx2007 model website. `http://www.dcc.ufmg.br/~fbraz/ptx2007/`.

Braz, F. A. F. (2012e). ptx2008a model website. http://www.dcc.ufmg.br/~fbraz/ptx2008a/.

Braz, F. A. F. (2013a). ptx2008b model website. http://www.dcc.ufmg.br/~fbraz/ptx2008b/.

Braz, F. A. F. (2013b). ptx2009 model website. http://www.dcc.ufmg.br/~fbraz/ptx2009/.

Bryant, R. E. (1986). Graph-based algorithms for boolean function manipulation. *IEEE Trans. Comput.*, 35:677--691. ISSN 0018-9340.

C. Dehnert, J.-P. K. and Parker, D. (2013). SMT-based bisimulation minimisation of Markov models. In Giacobazzi, R., Berdine, J., and Mastroeni, I., editors, *Proc. 14th International Conference on Verification, Model Checking, and Abstract Interpretation (VMCAI'13)*, volume 7737 of *LNCS*. Springer. To appear.

Calder, M., Gilmore, S., Hillston, J., and Vyshemirsky, V. (2010). Formal methods for biochemical signalling pathways. In Boca, P., Bowen, J. P., and Siddiqi, J., editors, *Formal Methods: State of the Art and New Directions*, pages 185–215. Springer London. 10.1007/978-1-84882-736-3_6.

Calvert, J., Parker, J., Vermeulen, N., and Penders, B. (2010). *Systems biology, interdisciplinarity and disciplinary identity*, page Chapter 3. Ashgate.

Calzone, L., Chabrier-rivier, N., Fages, F., and Soliman, S. (2006). Machine learning biochemical networks from temporal logic properties. *Transactions on Computational Systems Biology*, 4220:68--94.

CellML (2013). Cell modeling language project. http://cellml.org/.

CellML Models (2013). Cellml models. http://models.cellml.org/cellml.

Chapman, J. B., Johnson, E. A., and Kootsey, J. M. (1983). Electrical and biochemical properties of an enzyme model of the sodium pump. *J. Membr. Biol.*, 74(2):139--153.

Chaves, M., Eissing, T., and Allgöwer, F. (2009). Regulation of apoptosis via the NFκB pathway: Modeling and analysis. In Ganguly, N., Deutsch, A., Mukherjee, A., and Bellomo, N., editors, *Dynamics On and Of Complex Networks*, Modeling and Simulation in Science, Engineering, & Technology, pages 19–33. Birkhauser Boston. 10.1007/978-0-8176-4751-3_2.

Ciocchetta, F. and Hillston, J. (2009). Bio-PEPA: A framework for the modelling and analysis of biological systems. *Theoretical Computer Science*, 410(33-34):3065 – 3084. ISSN 0304-3975.

Clarke, E. and Emerson, E. (1982). Design and synthesis of synchronization skeletons using branching time temporal logic. In Kozen, D., editor, *Logics of Programs*, volume 131 of *Lecture Notes in Computer Science*, pages 52–71. Springer Berlin / Heidelberg. 10.1007/BFb0025774.

Clarke, E., Emerson, E., Jha, S., and Sistla, A. (1998). Symmetry reductions in model checking. In Hu, A. and Vardi, M., editors, *Computer Aided Verification*, volume 1427 of *Lecture Notes in Computer Science*, pages 147–158. Springer Berlin Heidelberg.

Clarke, E., Grumberg, O., and Peled, D. (1999). *Model Checking*. MIT Press.

Clarke, E. M., Emerson, E. A., and Sistla, A. P. (1986). Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Trans. Program. Lang. Syst.*, 8:244--263. ISSN 0164-0925.

Clarke, E. M., Faeder, J. R., Langmead, C. J., Harris, L. A., Jha, S. K., and Legay, A. (2008). Statistical model checking in BioLab: Applications to the automated analysis of T-Cell receptor signaling pathway. In *Proceedings of the 6th International Conference on Computational Methods in Systems Biology*, CMSB '08, pages 231--250, Berlin, Heidelberg. Springer-Verlag.

Crepalde, M. (2011). Modelagem e análise de sistemas de transporte de Íons em membranas celulares usando verificação de modelos. Master's thesis, Universidade Federal de Minas Gerais.

Dickinson (2013). Thinking allowed - cell membrane structure. `http://dickinsonn.ism-online.org/2009/11/16/cell-membrane-structure/`.

DiMasi, J. A. and Grabowski, H. G. (2012). *The Oxford Handbook of The Economics of the Biopharmaceutical Industry*, chapter R&D Costs and returns to new drug development: a review of the evidence, pages 21--46. Oxford University Press, Oxford, UK.

DiMasi, J. A., Hansen, R. W., and Grabowski, H. G. (2003). The price of innovation: new estimates of drug development costs. *J Health Econ*, 22(2):151--185.

Dworakowska, B. and Dolowy, K. (2000). Ion channels-related diseases. *Acta Biochim. Pol.*, 47(3):685--703.

Emerson, E. A. and Clarke, E. M. (1980). Characterizing correctness properties of parallel programs using fixpoints. In *Proceedings of the 7th Colloquium on Automata, Languages and Programming*, pages 169--181, London, UK. Springer-Verlag.

Fages, F. (2006). From Syntax to Semantics in Systems Biology Towards Automated Reasoning Tools. In *Transactions on Computational Systems Biology IV*, pages 68--70. Springer-Verlag.

Fages, F. and Soliman, S. (2008). Formal cell biology in BIOCHAM. In *Proceedings of the Formal methods for the design of computer, communication, and software systems 8th international conference on Formal methods for computational systems biology*, SFM'08, pages 54--80, Berlin, Heidelberg. Springer-Verlag.

Ferreira, B., Braz, F. A., and Campos, S. V. (2012). A probabilistic model checking approach to investigate vehicular networks. In *Proceedings of the 15th Brazilian Symposium on Formal Methods (SBMF)*. short paper.

Filho, F. (2007). *Algoritmos Numéricos*. LTC. ISBN 9788521615378.

Fujita, M., McGeer, P., and Yang, J.-Y. (1997). Multi-terminal binary decision diagrams: An efficient data structure for matrix representation. *Formal Methods in System Design*, 10:149–169. ISSN 0925-9856. 10.1023/A:1008647823331.

Gay, S., Soliman, S., and Fages, F. (2010). A graphical method for reducing and relating models in systems biology. *Bioinformatics (Oxford, England)*, 26(18):i575--i581. ISSN 1367-4811.

GEXF (2013). Graph exchange xml format. `http://www.gexf.net/`.

Gillespie, D. T. (1977). Exact stochastic simulation of coupled chemical reactions. *The Journal of Physical Chemistry*, 81(25):2340--2361.

Godefroid, P., Peled, D., and Staskauskas, M. (1996). Using partial-order methods in the formal validation of industrial concurrent programs. *Software Engineering, IEEE Transactions on*, 22(7):496 –507. ISSN 0098-5589.

GraphML (2013). Graph modeling language. `http://graphml.graphdrawing.org/`.

GraphViz (2013). Graph visualization software. `http://www.graphviz.org/`.

Hahn, E. M., Hermanns, H., Wachter, B., and Zhang, L. (2009). INFAMY: An infinite-state Markov model checker. In *Proc. 21st International Conference on Computer Aided Verification (CAV'09)*, volume 5643 of *LNCS*, pages 641–647. Springer.

Hahn, E. M., Hermanns, H., Wachter, B., and Zhang, L. (2010). PARAM: A model checker for parametric Markov models. In *Proc. 22nd International Conference on Computer Aided Verification (CAV'10)*, volume 6174 of *LNCS*, pages 660–664. Springer.

Heath, J., Kwiatkowska, M., Norman, G., Parker, D., and Tymchyshyn, O. (2008). Probabilistic model checking of complex biological pathways. *Theor. Comput. Sci.*, 391(3):239--257. ISSN 0304-3975.

Heiner, M. and Sriram, K. (2010). Structural analysis to determine the core of hypoxia response network. *PLoS One*, 5(1):e8600. ISSN 1932-6203.

Hernández, J. and Chifflet, S. (2000). Electrogenic properties of the sodium pump in a dynamic model of membrane transport. *Journal of Membrane Biology*, 176:41–52. ISSN 0022-2631. 10.1007/s00232001074.

Hille, B. (2001). *Ion Channels of Excitable Membranes*. Sinauer Associates, 3 edition. ISBN 0878933212.

Johnny Vincent C. (2013). Palythoa toxica. `http://reefbuilders.com/2012/03/04/palytoxin/#more-55617`.

Katoen, J.-P., Zapreev, I. S., Hahn, E. M., Hermanns, H., and Jansen, D. N. (2011). The ins and outs of the probabilistic model checker MRMC. *Perform. Eval.*, 68(2):90--104. ISSN 0166-5316.

Kwiatkowska, M., Norman, G., and Parker, D. (2004). Probabilistic symbolic model checking with PRISM: A hybrid approach. *International Journal on Software Tools for Technology Transfer (STTT)*, 6(2):128--142.

Kwiatkowska, M., Norman, G., and Parker, D. (2007). Stochastic model checking. In Bernardo, M. and Hillston, J., editors, *Formal Methods for the Design of Computer, Communication and Software Systems: Performance Evaluation (SFM'07)*, volume 4486 of *LNCS (Tutorial Volume)*, pages 220–270. Springer.

Kwiatkowska, M., Norman, G., and Parker, D. (2008). Using probabilistic model checking in systems biology. *SIGMETRICS Perform. Eval. Rev.*, 35(4):14--21. ISSN 0163-5999.

Kwiatkowska, M., Norman, G., and Parker, D. (2009). Quantitative verification techniques for biological processes. In Condon, A., Harel, D., Kok, J. N., Salomaa, A., and Winfree, E., editors, *Algorithmic Bioprocesses*, Natural Computing Series, pages 391–409. Springer Berlin Heidelberg. 10.1007/978-3-540-88869-7_20.

Kwiatkowska, M., Norman, G., and Parker, D. (2010). *Symbolic Systems Biology*, chapter Probabilistic Model Checking for Systems Biology, pages 31--59. Jones and Bartlett, Sudbury, Mass., USA.

Kwiatkowska, M., Norman, G., and Parker, D. (2011). PRISM 4.0: Verification of probabilistic real-time systems. In Gopalakrishnan, G. and Qadeer, S., editors, *Proc. 23rd International Conference on Computer Aided Verification (CAV'11)*, volume 6806 of *LNCS*, pages 585--591. Springer.

Kwiatkowska, M. Z. and Heath, J. K. (2009). Biological pathways as communicating computer systems. *J. Cell. Sci.*, 122(Pt 16):2793--2800.

Laubenbacher, R. and Jarrah, A. S. (2009). Chapter 7 - algebraic models of biochemical networks. In Brand, M. L. J. L., editor, *Methods in Enzymology*, volume 467 of *Methods in Enzymology*, pages 163 – 196. Academic Press.

Lehninger, A., Nelson, D. L., and Cox, M. M. (2008). *Lehninger Principles of Biochemistry*. W. H. Freeman, 5th edition.

Mateescu, R., Monteiro, P., Dumas, E., and de Jong, H. (2011). Ctrl: Extension of ctl with regular expressions and fairness operators to verify genetic regulatory networks. *Theoretical Computer Science*, 412(26):2854–2883.

McMillan, K. L. (1992). *Symbolic model checking: an approach to the state explosion problem*. PhD thesis, Carnegie Mellon University.

Monteiro, P., Ropers, D., Mateescu, R., Freitas, A., and de Jong, H. (2008). Temporal logic patterns for querying dynamic models of cellular interaction networks. *Bioinformatics*, 24(16):i227--i233.

Monteiro, P. T., Dumas, E., Besson, B., Mateescu, R., Page, M., Freitas, A. T., and de Jong, H. (2009). A service-oriented architecture for integrating the modeling and formal verification of genetic regulatory networks. *BMC Bioinformatics*, 10(1):450.

Murata, T. (1989). Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77(4):541--580. ISSN 00189219.

Nash, D. B. (2008). PhRMA 2008. *P T*, 33(12):685.

Next Generation Pharmaceutical (2013). Systems biology solution for drug discovery and personalized medicine. `http://www.ngpharma.com/article/Systems-Biology-Solution-for-Drug-Discovery-and-Personalized-Medicine/`.

Oka, C., Cha, C. Y., and Noma, A. (2010). Characterization of the cardiac Na+/K+ pump by development of a comprehensive and mechanistic model. *Journal of Theoretical Biology*, 265(1):68 – 77. ISSN 0022-5193.

OpenStax College (2013a). Biology - eukaryotic cells. `http://cnx.org/content/m44407/latest/?collection=col11448/latest`.

OpenStax College (2013b). Concepts of biology - active transport. `http://cnx.org/content/m45435/latest/?collection=col11487/latest`.

Parker, D. (2002). *Implementation of Symbolic Model Checking for Probabilistic Systems*. PhD thesis, University of Birmingham.

Post, R., Sen, A., and Rosenthal, A. (1965). A phosphorylated intermediate in adenosine triphosphate-dependent sodium and potassium transport across kidney membranes. *J. Biol. Chem*, 240:1437–1445.

Post, R. L., Hegyvary, C., and Kume, S. (1972). Activation by adenosine triphosphate in the phosphorylation kinetics of sodium and potassium ion transport adenosine triphosphatase. *J. Biol. Chem.*, 247(20):6530--6540.

Queille, J. P. and Sifakis, J. (1982). A temporal logic to deal with fairness in transition systems. In *Proceedings of the 23rd Annual Symposium on Foundations of Computer Science*, SFCS '82, pages 217--225, Washington, DC, USA. IEEE Computer Society.

Reactome (2013). Reactome. `http://reactome.org`.

Riedl, M., Schuster, J., and Siegle, M. (2008). Recent extensions to the stochastic process algebra tool CASPA. In *Proc. 5th International Conference on Quantitative Evaluation of Systems (QEST'08)*, pages 113–114. IEEE CS Press.

Risken, H. (1984). *The Fokker-Planck Equation: Methods of Solution and Applications*. World Publishing Corporation. ISBN 9787506214223.

Rodrigues, A. M., Almeida, A.-C. G., Infantosi, A. F., Teixeira, H. Z., and Duarte, M. A. (2009a). Investigating the potassium interactions with the palytoxin induced channels in Na+/K+ pump. *Computational Biology and Chemistry*, 33(1):14 – 21. ISSN 1476-9271.

Rodrigues, A. M., Almeida, A.-C. G., and Infantosi, A. F. C. (2008a). Effect of palytoxin on the sodium-potassium pump: model and simulation. *Physical Biology*, 5(3):036005.

Rodrigues, A. M., Almeida, A.-C. G., Infantosi, A. F. C., Teixeira, H. Z., and Duarte, M. A. (2008b). Model and simulation of Na^+/K^+ pump phosphorylation in the presence of palytoxin. *Comput. Biol. Chem.*, 32(1):5--16. ISSN 1476-9271.

Rodrigues, A. M., Infantosi, A. F., and de Almeida, A. C. (2009b). Palytoxin and the sodium/potassium pump–phosphorylation and potassium interaction. *Phys Biol*, 6(3):036010.

Sauro, H., Uhrmacher, A., Harel, D., Hucka, M., Kwiatkowska, M., Mendes, P., Shaffer, C., Strömback, L., and Tyson, J. (2006). Challenges for modeling and simulation methods in systems biology. In Perrone, L., Wieland, F., Liu, J., Lawson, B., Nicol, D., and Fujimoto, R., editors, *Proc. 2006 Winter Simulation Conference*, pages 1720–1730. IEEE.

SBML (2013). Systems biology markup language. `http://sbml.org/`.

Song, M. A. J. (2004). *The UML-CAFE: an Environment to Specify and Verify Tran-sactional Systems*. PhD thesis, Universidade Federal de Minas Gerais.

Tosteson, M., Thomas, J., Arnadottir, J., and Tosteson, D. (2003). Effects of palytoxin on cation occlusion and phosphorylation of the (Na^+,K^+)-ATPase. *Journal of Membrane Biology*, 192:181–189. ISSN 0022-2631. 10.1007/s00232-002-1074-9.

Tribastone, M. (2007). The PEPA plug-in project. In *Proc. 4th International Conference on Quantitative Evaluation of Systems (QEST'07)*, pages 53–54. IEEE Computer Society.

WetWebMedia Forum (2013). Palytoxin in zoanthids and palythoas.... `http://wetwebmediaforum.com/showthread.php?2268-Palytoxin-in-Zoanthids-and-Palythoas....`

Yamada, K. and Inagaki, N. (2002). ATP-sensitive K^+ channels in the brain: sensors of hypoxic conditions. *News Physiol. Sci.*, 17:127--130.

Younes, H. (2005). Ymer: A statistical model checker. In Etessami, K. and Rajamani, S., editors, *Computer Aided Verification*, volume 3576 of *Lecture Notes in Computer Science*, pages 171–179. Springer Berlin / Heidelberg. 10.1007/11513988_43.

Younes, H., Kwiatkowska, M., Norman, G., and Parker, D. (2006). Numerical vs. statistical probabilistic model checking. *International Journal on Software Tools for Technology Transfer (STTT)*, 8(3):216–228.