DESIGN DE CIRCUITOS LÓGICOS BASEADOS EM DNA VISANDO A SÍNTESE DE SISTEMAS COMPUTACIONAIS

DANIEL KNEIPP DE SÁ VIEIRA

DESIGN DE CIRCUITOS LÓGICOS BASEADOS EM DNA VISANDO A SÍNTESE DE SISTEMAS COMPUTACIONAIS

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: Omar Paranaíba Vilela Neto
Coorientador: Marcos Viero Guterres

Belo Horizonte
Junho de 2018

Vieira, Daniel Kneipp de Sá

V658d *Design* de circuitos lógicos baseados em DNA visando a síntese de sistemas computacionais / Daniel Kneipp de Sá Vieira. — Belo Horizonte, 2018
xxi, 107 f. : il. ; 29cm.

Dissertação (mestrado) — Universidade Federal de Minas Gerais — Departamento de Ciência da Computação.

Orientador: Omar Paranaíba Vilela Neto
Coorientador: Marcos Viero Guterres

1. Computação — Teses. 2. Nanocomputação.
3. Computação com DNA. I. Orientador.
II. Coorientador. III. Título.

CDU 519.6*82(043)

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

FOLHA DE APROVAÇÃO

Design de Circuitos Lógicos Baseados em DNA Visando a Síntese de Sistemas Computacionais

DANIEL KNEIPP DE SÁ VIEIRA

Dissertação defendida e aprovada pela banca examinadora constituída pelos Senhores:

PROF. OMAR PARANAÍBA VILELA NETO - Orientador
Departamento de Ciência da Computação - UFMG

DR. MARCOS VIERO GUTERRES - Coorientador
CDTN - Centro de Desenvolvimento da Tecnologia Nuclear

PROFA. LIZA FIGUEIREDO FELICORI VILELA
Departamento de Bioquímica e Imunologia - UFMG

PROFA. GISELE LOBO PAPPA
Departamento de Ciência da Computação - UFMG

Belo Horizonte, 13 de junho de 2018.

Agradecimentos

Em primeiro lugar, gostaria de agradecer aos meus pais, Érica e Liomar, que sempre estiveram do meu lado e me suportaram de todas as formas possíveis durante toda minha vida. Às minhas avós, Elza e Ivone, pela sabedoria e apoio, eu agradeço. Sem vocês nada disso seria possível.

Aos amigos formados nesta jornada, em especial ao Geanderson, os quais me mostraram que é possível formar amizades mesmo em tempos difíceis. Aos amigos adquiridos na graduação, em especial ao Fredy e ao Rodrigo, sempre com o *"Rocketwatch"/"Overleague"* das madrugadas que ajudaram a me manter nos trilhos quando eu mais queria descarrilar. Ao Leandro ("nós vamos trampar juntos") Pessoa, obrigado pelo interesse no meu trabalho (significa muito para mim). Aos amigos de longa data Wagner e Luiz, "topando" em casa para dar aquela descontraída que eu nunca pedia, mas sempre queria hehe.

Ao laboratório NanoComp, que me proporcionou um ambiente calmo e agradável para conduzir a pesquisa, além de me permitir conhecer pessoas magníficas trabalhando nos temas mais variados e fascinantes.

Às empresas que trabalhei, em especial à Invent Vision e ao Luiz (CTO), pelo tempo e conselhos dados, os quais contribuíram para meu crescimento profissional e pessoal. Jamais esquecerei o apoio.

Um agradecimento especial ao professor Omar, com sua orientação imprescindível me apontando na direção correta e provendo os recursos necessários para conclusão deste trabalho. Também ao meu coorientador Marcos, sem seu conhecimento profundo em Química e áreas relacionadas este trabalho seria apenas o título.

"Do what you can, with what you have, where you are."
(Theodore Roosevelt)

Resumo

Computação com DNA é um dos ramos da computação molecular que vem sendo usado nos últimos anos para desenvolver dispositivos implantados *in-vitro* e *in-vivo* ou até mesmo substituir a tecnologia CMOS em algumas aplicações. Utilizando a técnica conhecida como reação de deslocamento de cadeia, fragmentos de DNA são combinados e manipulados de forma programática seguindo uma certa lógica. Este modelo de computação permite implementar fisicamente comportamentos teorizados por meio da linguagem das redes de reações químicas, a qual descreve textualmente a cinética de uma rede de reações (CRNs). Com uma linguagem para descrever dispositivos baseados em DNA, modelagens de estruturas base e elementos de baixo nível para desenvolvimento de circuitos são definidos, aumentando consideravelmente a complexidade dos dispositivos e, por consequência, se torna mais difícil simula-los. Neste trabalho um pacote para simulação de circuitos de DNA de larga escala é apresentado. Chamado de DNAr, este pacote foi desenvolvido para simular tanto CRNs formais (redes de reações teóricas) quanto as CRNs de DNA (reações que representam interações entre moléculas de DNA). Este pacote também é capaz de transformar uma CRN formal em uma CRN de DNA sem intervenção do usuário.

CRNs clássicas (e.g.: Oscilador Lotka-Volterra, *Consensus*, etc.) foram utilizadas para validar o simulador. Dois estudos de caso foram desenvolvidos de forma a validar e exemplificar a capacidade de expansão do DNAr. Os estudos de caso se baseiam em duas abordagens de construção de portas lógicas químicas. Uma modela portas lógicas diretamente e a outra possui neurônios químicos como elemento base. As duas abordagens foram implementadas como extensões do DNAr e podem ser utilizadas para o desenvolvimento de outros circuitos. Um circuito lógico que detecta células cancerígenas foi utilizado como circuito alvo a ser implementado pelas duas abordagens. Os resultados mostraram que as duas abordagens alcançaram o comportamento desejado. O fato de se ter gerado CRNs com mais de 200 reações (podendo chegar próximo de 400) indica a necessidade do uso de uma ferramenta como o DNAr.

Abstract

DNA computing is one of the branches of molecular computing that has been used in recent years to develop devices implanted *in-vitro*, *in-vivo* or even replace the CMOS technology in some applications. Using a technique known as Strand Replacement Reaction. DNA strands are combined and manipulated in programmatic a fashion following a certain logic. This computing model allows us to physically implement theoretical behaviors specified through the Chemical Reaction Networks (CRNs) language, which textually describes the reaction network kinetics. Building blocks for circuitry development are defined with a language used to describe DNA-based devices, increasing considerably the complexity of the device and, consequently, simulating these devices becomes difficult. In this dissertation a simulation package for large scale DNA-based devices is presented. It is called DNAr and it was developed to simulate formal CRNs (theoretical reaction networks) and DNA-based CRNs (reactions that represent interactions between DNA molecules). This package is also capable of transforming a formal CRN to a DNA-based CRN without user interference.

Well-known CRNs (e.g.: Lotka-Volterra oscillator, Consensus, etc.) were used to validate the simulator. Two case studies were conducted in order to validate and exemplify the DNAr expansion capabilities. These case studies are based on two approaches for constructing chemical logic gates. One of them models a logic gate directly while the other uses chemical neurons as its building block. Both approaches were implemented as DNAr extensions and they can be used to design other circuits. A logic circuit capable of detecting cancer cells was used as the target circuit to be implemented through these approaches. The results show that both approaches achieved the desired behavior. The fact that CRNs with 200 to 400 reactions were generated indicates the need of a tool such as DNAr.

Lista de Figuras

1.1 Nanorobô transportador baseado em DNA 3
1.2 Computador molecular 4
1.3 Esquema de um sistema cognitivo 5

2.1 Comportamento de $A \longrightarrow B$. 13
2.2 Comportamentos da CRN Lotka-Volterra. 19
2.3 Comportamento da CRN *Consensus* 21
2.4 Representação gráfica da reação $A \longrightarrow B$ simulada. 25
2.5 Comportamento das portas AND e OR. 29

3.1 Estrutura do DNA 32
3.2 Representações do DNA 33
3.3 Exemplo de reação de deslocamento de cadeia 34
3.4 Configuração inicial do exemplo de reação em cascata 35
3.5 Segunda etapa de uma reação em cascata 36
3.6 Produtos gerados pela reação em cascata 36
3.7 Porta lógica que as reações em cascata representam 37
3.8 Fragmentos das espécies da reação unimolecular 39
3.9 Fragmentos das estruturas auxiliares da reação unimolecular 39
3.10 Primeira etapa de uma reação unimolecular em DNA 39
3.11 Segunda etapa de uma reação unimolecular em DNA 40
3.12 Primeira reação da primeira etapa para reações bimoleculares 41
3.13 Segunda reação da primeira etapa para reações bimoleculares 42
3.14 Reação da segunda etapa para reações bimoleculares 42
3.15 Efeito de *buffer* ocorrendo no oscilador Lotka-Volterra 44
3.16 Módulo de *buffer* 45
3.17 Efeito de *buffer* corrigido no oscilador Lotka-Volterra 46
3.18 Interface principal do Visual DSD 47

3.19 Especificando moléculas de DNA no Visual DSD 48
3.20 Exemplo de reação improdutiva 49

4.1 Exemplo de visualização gráfica no DNAr 55
4.2 Reação unimolecuar no Visual DSD 62
4.3 Reações simuladas a partir do *script* 62
4.4 Sequência de nucleotídeos dos domínios 63
4.5 Comportamento errôneo do Lotka-Volterra 64
4.6 Comportamento corrigido do Lotka-Volterra 64
4.7 Fluxograma de trabalho com o DNAr e Visual DSD 66

5.1 Modelo de neurônio McCulloch–Pitts 72
5.2 Neurônio como uma porta `NOT` 73
5.3 Comportamento de um neurônio 77
5.4 Estado de convergência do neurônio em função da entrada 78
5.5 Porta `AND` formada por neurônios 84
5.6 Comportamento das portas lógicas `AND` e `OR` 85
5.7 Comportamento de uma MG. 86
5.8 Circuito lógico 87
5.9 Resultados dos testes da CRN de DNA 89
5.10 Comportamento da CRN de portas lógicas 93

Lista de Tabelas

2.1 Comportamento de $A \longrightarrow B$. 12
2.2 Exemplos de funções de velocidade para diferentes reações 15
2.3 Tipos de reações 16

4.1 Reações que podem ser exportadas para Visual DSD 59
4.2 Tabela comparativa do DNAr com o Visual DSD 65
4.3 Principais funções do DNAr 69

5.1 Constantes de velocidade das reações do neurônio 76
5.2 Concentrações inicias das espécies do neurônio 77
5.3 Configurações em que a CRN de DNA foi simulada 89
5.4 NRMSE comparando o comportamento da CRN formal baseada em neurônios com a CRN de DNA 90
5.5 NRMSE comparando o comportamento da CRN formal baseada na abordagem Jiang et al. [2013] com a CRN de DNA 93

Sumário

Agradecimentos **vii**

Resumo **xi**

Abstract **xiii**

Lista de Figuras **xv**

Lista de Tabelas **xvii**

1 Introdução **1**

1.1 Aplicações com DNA 2

1.1.1 Computação 2

1.1.2 Robótica 3

1.1.3 Armazenamento de dados 3

1.1.4 Interação com organismos vivos 4

1.2 Descrição e Objetivos do Trabalho 5

1.3 Contribuições 6

1.4 Organização da Dissertação 6

2 Redes de Reações Químicas **9**

2.1 Reação Química 9

2.1.1 Coeficientes estequiométricos 10

2.1.2 Reações reversíveis 10

2.1.3 Concentrações das espécies 11

2.1.4 Velocidade das reações químicas 12

2.1.5 Modelagem de reações químicas formais 15

2.2 Reações Químicas Acopladas 17

2.2.1 Oscilador Lotka–Volterra 18

2.2.2 *Consensus* . . . 20
2.3 Simulando Reações Químicas no `R` . . . 21
2.3.1 Exemplo: reação unimolecular simples . . . 22
2.4 Lógica Booleana com Reações Químicas . . . 26
2.4.1 Representação de um bit . . . 26
2.4.2 Implementando uma porta `AND` . . . 27
2.4.3 Porta `OR` . . . 28

3 Implementando Reações Químicas com DNA 31
3.1 Estrutura do DNA . . . 32
3.2 Reação de Deslocamento de Cadeia . . . 33
3.2.1 Cascateando reações . . . 35
3.3 Transformando Reações Químicas Formais em Reações de DNA . . . 37
3.3.1 Reações unimoleculares . . . 38
3.3.2 Reações bimoleculares . . . 41
3.3.3 Efeito de *buffer* . . . 43
3.4 Microsoft Visual DSD . . . 47
3.4.1 Linguagem de especificação . . . 48
3.4.2 Compilação . . . 49
3.4.3 Simulação . . . 50

4 DNAr – Simulando Redes de Reações de Larga Escala 51
4.1 Simulando reações formais . . . 53
4.2 Modelagem vetorial das EDOs . . . 55
4.3 Simulando CRNs baseadas em DNA . . . 57
4.4 Exportando CRNs para o Microsoft Visual DSD . . . 58
4.4.1 Limitações . . . 63
4.5 Comparação entre DNAr e o Visual DSD . . . 63
4.5.1 Fluxo de trabalho com o DNAr e o Visual DSD . . . 65
4.6 Estrutura detalhada do pacote . . . 65
4.6.1 Principais funções . . . 68

5 Estudo de Caso: Neurônio Químico 71
5.1 Modelo McCulloch–Pitts . . . 72
5.1.1 Construindo Portas Lógicas . . . 73
5.2 Modelagem Química do Neurônio . . . 74
5.2.1 Conectando neurônios . . . 78
5.2.2 Construindo Portas Lógicas . . . 80

5.3 Estudo de Caso: Detecção de Células Específicas de Câncer 86
5.3.1 Comparação com outra abordagem 90

6 Conclusões e Trabalhos Futuros **95**
6.1 Trabalhos Futuros 98

Referências Bibliográficas **101**

Capítulo 1

Introdução

Várias áreas do conhecimento utilizam a computação como meio para alavancarem seus avanços. Para suprir a necessidade existente, os recursos que compõem o computador moderno estão atingindo seus limites físicos. É o caso da arquitetura baseada em silício que está perto do seu limite de miniaturização [Cavin et al., 2012]. Uma das alternativas que vem sendo pesquisadas para serem utilizadas como forma de fazer computação é chamada de computação molecular.

Essa área de pesquisa foi iniciada com o trabalho de Adleman [1994] que demonstra como moléculas de DNA (*deoxyribonucleic acid* – ácido desoxirribonucleico) podem ser utilizadas para fazer computação e resolver problemas combinatórios. No caso, o problema clássico conhecido como caminho Hamiltoniano direcionado foi utilizado como estudo de caso.

O potencial desta tecnologia é apresentado pelo autor mostrando que se a ligação entre duas moléculas de DNA for considerada como uma operação, seria inteiramente plausível construir um sistema que faria 10^{20} operações apenas na etapa inicial da resolução do problema. Os supercomputadores da época conseguiam alcançar algo em torno de 10^{12} operações por segundo. Além disso, o sistema baseado em DNA consegue executar 2×10^{19} operações com 1 Joule, notavelmente mais eficiente que um supercomputador da época (10^9 operações por Joule). Isto representa uma computação massivamente paralela e energeticamente eficiente. A capacidade de armazenamento de 1 bit por nanômetro cubico também é destacada.

Algo importante de salientar é que, por mais que a quantidade de interações simultâneas entre moléculas de DNA seja consideravelmente alta, DNA não é feito para substituir o silício completamente. Reações entre moléculas podem demorar horas, e a conclusão da execução de um circuito, dias. Portanto, o uso de DNA é mais apropriado para a solução de problemas complexos altamente paralelizáveis. Além

disso, um dos pontos fortes destacados sobre circuitos e dispositivos baseados em DNA é sua capacidade de interfaceamento com organismos vivos [Chen et al., 2015], não sua eficiência em computação.

Trabalhos teóricos e práticos foram apresentados ao longo dos anos utilizando o DNA nas mais diversas aplicações, como computação, robótica e armazenamento de dados, por exemplo. Alguns destes trabalhos são resumidos na seguinte seção.

1.1 Aplicações com DNA

Com o custo da preparação e purificação de oligonucleotídeos caindo exponencialmente [Carlson, 2009], o uso de DNA (ácido desoxirribonucleico) como matéria prima para o desenvolvimento dos mais variados mecanismos se tornou bastante interessante. Isso tem levado a usos não biológicos do DNA como a criação de estruturas auto-organizáveis e computação molecular, formando as bases do campo da nanotecnologia baseada em DNA [Zhang & Seelig, 2011].

Com o avanço da tecnologia baseada em DNA, hoje podem ser feitos dispositivos auto-organizáveis (prática conhecida como *DNA origami*) capazes de fazer computação massivamente paralela, armazenar informação com alta densidade e interagir com organismos vivos, motivando a pesquisa e uso desta tecnologia para várias aplicações. Algumas destas são descritas nas seções a seguir.

1.1.1 Computação

Qian et al. [2011] mostra como construir uma rede neural (treinada *in silico* – simulações computacionais) baseada em DNA utilizando reações de deslocamento de cadeia (*strand-displacement reactions* – reações que ocorrem entre moléculas de DNA com o propósito de combinar e modificar sua estrutura) [Zhang & Seelig, 2011].

Com a arquitetura da rede sendo construída com uma porta lógica projetada para proporcionar a síntese de circuitos em larga escala baseados em DNA [Qian & Winfree, 2011b], se torna possível construir um sistema com várias camadas operando em cascata de que opera de forma robusta [Qian & Winfree, 2011a].

De acordo com Qian et al. [2011], seus resultados sugerem que reações de deslocamento de cadeia em cascata podem ser utilizadas para desenvolver sistemas químicos autônomos com capacidade de reconhecer padrões e tomar decisões.

1.1.2 Robótica

DNA origami é a prática de criar nanoestruturas em duas ou três dimensões utilizando interações entre bases complementares. Douglas et al. [2012] aplica este método para descrever um nanorobô autônomo baseado em DNA capaz de transportar cargas moleculares para células e alterar sua própria forma para entregar as mesmas ao destino. A Figura 1.1 mostra o nanorobô em diferentes configurações.

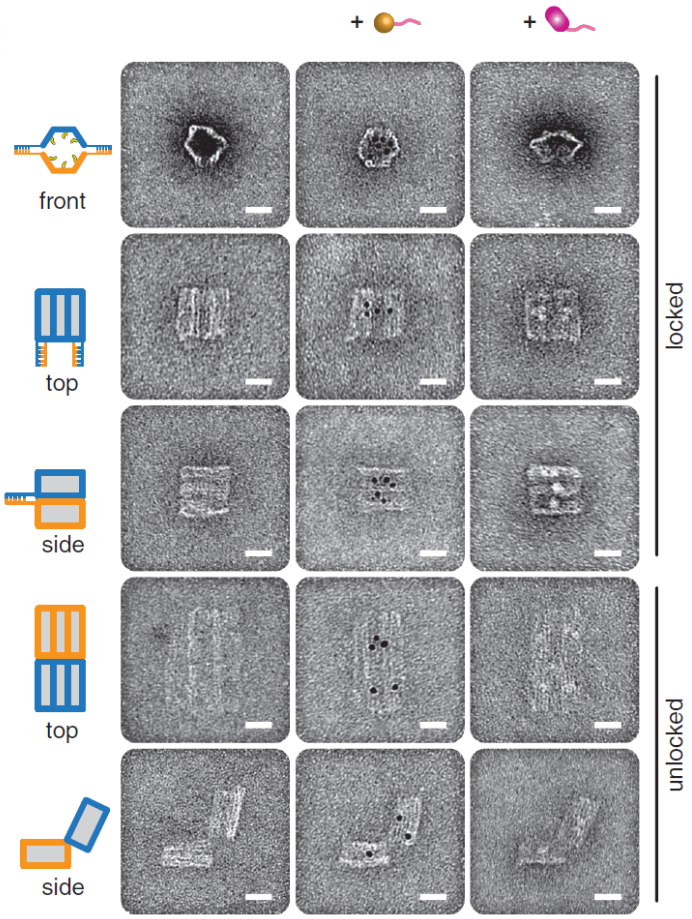


Figura 1.1: Imagens de um microscópio eletrônico de transmissão mostrando nanorobôs nas configurações aberta (*unlocked*) e fechada (*locked*). Na coluna da esquerda o nanorobô está vazio e nas colunas do meio e da direita o nanorobô está carregado com 5 nm de partículas de ouro e fragmentos de Fab (*fragment antigen-binding*), respectivamente. Escala de 20 nm por barra branca contida em cada imagem [Douglas et al., 2012].

1.1.3 Armazenamento de dados

Já o trabalho de Mayer et al. [2016] argumenta que DNA combina uma longevidade notável para preservar informação com capacidade de armazenamento de alta densidade.

É demonstrado a capacidade do DNA de conter dados armazenando simultaneamente múltiplas camadas de informação (3 imagens) em um único modelo de DNA (*DNA template*).

1.1.4 Interação com organismos vivos

Benenson et al. [2004] mostra o potencial do uso de DNA para o diagnóstico e até mesmo tratamento médico, apresentando um computador molecular autônomo que, ao menos *in vitro*, é capaz de analisar níveis de espécies mRNA (RNA mensageiro) e gerar como saída moléculas capazes de afetar os níveis de expressões de genes. A Figura 1.2 mostra uma representação modular do sistema proposto.

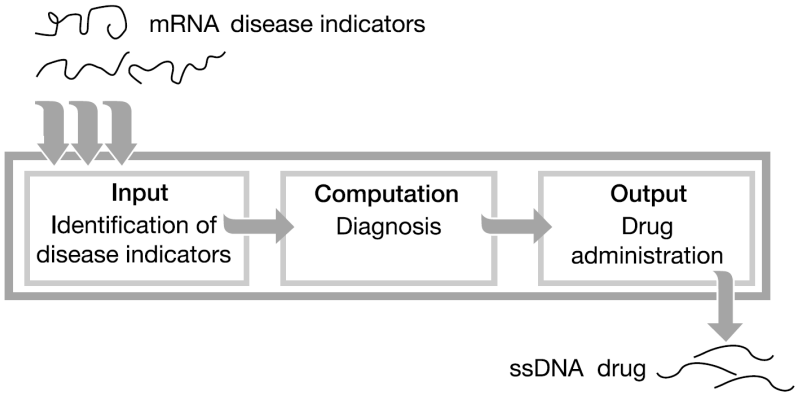


Figura 1.2: Organização modular do computador molecular [Benenson et al., 2004]. O módulo de entrada recebe específicos níveis de espécies de mRNA e inicia um processo em cascata que "diagnostica" a doença – no caso, um modelo simplificado de câncer. Em caso de diagnóstico positivo, é produzido um fármaco baseado em DNA de cadeia simples.

O Desenvolvimento de dispositivos com interface cérebro-máquina também pode servir de campo de aplicação para a nanotecnologia com DNA. Arnon et al. [2016] mostram um sistema que permite um humano controlar nanorobôs que estão dentro de um ser vivo por meio atividade cerebral.

De acordo com os autores dessa tecnologia, ligar ou desligar em tempo real uma molécula bioativa devido ao estado cognitivo de um agente possui implicações em potencial no tratamento de desordens tais como a esquizofrenia, depressão e déficit de atenção. O esquema da Figura 1.3 descreve o funcionamento do sistema superficialmente.

Estes são apenas alguns exemplos de aplicações que estão surgindo utilizando nanotecnologia com o DNA, mostrando a importância e relevância que este campo de

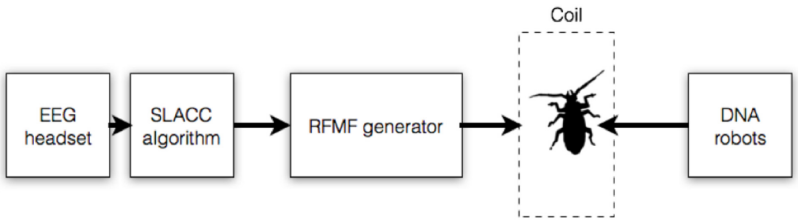


Figura 1.3: Por meio de eletroencefalografia (*EEG headset*), é registrada a atividade cerebral e esta é enviada para ser monitorada pelo algoritmo SLACC (classificador que descrimina atividade de inatividade cerebral). Caso este classificador retorne positivo para atividade cerebral, um campo magnético induzido por frequência (RFMF, que significa *frequency-induced electromagnetic field*) é ativado em uma bobina indutora. O animal está dentro da bobina com bilhões de nanorobôs (baseados em *DNA origami*). A existência do campo magnético acarreta em um aquecimento local nestes nanorobôs e, consequentemente, na ativação reversível dos mesmos (o que acarreta na liberação de uma carga transportada pelos nanorobôs) [Arnon et al., 2016].

pesquisa está obtendo com o passar dos anos e, portanto, motivando o desenvolvimento deste trabalho.

1.2 Descrição e Objetivos do Trabalho

O foco principal deste trabalho é explorar o desenvolvimento de circuitos lógicos para fazer computação implementados utilizando reações de deslocamento de cadeias de DNA. Para descrever tais circuitos, o arcabouço teórico das redes de reações químicas é utilizado.

Chen et al. [2013] deixa claro que o formalismo matemático das redes de reações químicas (CRNs) pode ser usado como linguagem de programação para projetar comportamentos complexos, logo, dispositivos baseados em DNA já podem ser formalmente modelados se embasando em uma teoria amplamente estudada [Soloveichik et al., 2010].

O objetivo principal desta dissertação é pesquisar e desenvolver circuitos lógicos baseados em DNA especificados por meio da linguagem das reações químicas, explorando diferentes abordagens de se implementar tais circuitos utilizando as reações de deslocamento de cadeia como elemento base para computação.

1.3 Contribuições

Com o intuito de facilitar a simulação de redes de reações químicas e permitir a simulação de circuitos de larga escala, um simulador de reações químicas foi desenvolvido. Este simulador, chamado de DNAr, permite que o comportamento de reações químicas sejam simuladas apenas especificando textualmente as reações existentes. Isto facilita bastante o processo de simulação, já que não se faz necessário programar a modelagem matemática que descreve o comportamento das reações. Além disso, este simulador é capaz de gerar um conjunto de reações entre fragmentos de DNA que emulam o comportamento de uma CRN alvo desejada de forma automática.

Um estudo de caso foi conduzido utilizando este simulador. O objetivo foi definir CRNs capazes de computar um circuito lógico booleano que classifica células como cancerígenas. Para isso, duas abordagens de construção de portas lógicas com reações químicas foram utilizadas. A implementação em DNA das CRNs foi gerada e simulada com o DNAr, obtendo-se resultados satisfatórios nas duas abordagens. A utilização do simulador desenvolvido foi necessária tendo em vista que os circuitos gerados são compostos por centenas de reações e seria inviável escrever manualmente o código necessário para simulá-las. Com o DNAr foi possível definir os circuitos programaticamente por meio de funções simples implementadas de forma modular em relação às funções de simulação, demonstrando o potencial de expansão do pacote de simulação.

1.4 Organização da Dissertação

A dissertação está divida nos capítulos descritos a seguir:

- **Capítulo 2:** apresenta os fundamentos teóricos da cinética química, necessários para o entendimento de como as reações químicas são modeladas e simuladas;
- **Capítulo 3:** mostra como utilizar reações entre moléculas de DNA para reproduzir reações químicas teóricas;
- **Capítulo 4:** descreve em detalhes o pacote DNAr, o qual contém o simulador e outras ferramentas que auxiliam o pesquisador na tarefa de simulação de circuitos químicos e os baseados em DNA;
- **Capítulo 5:** é apresentado um estudo de caso da utilização do DNAr, o qual são especificados circuitos lógicos para a detecção de células cancerígenas específicas utilizando duas abordagens (neurônios químicos e baseada em lógica booleana);

- **Capítulo 6:** são feitas conclusões sobre os resultados obtidos e sugestões para trabalhos futuros.

Capítulo 2

Redes de Reações Químicas

Neste capítulo é apresentado o fundamental teórico para o entendimento de como reações químicas são descritas e simuladas. É mostrado como estas reações podem ser combinadas em redes de reações (CRNs – *Chemical Reaction Networks*) para se obter comportamentos mais complexos. Alguns exemplos de CRNs são mostradas como exemplos para fins de entendimento. Por fim, é demonstrado como a linguagem das reações químicas pode ser utilizada para desempenhar operações de lógica booleana.

2.1 Reação Química

Baseado em Kotz & Treichel [2017], uma reação química é um processo em que uma ou mais substâncias são transformadas em um outro conjunto de substâncias. Uma substância pode ser simples (um elemento químico) ou composta (combinação de elementos). Neste trabalho, uma substância é referenciada por espécie.

Toda reação química é definida por um conjunto de espécies (substâncias) e uma constante de velocidade (descrita posteriormente). Uma reação com um conjunto de espécies $\mathcal{S} = \{A, B\}$, em que A se transforma em B, e uma constante de velocidade k é descrita com a seguinte notação:

$$A \xrightarrow{k} B. \tag{2.1}$$

As espécies podem ser categorizadas como reagentes ou produtos. Reagentes são as espécies transformadas, produtos são as espécies geradas pela transformação. Ou seja, na equação 2.1, A é o reagente e B é o produto. k expressa a velocidade em que a reação ocorre. Quanto maior o valor de k, mais rápido A se transforma em B.

A reação (2.2) mostra a interação entre duas espécies distintas A e B que gera a

espécie C como produto. Não confunda o símbolo de + entre os reagentes com uma soma. Este símbolo representa a combinação de duas espécies.

$$A + B \xrightarrow{k} C. \tag{2.2}$$

Reações que geram mais de um produto são descritas como:

$$A \xrightarrow{k} B + C, \tag{2.3}$$

em que A gera B e C. O símbolo + entre os produtos indica que os dois são gerados pela reação. Isto não significa que os produtos estarão combinados de alguma forma após o fim da reação.

2.1.1 Coeficientes estequiométricos

Coeficientes estequiométricos são valores que precedem o nome das espécies. No seguinte exemplo:

$$2A \xrightarrow{k} B, \tag{2.4}$$

2 é o coeficiente estequiométrico de A e 1 é o da espécie B, o que indica que 2 mols de A reagem para formar 1 mol de B (mol é uma unidade de medida que expressa quantidade de uma substância e é descrito mais a frente na subseção 2.1.3). Estes coeficientes servem para equilibrar o número de átomos dos reagentes com o número de átomos dos produtos (seguindo a lei da conservação de massas [Sterner et al., 2011]).

2.1.2 Reações reversíveis

Em um sistema de reações, podem existir reações químicas no seguinte formato:

$$\begin{aligned} A &\xrightarrow{k_1} B, \\ B &\xrightarrow{k_2} A. \end{aligned} \tag{2.5}$$

Ou seja, uma das reações possui produtos que são reagentes de outra, e *vice-versa*. Nestes casos, uma notação mais compacta pode ser usada. É possível representar as duas reações da seguinte forma:

$$A \underset{k_2}{\overset{k_1}{\rightleftharpoons}} B. \tag{2.6}$$

A reação 2.6 é chamada de reação reversível, no sentido de que A se transforma em B, mas B também pode se transformar em A. A forma como essas reações interagem é descrita posteriormente.

2.1.2.1 Constante de equilíbrio

Em um sistema com duas reações que interagem de forma competitiva, como é o caso da reação (2.5), em um determinado momento A produzirá B à mesma taxa em que B produzirá A (entende-se A e B como os reagentes e produtos de uma reação reversível, respectivamente). Isso faz com que a reação entre em equilíbrio, ou seja, as concentrações de A e B ficam constantes.

A relação entre a concentração dos produtos e reagentes em equilibro é dada pelo que é chamado de constante de equilíbrio, descrito pela seguinte equação: $K = \frac{[B]}{[A]}$, em que $[A]$ e $[B]$ são as concentrações de A e B no equilíbrio, respectivamente.

Para uma reação genérica $aA + bB \underset{k_2}{\overset{k_1}{\rightleftharpoons}} cC + dD$, a constante de equilíbrio pode ser estendida para:

$$K = \frac{[C]^c[D]^d}{[A]^a[B]^b}. \tag{2.7}$$

Um valor de $K > 1$ significa que, em equilíbrio, existe uma concentração maior dos produtos em relação à concentração dos reagentes. Caso $K < 1$, a concentração dos reagentes é maior que a dos produtos. Já quando $K = 1$, produtos e reagentes possuem o mesmo nível de concentração [Chemistry LibreTexts, 2017].

2.1.3 Concentrações das espécies

Para o total entendimento das modelagens matemáticas que expressam o comportamento das reações, é importante entender como a quantidade das espécies são medidas no sistema. Entende-se como sistema, o ambiente no qual as espécies estão inseridas e as reações estão ocorrendo e.g.: solução aquosa [Brown et al., 2017c].

Normalmente, para a realização de uma reação química, as espécies são inseridas em uma solução (mistura composta de duas ou mais substâncias [Ewing et al., 1994]). Por conta disso, a quantidade de uma espécie é expressa por meio de sua concentração, que é uma relação entre a quantidade da espécie e o volume da solução.

A concentração de uma espécie é descrita por uma unidade de medida chamada Molar [Mills et al., 1993]. Ela é definida pela número de mols (quantidade da espécie) divida por litro (volume da solução), sendo representada pelo símbolo mol/L ou M

Mol é uma unidade de medida que expressa a quantidade de uma certa substância. Seguindo a definição de Marquardt et al. [2018], o *mol* é a quantidade de uma substância

que contém 6.022×10^{23} entidades elementares desta certa substância. Este número é conhecido como constante de Avogadro [Mohr et al., 2016]. Entende-se como entidade elementar o que compõe a substância, podendo ser partícula, átomo, molécula, em diante. Ou seja, para ser ter 1 *mol* da espécie A, é necessário ter 6.022×10^{23} moléculas de A.

A medida de Mol é importante porque é utilizada para definir uma relação entre a quantidade de moléculas ou átomos de uma substância e sua massa. O isótopo de carbono ^{12}C possui 12 gramas por mol (12 g a cada 6.022×10^{23} átomos). Esta relação é chamada de massa molar. Ou seja, ^{12}C possui massa molar de 12 g/mol [Brown et al., 2017a].

2.1.4 Velocidade das reações químicas

A notação $A \longrightarrow B$ define que uma espécie A se transforma na espécie B a medida que o tempo passa, mas ainda é necessário descrever como isso ocorre. Ou seja, como as concentrações de A e B variam no tempo. Para isto, existe uma área da Química chamada de Cinética Química, que estuda sobre a velocidade que uma reação química ocorre [Laidler, 2017].

Para fazer o estudo da velocidade da reação $A \longrightarrow B$ (também chamada de taxa), em uma situação prática, um químico realiza um experimento desta reação. Enquanto o experimento ocorre, as concentrações de A e B (denotadas por $[A]$ e $[B]$) são medidas em diferentes momentos no tempo. Suponha, para fins de exemplificação, que as medições de concentração resultaram nos valores mostrados na Tabela 2.1.

Tabela 2.1: Comportamento de uma reação hipotética $A \longrightarrow B$ medida experimentalmente. Tempo medido em horas e concentrações em molar (M).

Tempo (h)	$[A]$	$[B]$
0	1.00E-05	0
2.222222	6.70E-06	3.30E-06
4.444444	4.49E-06	5.51E-06
6.666667	3.01E-06	6.99E-06
8.888889	2.02E-06	7.98E-06
11.11111	1.35E-06	8.65E-06
13.33333	9.07E-07	9.09E-06
15.55556	6.08E-07	9.39E-06
17.77778	4.08E-07	9.59E-06
20	2.73E-07	9.73E-06

Estes valores são representados graficamente pela Figura 2.1. O objetivo é definir

uma forma compacta e clara de representar o comportamento da reação (concentrações das espécies A e B) em qualquer intervalo de tempo.

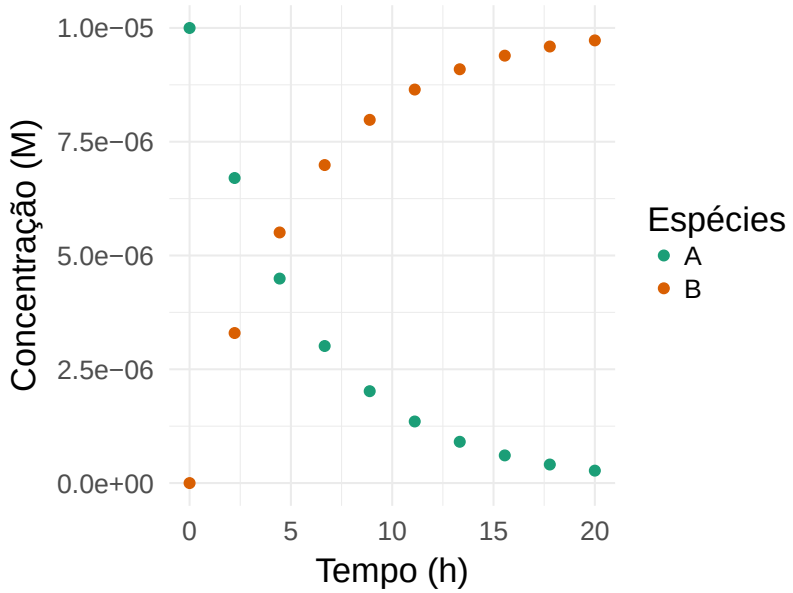


Figura 2.1: Representação gráfica da concentração das espécies na reação $A \longrightarrow B$.

Uma forma de modelar matematicamente o comportamento da concentração das espécies no tempo é se baseando na lei da ação das massas. Esta lei afirma que a velocidade de uma reação química em temperatura constante é diretamente proporcional à concentração dos reagentes [Chemistry LibreTexts, 2015; Chellaboina et al., 2009].

Com isso, é possível definir uma expressão para a velocidade da reação $A \longrightarrow B$ como sendo:

$$r(t) = k[A]_t^n; \qquad k,\ n, \in \mathbb{R}; \quad t \in \mathbb{N}; \tag{2.8}$$

em que $r(t)$ expressa a taxa no tempo t e $[A]_t$ é a concentração de A no tempo t.

Para uma reação mais genérica $aA + bB + \cdots \longrightarrow$ produtos, temos:

$$r(t) = k \times [A]_t^n \times [B]_t^m \times \cdots ; \qquad k,\ n,\ m\ \in \mathbb{R}; \quad t \in \mathbb{N}. \tag{2.9}$$

A função de velocidade de uma reação é chamada de cinética da reação. A cinética de uma rede reações é a associação de uma função de velocidade para cada reação na rede [Feinberg, 1979].

Note que o k (também chamado de constante de velocidade) usado na notação da reação 2.1 (na parte superior da seta) é o mesmo utilizado na equação de velocidade. Esta constante determina a proporcionalidade entre reagentes e produtos. Ela

é definida com base nas condições da reação, como temperatura e tipos de reagentes, por exemplo [Brown et al., 2017b].

Os coeficientes m e n definem a ordem da reação. Eles expressam o quanto a concentração de A e B afetam a velocidade da reação, respectivamente. É importante ressaltar que estes valores não são necessariamente iguais aos coeficientes estequiométricos das espécies (a e b). Na prática, estes valores são obtidos experimentalmente. A ordem da reação é definida como a soma da ordem dos reagentes, no caso sendo $m+n$.

Sendo a velocidade nada mais do que a variação da concentração das espécies no tempo, também podemos reescrever a função de velocidade para a reação $A \xrightarrow{k} B$ como sendo:

$$r(t) = -\frac{\Delta[A]_t}{\Delta t} = \frac{\Delta[B]_t}{\Delta t}; \tag{2.10}$$

em que $\Delta[A]_t$ representa a variação da concentração de A do tempo $t-1$ a t e $\Delta t = t-(t-1)$.

A velocidade é dada pelo negativo da variação da concentração de A porque A está sendo consumida na reação enquanto B está sendo formada. Portanto, unindo as equações 2.8 e 2.10, temos:

$$-\frac{\Delta[A]_t}{\Delta t} = \frac{\Delta[B]_t}{\Delta t} = k[A]_t^n. \tag{2.11}$$

Uma equação que expressa a velocidade de uma reação em termos da variação da concentração das espécies é chamada de lei de velocidade diferencial [Brown et al., 2017b]. Com $\Delta t \to 0$, tem-se:

$$-\frac{d[A]}{dt} = \frac{d[B]}{dt} = k[A]_t^n. \tag{2.12}$$

A expressão 2.12 pode ser reescrita da seguinte forma:

$$\begin{cases} \dfrac{d[A]}{dt} = -k[A]_t^n \\ \dfrac{d[B]}{dt} = k[A]_t^n \end{cases}. \tag{2.13}$$

O sistema de equações diferencias ordinárias (EDOs) 2.13 é uma das formas de expressar matematicamente o comportamento de uma ou uma rede de reações químicas.

2.1.4.1 Relação entre as constantes de velocidade e a constante de equilíbrio

Uma reação reversível está em equilíbrio quando as velocidades das reações de ida e volta são iguais (culminando no estado estacionário das concentrações das espécies).

Ou seja, uma reação genérica $aA + bB \underset{k_2}{\overset{k_1}{\rightleftharpoons}} cC + cD$, em equilíbrio, implica em:

$$k_1[A]^a[B]^b = k_2[C]^c[D]^d, \tag{2.14}$$

assumindo as velocidades $k_1[A]^a[B]^b$ e $k_2[C]^c[D]^d$ para as reações de ida e volta, respectivamente. Esta equação pode ser reescrita como:

$$\frac{k_1}{k_2} = \frac{[C]^c[D]^d}{[A]^a[B]^b}. \tag{2.15}$$

A constante de equilíbrio K para esta reação é formulada como $K = \dfrac{[C]^c[D]^d}{[A]^a[B]^b}$. Isso quer dizer que K também poder ser expresso em termos das constantes de velocidade da seguinte forma: $K = \dfrac{k_1}{k_2}$

2.1.5 Modelagem de reações químicas formais

Como já foi dito anteriormente, em uma situação prática, a ordem da reação e a constante de velocidade são obtidos experimentalmente. Mas como determinar estes valores em reações hipotéticas?

Em situações teóricas em que se quer estudar o comportamento de uma reação hipotética (também chamada de formal) $A \xrightarrow{k} B$, são utilizadas as chamadas funções de velocidade. Estas funções determinam equações de velocidade para diferentes conjuntos de reações. As mais usadas são as baseadas na lei de ação das massas [Anderson, 2014]. A Tabela 2.2 mostra funções de velocidade de reações para algumas reações.

Tabela 2.2: Exemplos de funções de velocidade para diferentes reações. As três últimas colunas mostram as taxas de variação no tempo da concentração das espécies A, B e C em função da velocidade da reação.

Reação	Função de velocidade	Taxa de $[A]$	Taxa de $[B]$	Taxa de $[C]$
1: $A \xrightarrow{k} B$	$r(t) = k[A]_t$	$r_A(t) = -r(t)$	$r_B(t) = r(t)$	$r_C(t) = 0r(t) = 0$
2: $A \xrightarrow{k} B + C$	$r(t) = k[A]_t$	$r_A(t) = -r(t)$	$r_B(t) = r(t)$	$r_C(t) = r(t)$
3: $A + B \xrightarrow{k} C$	$r(t) = k[A]_t[B]_t$	$r_A(t) = -r(t)$	$r_B(t) = -r(t)$	$r_C(t) = r(t)$
4: $A + A \xrightarrow{k} B$	$r(t) = k[A]_t^2$	$r_A(t) = -2r(t)$	$r_B(t) = r(t)$	$r_C(t) = 0r(t) = 0$
5: $A + A \xrightarrow{k} A$	$r(t) = k[A]_t^2$	$r_A(t) = -r(t)$	$r_B(t) = 0r(t) = 0$	$r_C(t) = 0r(t) = 0$
6: $A + A \xrightarrow{k} B + B$	$r(t) = k[A]_t^2$	$r_A(t) = -2r(t)$	$r_B(t) = 2r(t)$	$r_C(t) = 0r(t) = 0$

Como mostradas na Tabela 2.2, as taxas de variação da concentração em função do tempo das espécies são definidas em função da velocidade das reações. Note que a

taxa de uma espécie X é representada por $r_X(t) = \frac{d[X]}{dt}$.

Para generalizar a definição das taxas das concentrações das espécies, é necessário definir os tipos de reações formais tratadas neste trabalho, os quais são apresentados na Tabela 2.3. As reações da tabela representam de forma genérica as reações que se enquadram nas classes referenciadas por elas.

Tabela 2.3: Tipos de reações. A expressão $\cdots$ representa qualquer conjunto de espécies.

Tipo	Reação
Unimolecular	$A \xrightarrow{k} \cdots$
Bimolecular	$A + B \xrightarrow{k} \cdots$
Trimolecular	$A + B + C \xrightarrow{k} \cdots$
Geração	$\emptyset \xrightarrow{k} \cdots$
Degradação	$\cdots \xrightarrow{k} \emptyset$

Em seguida tem-se uma descrição textual de cada tipo de reação:

- Unimolecular, Bimolecular e Trimolecular: reações formais que possuem apenas uma, duas ou três moléculas de uma ou várias espécies como reagentes, respectivamente;
- Geração: reações formais em que a concentração dos reagentes são tidas como constantes. Isso faz com que os produtos sejam gerados à uma taxa constante k;
- Degradação: o conjunto de produtos gerados por reações deste tipo são ignorados (considerando o conjunto de produtos, vazio).

Existem alguns casos particulares que merecem ser descritos com mais detalhes. Por exemplo, uma reação bimolecular em que $A = B$, A reação $A + A \xrightarrow{k} \cdots$ pode ter sua notação simplificada por $2A \xrightarrow{k} \cdots$. No entanto, o valor 2 que precede A não possui o mesmo significado de um coeficiente estequiométrico 2 (mencionado na subseção 2.1.1). Deste ponto em diante no trabalho, valores que precedem nomes de espécies são tratados apenas como uma simplificação de notação e não como um coeficiente estequiométrico.

Com o intuito de generalizar a definição das taxas das concentrações das espécies, são estipulados $s_r(A)$ e $s_p(A)$ como sendo a quantidade de moléculas da espécie A como reagente e produto de uma reação qualquer, respectivamente. Ou seja, em uma reação formal $2A \xrightarrow{k} A$, $s_r(A) = 2$ e $s_p(A) = 1$. Com isso, tem-se que $s(A) = s_p(A) - s_r(A)$,

o que define a quantidade de moléculas de A em uma determinada reação. A taxa de variação da concentração de A em função do tempo é descrita por:

$$r_A(t) = s(A) \times r(t). \tag{2.16}$$

Como exemplo, observe a reação 5 da Tabela 2.2. Neste caso, $s_r(A) = 2$ e $s_p(A) = 1$. Portanto, $s(A) = 1 - 2 = -1$ resultando em $r_A(t) = -r(t)$.

Note que neste exemplo, $r(t) = k[A]_t^2$, ou seja, a ordem n do reagente A é $n = 2$. Este valor foi obtido com base na modelagem teórica adotada. Como na reação 5 são necessárias duas moléculas de A para a reação ocorrer, a velocidade é definida como $r(t) = k([A]_t[A]_t) = k[A]_t^2$. Lembrando que, em uma situação prática, a ordem dos reagentes não pode ser definida desta forma, sendo obtida experimentalmente. Neste caso a ordem de reação foi definida teoricamente por convenção já que é o mecanismo de uma reação formal definida teoricamente.

2.2 Reações Químicas Acopladas

Até agora, foi discutida a modelagem matemática para o comportamento de uma reação. Quando se tem não uma reação, mas uma rede de reações químicas (*Chemical Reaction Network* – CRN) ocorrendo ao mesmo tempo, as espécies podem influenciar e serem influenciadas por múltiplas reações simultaneamente. Para generalizar a modelagem realizada para um conjunto de reações é necessária apenas algumas mudanças na notação já definida.

Em uma CRN, a definição da função de velocidade não sofre alteração, continuando sendo definida em função da concentração dos reagentes. Mas a taxa de variação da concentração das espécies sofre alteração. Seja i uma reação no conjunto de reações R, a taxa de uma espécie A para uma reação i é definida como:

$$r_A(t, i) = s_i(A) \times r_i(t); \qquad s_i(A) = s_p(A, i) - s_r(A, i); \tag{2.17}$$

onde $r_i(t)$ é a função de velocidade da reação i, $s_p(A, i)$ e $s_r(A, i)$ representam a quantidade de moléculas de A como reagente e produto da reação i, respectivamente.

Sendo assim, a taxa de $[A]$ levando em conta todas as reações em R, tem-se:

$$r_A(t) = \sum_{i \in R} s_i(A) \times r_i(t). \tag{2.18}$$

A seguir são apresentados alguns exemplos de CRNs.

2.2.1 Oscilador Lotka–Volterra

Este oscilador químico proposto em 1920 por A. J. Lotka é, possivelmente, uma das primeiras CRNs de comportamento oscilante propostas [Noyes, 1989]. Esta CRN possui as seguintes reações:

$$A + X \xrightarrow{k_1} 2X; \tag{2.19a}$$

$$X + Y \xrightarrow{k_2} 2Y; \tag{2.19b}$$

$$Y \xrightarrow{k_3} P. \tag{2.19c}$$

As reações (2.19a) e (2.19b) são chamadas de autocatalíticas porque elas geram produtos que são utilizados como reagentes delas mesmas [Poshusta, 2016], acarretando uma retroalimentação da reação.

Esta CRN pode ser interpretada como uma modelagem do comportamento populacional de duas espécies: uma predadora Y interagindo com uma espécie presa X. A representa o ecossistema em que X e Y estão. A reação (2.19a) modela a etapa de procriação da presa. Na reação (2.19b) os predadores (Y), consomem as presas (X) para se manterem e procriarem. A morte dos predadores é representada pela reação (2.19c).

Para simplificar a modelagem, pode-se assumir que a concentração de A não varia no tempo [Soloveichik et al., 2010]. Assim, a reação (2.19a) se torna:

$$X \xrightarrow{k_1^{'}} 2X; \qquad k_1^{'} = k_1[A]. \tag{2.20}$$

Na prática, a concentração constante de A poderia ser alcançada inserindo A no sistema a uma taxa constante (a mesma em que ela seria consumida). Para adicionar um reagente continuamente, tem-se que fazer a reação em uma tanque de mistura agitado com alimentação contínua (CSTR), onde o reagente é continuamente alimentado no vaso de reação [Dolnik et al., 1997] (este tipo de operação é comum em reatores industriais). No laboratório costuma-se fazer reações sem adição continua de reagentes (batelada) e, nesse caso, usa-se o reagente em excesso para obter a simplificação da equação (2.20).

As velocidades das reações são as seguintes:

$$r_1(t) = k_1^{'}[X]_t; \tag{2.21a}$$

$$r_2(t) = k_2[X]_t[Y]_t; \tag{2.21b}$$

$$r_3(t) = k_3[Y]_t, \tag{2.21c}$$

respectivamente.

Portanto, as taxas de variação das espécies são dadas por:

$$\begin{cases} r_X(t) = \dfrac{d[X]}{dt} = r_1(t) - r_2(t) = k_1'[X]_t - k_2[X]_t[Y]_t & \text{(2.22a)} \\ r_Y(t) = \dfrac{d[Y]}{dt} = r_2(t) - r_3(t) = k_2[X]_t[Y]_t - k_3[Y]_t \quad . & \text{(2.22b)} \\ r_P(t) = \dfrac{d[P]}{dt} = r_3(t) \qquad\quad = k_3[Y]_t & \text{(2.22c)} \end{cases}$$

O sistema de equações diferencias (2.22) representa a variação da concentração das espécies no tempo. Resolvendo este sistema para $k_1' = k_3 = \frac{1}{300}\ s^{-1}$, $k_2 = 5\text{E}5\ M^{-1}s^{-1}$, $[X]_0 = 20\text{E}{-9}\ M$, $[Y]_0 = 10\text{E}{-9}\ M$ e $[P]_0 = 0\ M$ (valores obtidos a partir do trabalho de Soloveichik et al. [2010]), tem-se o comportamento no tempo mostrado na Figura 2.2.

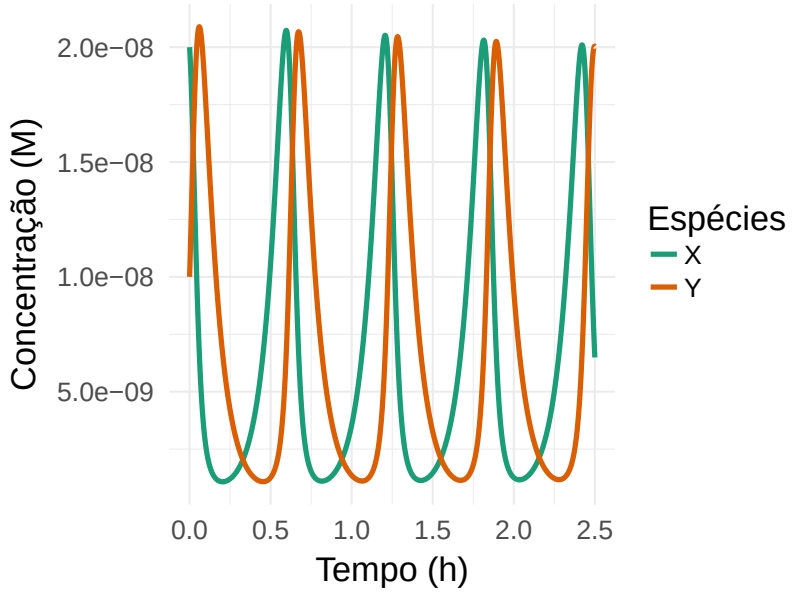


Figura 2.2: Representação gráfica da solução do sistema de EDOs (2.22) para as espécies X e Y.

Note que a unidade de medida de k varia de acordo com o tipo de reação. Em reações unimoleculares, k possui unidade s^{-1}. Já em reações bimoleculares, $M^{-1}s^{-1}$ representa a unidade de k. Isso ocorre para manter conformidade com a unidade de medida de uma velocidade de reação ou taxa de variação de espécie, que deve ser sempre Molar por tempo ($M \times s^{-1}$). Supondo a velocidade $[A]k$, a unidade de $k = s^{-1}$ para que $[A]k$ tenha a unidade $M \times s^{-1}$. Já uma velocidade $[A][B]k$, a relação direta não linear de duas espécies resulta em uma unidade M^2. Portanto, com k tendo a

unidade $M^{-1}s^{-1}$, se obtém a unidade de medida resultante $M^2 \times M^{-1}s{-}1 = M \times s^{-1}$.

2.2.2 *Consensus*

Este CRN foi proposta por Chen et al. [2013] e tem como propósito demonstrar as capacidades que uma rede de reações tem de se portar como um controlador. Ou seja, este circuito responde de maneiras determinadas dependendo da entrada que recebe. Entrada, neste caso, é a concentração de duas espécies (X e Y).

Esta CRN tem o seguinte comportamento: dado a concentração de duas espécies de entrada X e Y, a rede aumenta a concentração da espécie que possui concentração $\max([X], [Y])$ e consome a espécie com concentração igual a $\min([X], [Y])$. Ou seja, a espécie em maioria é intensificada enquanto a que está em minoria é consumida. Quando convergir, ou seja, quando esta rede atingir seu estado de equilíbrio químico (em que as concentrações não alteram mais), a espécie em maioria terá concentração igual a $[X]_0 + [Y]_0$ e a concentração da espécie em minoria será 0.

A rede possui as seguintes reações e funções de velocidade:

$$X + Y \xrightarrow{k_1} 2B, \qquad r_1(t) = k_1[X]_t[Y]_t; \tag{2.23a}$$

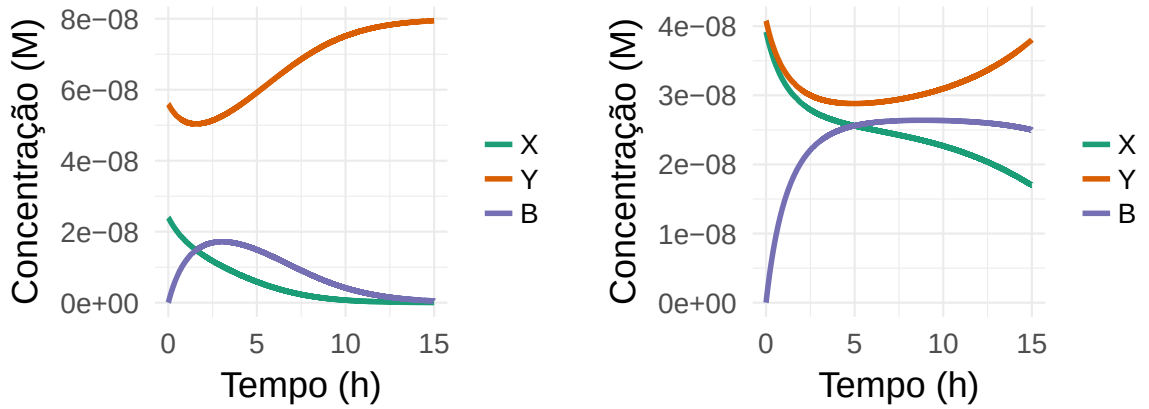
$$B + X \xrightarrow{k_2} 2X, \qquad r_2(t) = k_2[B]_t[X]_t; \tag{2.23b}$$

$$B + Y \xrightarrow{k_3} 2Y, \qquad r_3(t) = k_3[B]_t[Y]_t. \tag{2.23c}$$

Com isso, tem-se que as taxas de variação das espécies são:

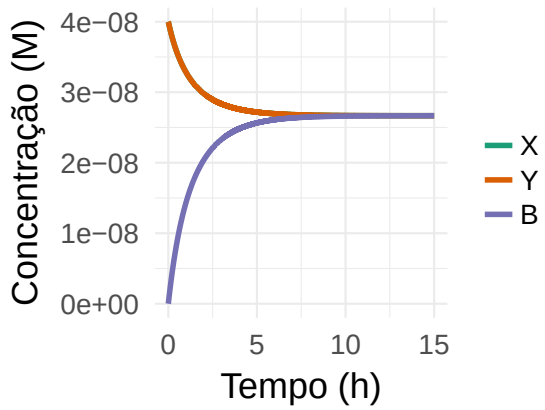
$$\begin{cases} \dfrac{d[X]}{dt} = -k_1[X]_t[Y]_t + k_2[B]_t[X]_t & (2.24a) \\ \dfrac{d[Y]}{dt} = -k_1[X]_t[Y]_t + k_3[B]_t[Y]_t & (2.24b) \\ \dfrac{d[B]}{dt} = 2k_1[X]_t[Y]_t - k_2[B]_t[X]_t - k_3[B]_t[Y]_t & (2.24c) \end{cases}.$$

Estas reações resultam no comportamento visto na Figura 2.3. Note que, mesmo em condições em que $[X]_0$ e $[Y]_0$ são muito próximos, como mostrado na Figura 2.3b, a rede se comporta como deveria, porém mais lentamente. Mas no caso particular em que as duas espécies de entrada possuem concentrações idênticas (Figura 2.3c), todas as espécies entram em um estado de equilíbrio, sem eliminar ou potencializar nenhuma das espécies de entrada

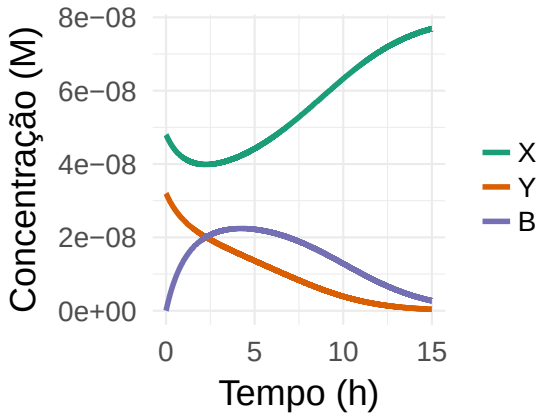


(a) $[X]_0 = 10\%,\ [Y]_0 = 70\%$

(b) $[X]_0 = 49\%,\ [Y]_0 = 51\%$



(c) $[X]_0 = 50\%,\ [Y]_0 = 50\%$



(d) $[X]_0 = 60\%,\ [Y]_0 = 40\%$

Figura 2.3: Comportamento da CRN *Consensus* com a variação da entrada. As concentrações nas legendas estão representando a porcentagem de 8E−8 M. $[B]_0$ é 0 M e $k_1 = k_2 = k_3 = 2\text{E}3\ M^{-1}s^{-1}$ em todos os casos. Note que na Figura 2.3c o comportamento de $[X]$ não está explicito porque ele está exatamente abaixo da linha de $[Y]$ (os dois possuem o mesmo comportamento).

2.3 Simulando Reações Químicas no R

A forma de simular uma CRN depende de como ela foi modelada. Utilizando a modelagem das leis das velocidades diferencias, simular uma CRN se resume em resolver o sistema de equações diferencias ordinárias (EDOs) resultante da modelagem. Existem vários métodos numéricos para resolver um sistema de EDOs dependendo das características do sistema em particular [Hairer & Wanner, 1991; Hairer et al., 1993; Butcher, 2001]. Também existem várias implementações destes métodos [Hindmarsh, 1983; Brown et al., 1989; Butcher, 1987] nas mais variadas linguagens de programação.

As simulações deste trabalho foram realizadas utilizando implementações de mé-

todos numéricos contidos em um pacote de *software* chamado `deSolve` [Soetaert et al., 2010]. `deSolve` possui implementações de várias métodos numéricos em linguagens de baixo nível como `C`, `C++` e `FORTRAN`, mas estes são acessados por meio de uma linguagem de alto nível chamada `R` [R Core Team, 2017].

`R` é uma linguagem de programação muito usada para análises estatísticas e de dados em geral por conta das ferramentas estatísticas e de visualização já incorporadas à linguagem [Vance, 2009]. Contudo, por conta de sua facilidade de integração com linguagens de baixo nível (como `C++`, por exemplo [Eddelbuettel et al., 2011]), `R` pode ser utilizado na implementação de soluções de problemas computacionalmente custosos.

O pacote `deSolve` utiliza dessa possibilidade de interoperabilidade com linguagens de baixo nível que `R` proporciona para prover interfaces para o usuário que agregam facilidade de uso e desempenho na utilização de métodos numéricos. Este pacote contém implementações de métodos numéricos que resolvem equações diferencias ordinárias, equações algébricas diferencias, equações diferencias parciais e equações diferencias com atraso [Soetaert et al., 2010]. Todas devem ter uma condição inicial definida, assim entrando para uma classe de problemas chamada de Problema de Valor Inicial (PVI). Dando as condições inicias da CRN, o sistema de EDOs que modela a CRN se torna um problema em que o `deSolve` pode ser utilizado para resolver.

A seguir é mostrado um exemplo de como simular uma reação $A \xrightarrow{k} B$ no `R` com o `deSolve`.

2.3.1 Exemplo: reação unimolecular simples

Este exemplo é apresentado para demonstrar uma das formas de simular CRNs no `R`. O objetivo é simular a reação $A \xrightarrow{k} B$.

Primeiro, é necessário carregar o pacote `deSolve`. É possível fazer isso com o seguinte Algoritmo:

```
1 library(deSolve)
```

Algoritmo 2.1: carregando o `deSolve`.

O *script* de simulação pode ser divido em duas partes:

- especificação do modelo, onde são definidas as expressões matemáticas que representam o modelo a ser simulado;
- execução da simulação, em que, dado o modelo especificado, métodos numéricos são utilizados para resolver de forma aproximada as expressões matemáticas definidas. Como resultado, tem-se o comportamento simulado da reação química.

Começando pela especificação do modelo, como é mostrado no Algoritmo 2.2, primeiramente são definidas as concentrações inicias de A e B (linha 1, $[A]_0 = 1\text{E}{-5}\ M$ e $[B]_0 = 0\ M$).

```
1  y0 <- c(A = 1e-5, B = 0.0) # [A]_0, [B]_0
2  pars <- list(k = 5e-5)     # Parâmetros do modelo
3                             # (constante de velocidade)
```

Algoritmo 2.2: Parâmetros do modelo

Na linha 2 são definidos todos os parâmetros que o modelo utiliza. No caso, apenas a constante de velocidade ($k = 5\text{E}{-5}\ s^{-1}$) é necessária. Caso houvesse outras reações, as outras constantes deveriam ser atribuídas à variável `pars`.

A próxima etapa é definir a função que representa o modelo matemático. Lembrando, o modelo da reação $A \xrightarrow{k} B$ é representando pelo sistema de EDOs (2.25).

$$\begin{cases} \dfrac{d[A]}{dt} = -k[A]_t \\ \dfrac{d[B]}{dt} = k[A]_t \end{cases}. \tag{2.25}$$

O Algoritmo 2.3 mostra a função que define o sistema 2.25. Esta função é chamada iterativamente por algum dos métodos numéricos do `deSolve` (escolhido pelo usuário).

```
1  fx <- function(t, y, pars) {
2      with(pars, {
3          dy <- numeric(2)
4          dy[1] <- -k*y[1]  # d[A]/dt
5          dy[2] <-  k*y[1]  # d[B]/dt
6          list(dy) # Saída
7      })
8  }
```

Algoritmo 2.3: Função modelo.

A linha 2 descompacta os parâmetros definidos na linha 2 do Algoritmo 2.2, para que seja possível utilizar os dados contidos em `pars` sem precisar referenciar o nome `pars` o tempo todo. No caso, apenas a variável `k` está contida em `pars`.

Na linha 3 é criado um vetor que comporta dois valores numéricos. Estes valores representam a variação da concentração de A e B. Os valores do vetor são atribuídos

nas linhas 4 e 5. Note que os cálculos feitos nestas linhas representam as expressões matemáticas do sistema 2.25. `y` é um vetor com $[A]$ e $[B]$ em um ponto no tempo.

Finalmente, na linha 6, o valor das derivadas de $[A]$ e $[B]$ são retornados. Com isso, tem-se definido a especificação do modelo.

Agora, partindo para execução da simulação, é necessário a definição do intervalo de tempo da simulação e do método numérico. O Algoritmo 2.4 mostra como definir o intervalo de tempo da simulação.

```
1  t <- seq(0, 72000, length.out = 10)    # (s)
```

Algoritmo 2.4: Definindo o intervalo de tempo.

No caso, a simulação vai ocorrer durante 72.000 segundos (20 horas), com a concentração das espécies sendo avaliada em 10 pontos igualmente distribuídos no tempo. Ou seja, o intervalo de tempo é discretizado em 10 pontos. Para se ter uma maior precisão na simulação (aumentando o número de pontos no tempo dentro de um intervalo), basta aumentar o parâmetro `length.out`.

O Algoritmo 2.5 mostra a etapa final do processo de simulação. É neste momento que a função `ode()`, do pacote `deSolve`, é utilizada para resolver numericamente um modelo matemático `fx()` com o estado inicial `y0`, com os parâmetros `pars` durante o intervalo de tempo `t`.

```
1  out <- ode(
2      times = t,
3      y     = y0,
4      func  = fx,
5      pars  = pars
6  )
```

Algoritmo 2.5: Uso do método numérico para resolver o sistema de EDOs.

Note que o modelo `fx()`, que é uma função, é passado como parâmetro para `ode()`. Isso porque `fx()` é chamado iterativamente pelo método numérico usado por `ode()`. Internamente, os parâmetros da função `fx()` são definidos em cada iteração e o retorno da mesma usado pelo método numérico.

A função `ode()` em si não é um método numérico, ela apenas é utilizada como interface para os métodos. É possível especificar um método utilizando o parâmetro `method`. Caso nenhum seja definido, o método chamado LSODA [Petzold, 1983] é utilizado como padrão

O resultado de `ode()` (contido em `out`) é o comportamento das concentrações das espécies no tempo, como mostrado a seguir:

```
> out
    time             A             B
1      0 1.000000e-05 0.000000e+00
2   8000 6.671976e-06 3.328024e-06
3  16000 4.462676e-06 5.537324e-06
4  24000 2.990997e-06 7.009003e-06
5  32000 2.004515e-06 7.995485e-06
6  40000 1.343019e-06 8.656981e-06
7  48000 8.997926e-07 9.100207e-06
8  56000 6.028607e-07 9.397139e-06
9  64000 4.039198e-07 9.596080e-06
10 72000 2.706274e-07 9.729373e-06
```

Algoritmo 2.6: Resultado da função `ode()`.

A representação gráfica deste comportamento pode ser observada na Figura 2.4.

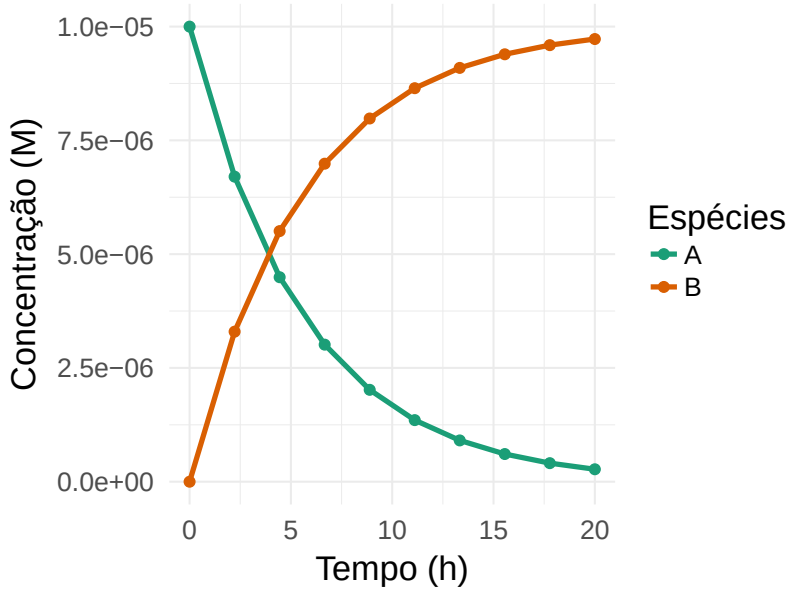


Figura 2.4: Representação gráfica do resultado da simulação apresentado no Algoritmo 2.6.

Com o entendimento do que uma reação química teórica (formal) representa e alguns de seus aspectos na prática, torna-se possível explorar possibilidades interessantes que este tipo de modelagem proporciona. Na próxima seção, um estudo de como modelar comportamento de lógica por meio de reações químicas é apresentado.

2.4 Lógica Booleana com Reações Químicas

Como foi visto anteriormente, uma reação química é definida por meio de um conjunto de reagentes, os quais se transformam em um conjunto de produtos. Os produtos são gerados apenas caso todos os reagentes existam no sistema. Observe a reação $A + B \longrightarrow C$, por exemplo. Nesta reação, uma molécula de C é gerada apenas se uma molécula de A e outra de B se chocarem. Ou seja, a produção de C ocorre apenas se A e B existirem.

Com isso, percebe-se que a modelagem de um comportamento booleano utilizando reações químicas pode ser uma mera questão de interpretação. Ao invés de ver C sendo gerado a partir de A e B, pode-se interpretar C como sinal que só é ativada quando A e B estão ativados. Sendo assim, C é resultado da operação lógica A `AND` B.

Este é apenas um exemplo simples de como uma reação química pode ser interpretada como uma operação `AND`. Para operações mais complexas e características necessárias de qualquer mecanismo booleano, como comportamento biestável (representando valores 0 e 1), por exemplo, uma abordagem mais elaborada se faz necessária.

A abordagem discutida neste trabalho é a apresentada por Jiang et al. [2013]. Ela descreve como construir portas lógicas (e.g.: `AND`, `OR`, `NOR`, etc.) com reações químicas, e como conecta-las, permitindo a criação de circuitos lógicos digitais. Existem outras formas de se construir portas lógicas com reações químicas [Arkin & Ross, 1994; Senum & Riedel, 2011], no entanto, esta foi escolhida pela sua simplicidade e robustez quanto ao valor da constante de velocidade das reações (todas as reações podem ter a mesma constate de velocidade) e estados de equilíbrio bem definidos (é bem previsível quando a CRN entrará em um de seus estados de equilíbrio e quais estados são estes).

2.4.1 Representação de um bit

Para representar o valor de um determinado bit X, esta abordagem utiliza o valor da concentração de duas espécies X_0 e X_1. A presença de X_0 representa $X = 0$. Já a presença de X_1 representa $X = 1$. Obviamente, X_0 e X_1 não podem estar presentes ao mesmo tempo, e isto é garantido pelas reações (2.26).

$$X_0 + X_1 \longrightarrow S_X; \tag{2.26a}$$

$$S_X + X_0 \longrightarrow 3X_0; \tag{2.26b}$$

$$S_X + X_1 \longrightarrow 3X_1. \tag{2.26c}$$

Caso existam X_0 e X_1 no sistema, S_X é gerado. Esta espécie é responsável por amplificar os dois sinais (X_0 e X_1). Lembrando da cinética da ação das massas,

a velocidade da reação é determinada por uma constante e pela concentração dos reagentes. Por conta disso, dentre as reações (2.26b) e (2.26c), a reação que tiver o sinal com maior concentração ocorrerá a uma velocidade maior, tendo um maior poder de amplificação. O sinal mais fraco é consumido no processo de amplificação.

Um conjunto de reações como as apresentadas em (2.26) é definido para cada sinal do sistema. Portanto, na implementação de um gate com duas entradas X e Y e uma saída Z, três conjuntos de reações (2.26) são definidos (um para X, outro para Y e o terceiro conjunto para Z).

2.4.2 Implementando uma porta `AND`

Para implementar uma porta `AND` com as entradas X, Y e a saída Z, dois comportamentos devem existir:

1. ativar Z quando $X = Y = 1$;
2. desativar Z se este não é o caso.

Para obter o comportamento 1, é necessária, basicamente, uma forma de transferir as concentrações de X_1 e Y_1 para Z_1. Fazer isso com a reação $X_1 + Y_1 \longrightarrow Z_1$ implica que X_1 e Y_1 seriam consumidos no processo, fazendo com que $X = Y = 0$ ao mesmo tempo que $Z = 1$. Ou seja, consumir a entrada pode levar a CRN para estados logicamente inválidos. A reação $X_1 + Y_1 \longrightarrow Z_1 + X_1 + Y_1$ mantém as concentrações das entradas inalteradas, mas faz com que Z_1 seja gerado indefinidamente, o que pode tornar a CRN instável onde Z seja usado como entrada.

A solução é utilizar a seguinte CRN:

$$X_1 + Y_1 \longrightarrow X_1 + Y_1 + Z_1^{'}; \tag{2.27a}$$

$$2Z_1^{'} \longrightarrow \emptyset; \tag{2.27b}$$

$$Z_1^{'} + Z_0 \longrightarrow Z_1. \tag{2.27c}$$

Nesta CRN, a reação (2.27a) gera um elemento intermediário $Z_1^{'}$ que indica que Z deve ser 1. A reação (2.27c) utiliza o indicador e Z_0 para gerar Z_1. Caso $Z_1^{'} = 1$, Z_0 (que, por construção, existirá no sistema no momento inicial) será consumido para gerar Z_1. Caso $Z_1^{'} = 0$, Z_1 e Z_0 permanecem inalterados. Para fazer com que $Z_1^{'}$ não seja gerado indefinidamente, a reação (2.27b) é definida. Entende-se $\longrightarrow \emptyset$ como a remoção de $Z_1^{'}$ do sistema, ou apenas que os produtos da reação não importam e são tratados como lixo e ignorados. Esta reação entra em estado de equilíbrio com as outras duas, fazendo com que a concentração de $Z_1^{'}$ se estabilize.

Com o intuito de obter o comportamento 2 (e, consequentemente, a biestabilidade desejada), a CRN (2.28) é utilizada. Nela, Z_1 é consumido na presença de X_0 ou Y_0 para Z_0 ser gerado. Ou seja, Caso X ou Y sejam 0, Z é desativado. Por mais que as concentrações de X_0 e Y_0 não seja alteradas, Z_1 é consumido, limitando a quantidade de Z_0 que será gerada.

$$X_0 + Z_1 \longrightarrow X_0 + Z_0; \tag{2.28a}$$
$$Y_0 + Z_1 \longrightarrow Y_0 + Z_0. \tag{2.28b}$$

2.4.3 Porta `OR`

Para a construção de uma porta `OR`, reações análogas as da porta `AND` são utilizadas, apenas alterando algumas espécies e produtos de algumas reações.

Para esta porta, ainda se tem dois comportamentos distintos desejados, mas que ocorrem em condições distintas. São estes:

1. desativar Z quando $X = Y = 0$;
2. ativar Z, caso contrário.

As reações (2.29) são definidas para se obter o comportamento 1.

$$X_0 + Y_0 \longrightarrow X_0 + Y_0 + Z_0^{'}; \tag{2.29a}$$
$$2Z_0^{'} \longrightarrow \emptyset; \tag{2.29b}$$
$$Z_0^{'} + Z_1 \longrightarrow Z_0. \tag{2.29c}$$

Note a semelhança entre elas e as reações (2.27). Como existem espécies que representam $Z = 0$, $X = 0$ e $Y = 0$, análogas às espécies que simbolizam $Z = 1$, $X = 1$ e $Y = 1$, as mesmas estruturas das reações (2.27) e (2.28) usadas para construir a porta `AND` podem ser utilizadas para construir a porta `OR`, apenas alterando as espécies participantes.

Da mesma forma, as reações (2.30), semelhantes as (2.28), modelam o comportamento 2.

$$X_1 + Z_0 \longrightarrow X_1 + Z_1; \tag{2.30a}$$
$$Y_1 + Z_0 \longrightarrow Y_1 + Z_1. \tag{2.30b}$$

Com isso, tem-se o comportamento mostrado na Figura 2.5. Em A é apresentado os valores de Z no equilíbrio (em um ponto no tempo em que as reações se estabilizaram

e as concentrações das espécies não se alteram) da porta `AND` para múltiplos valores de X e Y. O mesmo é feito em B, mas com a porta `OR`.

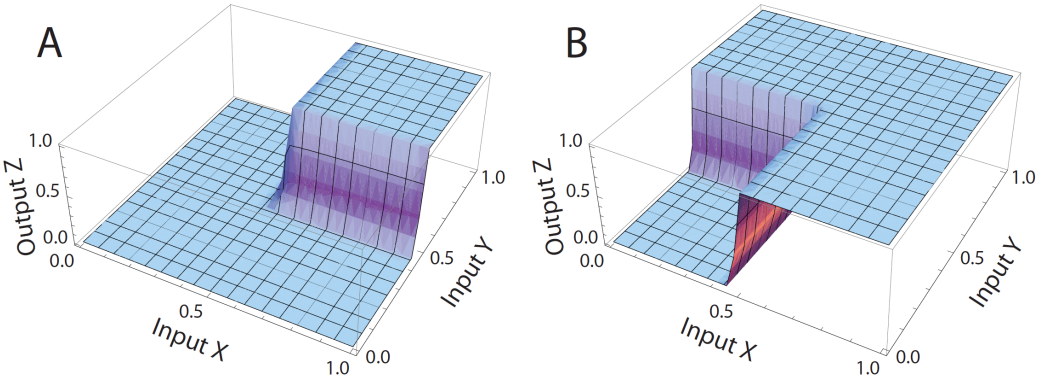


Figura 2.5: Comportamento das portas lógicas AND (A) e OR (B) implementadas com reações químicas [Jiang et al., 2013]. Os valores das concentrações das espécies foram normalizados.

Visto que é possível modelar teoricamente um comportamento lógico digital por meio de reações químicas, uma questão surge: como construir estas reações químicas na prática? A modelagem teórica assume que existem espécies que reagem com um certo conjunto de espécies, à velocidades específicas, para gerar um conjunto de produtos. O problema é que, no mundo real, espécies que se comportam desta forma (sejam estas elementos básicos ou compostos) podem não ser conhecidas ou, até mesmo, não existirem.

DNA (*deoxyribonucleic acid*) entra exatamente neste contexto. De acordo com Soloveichik et al. [2010], ele pode ser usado como uma matéria prima universal para as reações químicas. Isso é discutido mais detalhadamente no Capítulo 3.

Capítulo 3

Implementando Reações Químicas com DNA

Como foi visto na capítulo anterior, a modelagem matemática das reações químicas fornecem uma linguagem versátil para descrever o comportamento dinâmico de mecanismos químicos. Além disso, é possível utilizar esta linguagem para idealizar mecanismos teóricos interessantes de se construir na prática.

Um problema que surge com a criação destes mecanismos químicos teóricos é determinar o que serão, na prática, as espécies definidas nos circuitos químicos. Observe a reação $A + B \xrightarrow{k} C$, por exemplo. Para que esse mecanismo seja implantado em um ambiente real, é necessário encontrar elementos ou compostos que reajam a uma velocidade específica determinada pela constante k, e que gerem como produto C, estritamente.

Muitos mecanismos químicos teóricos não podem ser implementados na prática pelo fato de não se conhecerem reagentes que se comportem da maneira que o mecanismo exige. Um estudo que tenta mitigar este problema é o feito por Soloveichik et al. [2010]. Nele é proposto uma maneira programática de implementar CRNs teóricas na prática, utilizando DNA (Ácido desoxirribonucleico – *deoxyribonucleic acid*) como elemento base para todas as espécies.

Para entender o que este estudo propõe, é necessário conhecer a estrutura do DNA e como esta pode ser modelada para se comportar como uma reação química qualquer.

3.1 Estrutura do DNA

Moléculas de DNA são polímeros formados por monômeros (moléculas que podem se ligar a outras) chamados de nucleotídeos [Amos et al., 2002; Adams et al., 1986]. Seguindo o modelo de dupla hélice proposto por Watson et al. [1953], estes nucleotídeos (adenina (A), citosina (C), guanina (G), timina (T)) se ligam em pares por meio de pontes de hidrogênio. Como mostrado na Figura 3.1, os pares formados seguem a complementaridade de Watson-Crick, em que A sempre se liga com T e G sempre se liga com C. Pelo fato das ligações entre os nucleotídeos serem feitas de forma tão previsível, o comportamento de um dispositivo baseado em DNA apresentado em simulação é bem próximo do que ocorre no mundo real.

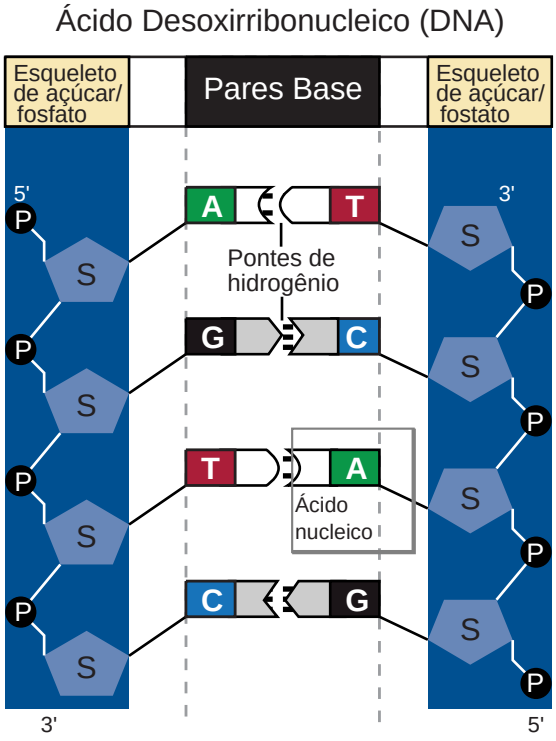


Figura 3.1: Estrutura do DNA de dupla hélice [Watson et al., 1953]. Os ácidos nucleicos (A, C, G, T) se uns ligam aos outros por meio de pontes de hidrogênio formando os pares de bases. Estas bases também estão ligadas à esqueletos formados por fosfato (P) e açúcar (desoxirribose (S)) quem mantém a estrutura do DNA. Imagem adaptada de National Institutes of Health [2016].

Diferentes representações gráficas são utilizadas para facilitar a descrição visual do DNA. A Figura 3.2 apresenta algumas destas representações. Sequências de nucleotídeos são definidos como domínios nomeados. Estas sequências possuem contradomínios, que são sequências que se ligam com os domínios já definidos. Na Figura 3.2, os

domínios são nomeados com números e seus contradomínios possuem o mesmo número mas acrescido de um *.

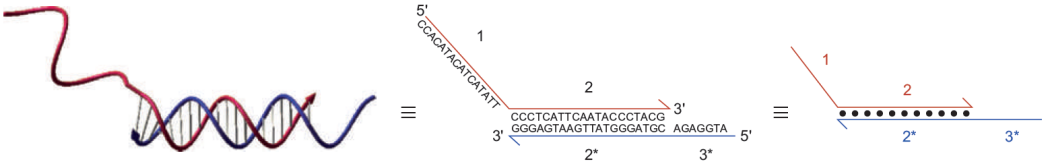


Figura 3.2: Possíveis representações do DNA [Zhang & Seelig, 2011]. Os valores numéricos representam domínios e aqueles com uma estrela representam os complementos daqueles domínios.

As marcações 3' e 5' que aparecem nos extremos dos fragmentos representam o sentido do fragmento de DNA.

As reações químicas são representadas em DNA por meio da combinação de fragmentos (reagentes), que resultam em outros fragmentos e complexos (produtos). A seguir é descrito como os fragmentos de DNA se ligam e de que forma estes fragmentos podem ser projetados para se ligarem segundo um mecanismo reações formais proposto.

3.2 Reação de Deslocamento de Cadeia

Fragmentos de DNA precisam ser construídos de uma maneira específica para que a ligação entre dois fragmentos possa ocorrer. Ou seja, cada fragmento ou complexo de DNA que representa uma espécie na CRN deve ter uma estrutura e sequência de domínios específica para que as reações ocorram como desejado.

Uma forma de fazer com que fragmentos de DNA se combinem gerando produtos é pelo processo de deslocamento de cadeia [Zhang & Seelig, 2011]. O processo se baseia nos seguintes conceitos:

- Hibridização: técnica usada para ligar fragmentos de DNA (ou RNA) de cadeia simples para a formação de complexos de cadeia dupla (dupla hélice) explorando a complementaridade de Watson-Crick [MeSH, 2017];

- Migração de ramo: movimento de troca de fragmentos entre complexos de DNA [Hastings, 2001]. Em outras palavras, é um fenômeno que ocorre quando um fragmento de um complexo de DNA é trocado por outro fragmento. Esta troca pode ocorrer entre dois complexos de dupla-hélice ou entre um fragmento simples e um complexo.

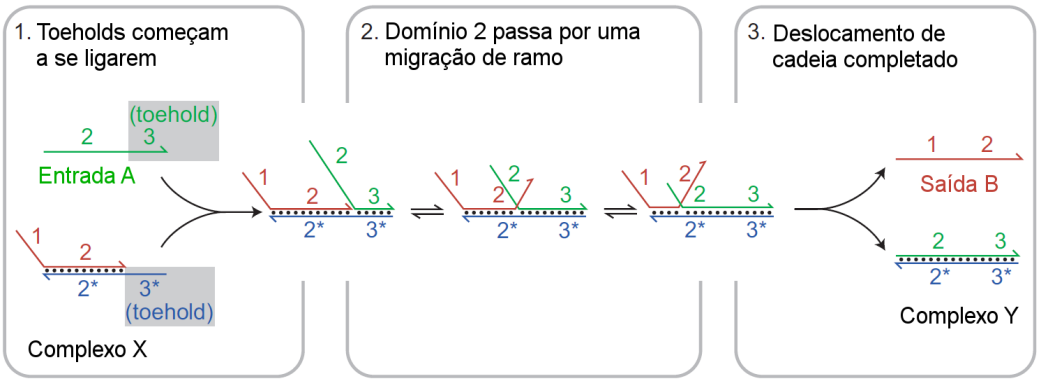


Figura 3.3: Exemplo de reação de deslocamento de cadeia [Zhang & Seelig, 2011]. Circuito que, recebendo uma entrada A (cadeia de DNA com domínios 2 e 3), gera B como saída (cadeia de DNA com domínios 1 e 2).

Estes dois conceitos são utilizados para definir o processo de interação entre moléculas de DNA, de forma que combinações possam ocorrer entre elas e produtos possam ser gerados a partir destas combinações.

A Figura 3.3 mostra um exemplo de reação de deslocamento de cadeia em 3 etapas. O sistema começa com dois tipos de moléculas de DNA: um fragmento simples chamado de A, e um complexo de dupla-hélice parcial X. A possui um domínio especial chamado de *toehold* (neste caso representado pelo número 3). Este tipo de domínio é menor (possui entre 4 e 10 nucleotídeos enquanto os outros domínios possuem no mínimo 20 [Lakin et al., 2012]) e facilita o início do processo de hibridização entre as duas moléculas, já que X apresenta o contra-domínio 3^*. Perceba que, dependendo do tamanho do *toehold* e de sua sequência de nucleotídeos, o processo de hibridização do mesmo pode se reverter mais facilmente. Nucleotídeos A/T possuem uma ligação mais fraca do que G/C.

O início da hibridização desencadeia a migração de ramo, mostrado na etapa 2. Os domínios de A começam a se ligar nos seus contradomínios em X, removendo as ligações do fragmento vermelho que estava associado ao complexo. Este fenômeno ocorre por meio de uma série de hibridizações e separações estocásticas dos nucleotídeos. Tem-se esta natureza estocástica porque os domínios do fragmento vermelho (que já estavam no complexo) e os do fragmento A podem se associar aos contradomínios existentes no complexo. Este comportamento pode ser modelados matematicamente como um passeio aleatório [Zhang & Seelig, 2011].

Após a migração de ramo, tem-se um novo fragmento (de saída) no sistema, chamado de B, derivado do complexo X. Também passa a existir um novo complexo, Y, formado por parte do complexo X e o fragmento de entrada A. A reação entre X e

A, formando B e Y, se completa porque o complexo Y possui configuração energética mais estável [Benenson, 2012].

O complexo Y não reage com B porque não existem *toeholds* em Y que possam facilitar a inicialização da hibridização de B com Y. O único *toehold* existente em Y é o domínio 3 (e 3*), e este já está hibridizado e, por conta disso, ele está inativo.

Note que, como o *toehold* é o facilitador para o processo de hibridização começar, o seu tamanho e sequência de nucleotídeos é utilizado para definir a velocidade da reação entre as moléculas de DNA. De acordo com Zhang & Seelig [2011], variando o tamanho do *toehold*, pode-se alcançar constantes de velocidade entre 1 M^{-1} s^{-1} e 6E6 M^{-1} s^{-1}.

3.2.1 Cascateando reações

O exemplo mostrado anteriormente descreve como duas moléculas de DNA podem reagir por meio do deslocamento de cadeia. Mas para ser possível implementar redes de reações químicas, várias reações entre DNAs devem ocorrer simultaneamente em um mesmo sistema. Mais do que isso, deve ser possível cascatear reações, de forma que o produto de uma reação seja o reagente de outra.

O seguinte exemplo descreve como reações entre moléculas de DNA ocorrem. Tome o estado inicial do sistema como tendo 3 tipos de moléculas: um complexo chamado de G_1 e dois fragmentos simples A e B. A Figura 3.4 mostra a configuração inicial do sistema.

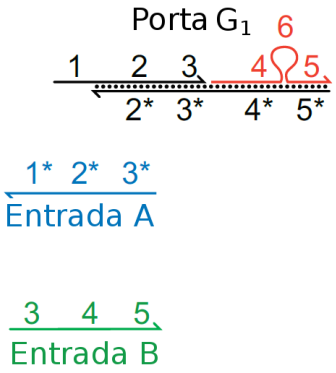


Figura 3.4: Configuração inicial do exemplo de reação em cascata. Imagem adaptada de Zhang & Seelig [2011].

G_1 possui um *toehold* ativo denominado 1. O contradomínio 1* presente em A pode se ligar com este *toehold*. Neste momento, B permanece estável. Com o deslocamento de cadeia ocorrendo entre A e G_1, tem-se que o fragmento com os domínios 1, 2

e 3 que compunha o complexo G_1 é separado do mesmo e ligado ao fragmento A. Isto é mostrado na Figura 3.5.

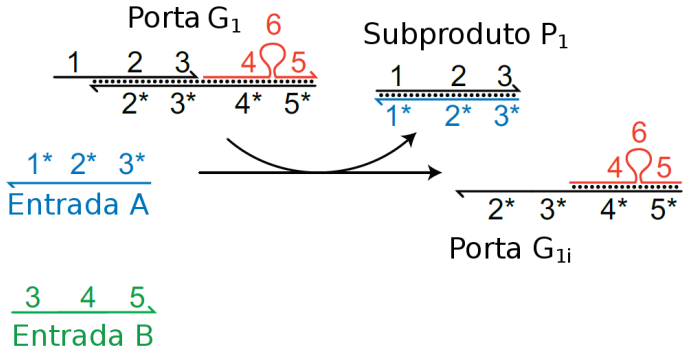


Figura 3.5: Estado do sistema após a reação entre G_1 e A. Imagem adaptada de Zhang & Seelig [2011].

Com a conclusão desta reação, duas moléculas passam a existir no sistemas: G_{1i} e P_1. P_1 já possui todos os seus domínios hibridizados e, portanto, está em uma configuração estável. Ou seja, este não deve ser mais usado em outras reações. Já G_{1i} possui dois domínios ativos que não existiam antes (2^* e 3^*). B, que ainda existente no sistema, pode agora usar 3^* em G_{1i} como *toehold* para começar sua hibridização.

A Figura 3.6 mostra a interação entre G_{1i} e B ocorrendo e o que é gerado como produto desta reação. B começa sua hibridização pelo *toehold* 3^*. A partir disso, o fragmento vermelho em G_{1i} é deslocado e separado do complexo, dando lugar ao B.

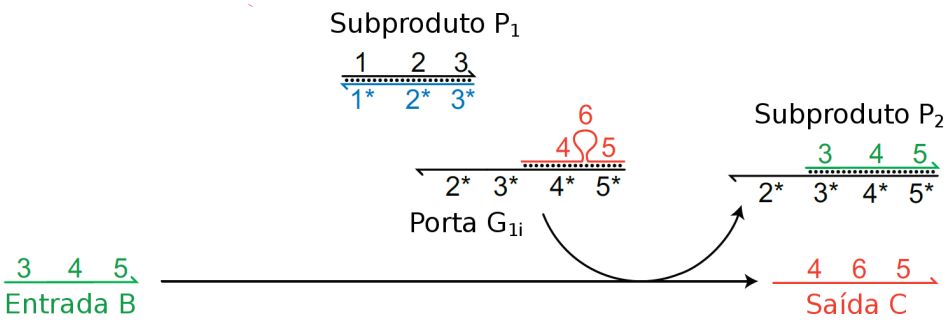


Figura 3.6: Estado final do sistema após as duas reações em cascata. Imagem adaptada de Zhang & Seelig [2011].

Como resultado, tem-se o fragmento simples em vermelho (chamado de C) e o complexo P_2. Como não existe uma molécula que possa iniciar o processo de hibridização com o contradomínio 2^*, os dois produtos permanecem estáveis, concluindo as reações do mecanismo.

O exemplo descrito pode ser representado como uma CRN a seguir:

$$G_1 + A \xrightarrow{k_1} P_1 + G_{1i}; \tag{3.1a}$$

$$G_{1i} + B \xrightarrow{k_2} P_2 + C, \tag{3.1b}$$

em que A e B reagem com um complexo (o qual é modificado no processo e possui dois estados chamados de G_1 e G_{1i}). Caso A e B existam no sistema, o fragmento C é gerado como saída. Note que este comportamento é o mesmo de uma porta `AND`, tendo como entrada A e B, e como saída C, como é mostrado na Figura 3.7.

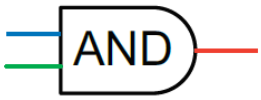


Figura 3.7: As reações em cascata representam uma porta `AND`, em que os fragmentos verde (A) e azul (B) são as entradas, e o fragmento vermelho (C) simboliza a saída. Ou seja, C existirá no sistema apenas se A e B existirem. Imagem adaptada de Zhang & Seelig [2011].

Perceba que neste conjunto de reações as duas entradas A e B são consumidas para formação de C. O complexo G_1, que representa a porta, também é utilizado no processo e sua estrutura é modificada, impossibilitando sua reutilização.

É necessário notar também que G_1 é compatível apenas com A e B, e consegue gerar apenas C como resposta. Ou seja, esta não é uma porta `AND` independente das moléculas de DNA. São necessários que os fragmentos de entrada e saída tenham uma sequência de domínios específica para a reação com a porta ocorrer. Caso as espécies de entrada e saída alterem, a estrutura da porta também precisa ser modificada.

3.3 Transformando Reações Químicas Formais em Reações de DNA

Com o entendimento de como moléculas de DNA podem interagir entre si e de que forma estas interações são cascateadas, é possível utilizar esses conhecimentos para construir uma série de mecanismos baseados em reações entre DNAs. Com a possibilidade de projetar fragmentos de DNA que reagem a uma certa velocidade, CRNs teóricas (formais) passam a ser passíveis de sintetização através do DNA como elemento base.

É necessário distinguir reações químicas formais de reações de DNA, já que reações de DNA também podem ter seu comportamento representado por um sistema de EDOs da mesma forma que a CRNs formais. Todavia, reações entre DNA podem ser sintetizadas para reproduzir o comportamento de outras reações químicas, estas apenas modeladas teoricamente (chamadas de reações químicas formais).

Para transformar uma CRN formal em uma CRN de DNA de forma programática é necessária uma abordagem de construção. Uma destas abordagens é a apresentada por Soloveichik et al. [2010], e ela permite até que reagentes sejam usados em múltiplas reações simultaneamente.

Existem outras abordagens para se fazer isso. Uma delas é descrita por Chen et al. [2013], a qual permite o uso de fragmentos naturais de DNA (que são mais puros e podem ser mais longos que os fragmentos produzidos em laboratório). No entanto, a abordagem de Soloveichik et al. [2010] foi a escolhida por sua simplicidade e clareza no momento de descrever como converter as constantes de velocidade das reações formais para as reações de DNA.

Esta abordagem se baseia em dividir uma reação química em duas etapas em cascata: recebimento das entradas (reagentes) e geração da saída (produtos). Isto é feito para tornar os fragmentos de entrada e saída independentes. Ou seja, os domínios dos fragmentos reagentes não influenciam nos domínios dos fragmentos dos produtos, permitindo a implementação de reações com espécies arbitrárias.

3.3.1 Reações unimoleculares

Suponha que se tem como objetivo implementar a reação $X_1 \xrightarrow{k_i} X_2 + X_3$ em DNA. Para tal, são necessárias que duas moléculas de DNA já existam no sistema. São estas G_i e T_i (G e T associados a uma reação i). G_i é responsável por verificar se a entrada X_1 existe e T_i libera os produtos X_2 e X_3 caso G_i confirme a existência da entrada.

A Figura 3.8 mostra os domínios que representam as espécies X_1, X_2 e X_3. As sequências de domínios destas espécies determinam os domínios de G_i e T_i, já que estes devem reagir diretamente com a entrada e saídas.

As estruturas de G_i e T_i são mostradas na Figura 3.9. Como é possível notar, em T_i estão contidos os dois fragmentos que representam X_2 e X_3.

Com as moléculas de DNA dadas por estas estruturas, a primeira reação que ocorre no sistema é entre X_1 e G_1, como mostrado na Figura 3.10. Esta reação ocorre a uma velocidade com a constante de velocidade q_i, liberando como produto O_i e uma outra molécula a qual contém a entrada, simbolizando o consumo da mesma. Esta

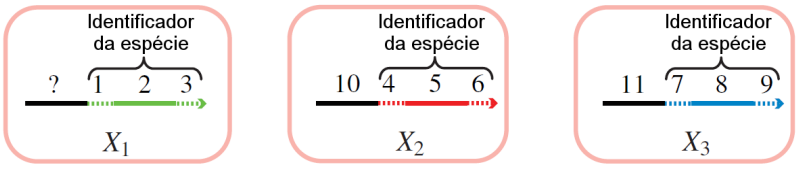


Figura 3.8: Fragmentos que representam as espécies X_1, X_2 e X_3 da reação unimolecular. Os *toeholds* são identificados com o traçado tracejado e comprimento menor. O domínio com o traçado contínuo em preto é apenas um marcador que não influencia nas reações. Imagem adaptada de Soloveichik et al. [2010].

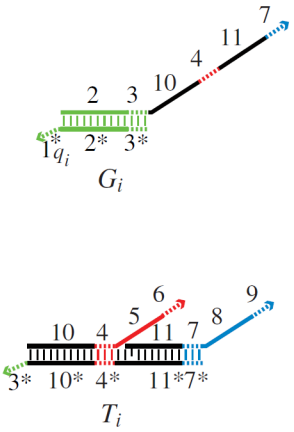


Figura 3.9: Fragmentos que representam as moléculas auxiliares G_i e T_i de uma reação unimolecular de dois produtos i. Assume-se que os fragmentos de entrada e saída possuem a estruturas mostradas na Figura 3.8. Imagem adaptada de Soloveichik et al. [2010].

outra molécula não é mais utilizada no sistema e, portanto, passa a ser considerada lixo e ignorada.

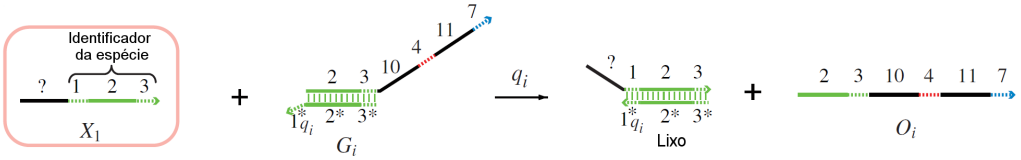


Figura 3.10: Primeira etapa da reação unimolecular formal em DNA. Imagem adaptada de Soloveichik et al. [2010].

O_i, também chamado de intermediador, serve como ponte entre a etapa 1 e a etapa 2. Ela associa o resultado da detecção da entrada com reação que libera as espécies de saída. Quando O_i é liberado, o mesmo se liga com T_i pelo *toehold* 3*, fazendo com que os fragmentos de saída X_2 e X_3 se desprendam do complexo T_i, liberando-os. Esta

segunda reação pode ser vista graficamente na Figura 3.11. Uma terceira molécula também é gerada como produto da etapa 2 e ignorada por estar em um estado inativo (todos os domínios hibridizados).

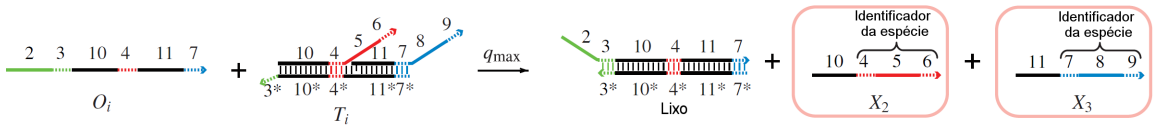


Figura 3.11: Segunda etapa da reação unimolecular formal em DNA. Imagem adaptada de Soloveichik et al. [2010].

Estas duas reações podem ser representadas usando a notação de reação química da seguinte forma:

$$X_1 + G_i \xrightarrow{q_i} O_i; \tag{3.2a}$$

$$O_i + T_i \xrightarrow{q_{max}} X_2 + X_3, \tag{3.2b}$$

sendo a reação 3.2a e a 3.2b as etapas 1 e 2, respectivamente.

Para alterar o número de produtos que são gerados pela reação, basta alterar o tamanho do fragmento O_i (inicialmente ligado ao complexo G_i) e os fragmentos do complexo T_i (os quais representam as espécies de saída).

A velocidade das reações 3.2 devem ser ajustadas para estas duas reações representarem o mesmo comportamento de $X_1 \xrightarrow{k_i} X_2 + X_3$. Para tal, define-se C_{max} como sendo a concentração inicial das espécies auxiliares G_i e T_i. Sendo τ como o tempo do experimento, de acordo com Soloveichik et al. [2010], a CRN 3.2 deve respeitar as seguintes restrições:

$$C_{max} \gg \tau k_i \max([X_1]); \tag{3.3a}$$

$$q_{max} C_{max} \gg k_i; \tag{3.3b}$$

$$q_{max} \geq q_i, \tag{3.3c}$$

onde $\max([X_1])$ simboliza a concentração máxima que o reagente atinge, levando em conta qualquer momento dentro do intervalo de tempo da reação.

Respeitando estas condições, tem-se que:

$$q_i = \frac{k_i}{C_{max}}. \tag{3.4}$$

Já que q_i é sempre menor ou igual a q_{max} (equação (3.3c)), a primeira reação (3.2a) é a que limita a velocidade da CRN (3.2). Portanto q_i é o parâmetro que deve ser ajustado para fazer com que as reações de DNA aconteçam a uma determinada

velocidade k_i. Lembrando que, para variar a velocidade de uma reação de DNA, basta manipular o tamanho e a sequência de nucleotídeos do *toehold* usado para facilitar a inicialização do processo de hibridização.

Portanto, resumindo o processo de implementação de uma reação química formal unimolecular utilizando DNA, tem-se a seguinte tradução:

$$X_1 \xrightarrow{k_i} X_2 + X_3 \Longrightarrow \begin{array}{l} X_1 + G_i \xrightarrow{q_i} O_i; \\ O_i + T_i \xrightarrow{q_{max}} X_2 + X_3, \end{array} \tag{3.5}$$

de forma que o lado esquerdo é a reação química formal, e o lado direito são as reações entre moléculas de DNA necessárias para emular o comportamento da reação química formal.

3.3.2 Reações bimoleculares

Agora suponha que se queira implementar a reação $X_1 + X_2 \xrightarrow{k_i} X_3$. Esta reação possui as mesmas espécies da reação anterior, mas com X_2 agora sendo um reagente, e não um produto. Isso faz com que essa reação seja bimolecular.

Para implementar esta reação em DNA usando a abordagem de Soloveichik et al. [2010], a mesma separação em duas etapas é usada: detecção dos reagentes e liberação dos produtos caso os reagentes sejam detectados. Todavia, este caso possui dois reagentes, e não apenas um como no exemplo anterior. Portanto, para se detectar os dois reagentes, duas reações são necessárias. A primeira, para detectar X_1, é mostrada na Figura 3.12.

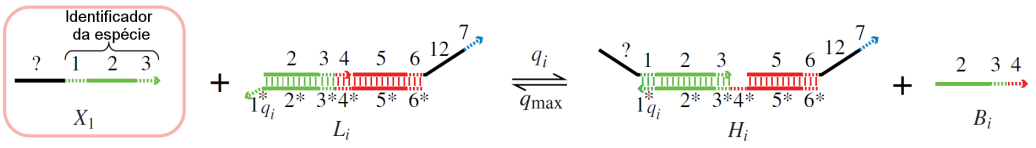


Figura 3.12: Primeira reação da primeira etapa para reações bimoleculares. Responsável por detectar X_1. Imagem adaptada de Soloveichik et al. [2010].

Note que esta reação é reversível. Quando a mesma ocorre, ainda resta um *toehold* ativo (4^*) que pode ser hibridizado pelo fragmento que foi liberado pela reação (B_i). Isto faz com que B_i reaja com H_i, liberando X_1 novamente no sistema. E como a constante de velocidade da reação de volta (q_{max}) é maior ou igual que a de ida (q_i, com base na restrição (3.3c)), a reação entra em equilíbrio (a reação de ida ocorre na mesma velocidade que a de volta, tornando estável a concentração das espécies) com

X_1 não sendo consumido. Caso esta reação não fosse reversível, X_1 seria consumido mesmo se X_2 não existisse, o que não refletiria o comportamento desejado.

A reação da Figura 3.13 detecta X_2. Note que esta reação ocorre apenas se a reação da Figura 3.12 tiver ocorrido, já que se faz necessário a molécula H_i, a qual só é gerada após a conclusão da primeira reação. Portanto, caso não exista X_1, X_2 não é consumido.

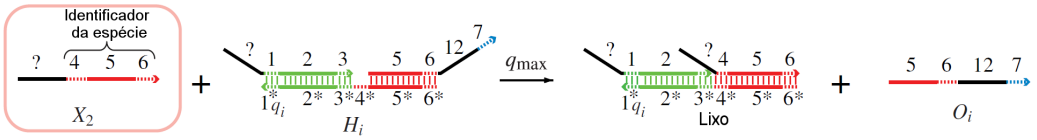


Figura 3.13: Segunda reação da primeira etapa para reações bimoleculares. Responsável por detectar X_2. Imagem adaptada de Soloveichik et al. [2010].

A molécula O_i, por sua vez, reage com a molécula T_i, como mostrado na Figura 3.14. Esta reação libera a espécie de saída X_3, concluindo a CRN.

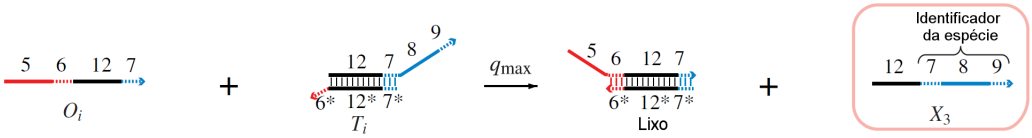


Figura 3.14: Reação da segunda etapa para reações bimoleculares. Responsável por liberar X_3 caso X_1 e X_2 já tenham sido detectados. Imagem adaptada de Soloveichik et al. [2010].

Novamente, para fazer uma reação bimolecular que gere mais de um produto, basta alterar a estrutura de O_i e T_i.

Da mesma forma que foi feito na subseção anterior, estas reações entre moléculas de DNA podem ser representadas pela notação das reações químicas. A transformação de uma reação química formal bimolecular para uma CRN de DNA se dá por:

$$X_1 + X_2 \xrightarrow{k_i} X_3 \Longrightarrow \begin{array}{l} X_1 + L_i \underset{q_{max}}{\overset{q_i}{\rightleftharpoons}} H_i + B_i \\ X_2 + H_i \xrightarrow{q_{max}} O_i \\ O_i + T_i \xrightarrow{q_{max}} X_3 \end{array} . \tag{3.6}$$

Das restrições apresentas em (3.3), apenas a (3.3a) é substituída por:

$$C_{max} \gg \tau k_i \max([X_1]) \max([X_2]); \tag{3.7a}$$

$$C_{max} \gg \max([X_1]), \tag{3.7b}$$

todas as outra se aplicam para o caso bimolecular. Respeitando tais restrições, a relação entre q_i e k_i é a seguinte:

$$q_i = k_i. \tag{3.8}$$

Ou seja, para conseguir sintetizar uma CRN em DNA que emule o comportamento de uma reação bimolecular com uma constante de velocidade k_i, basta manipular o tamanho e a sequência de nucleotídeos do *toehold* 1^*, alterando, por consequência, q_i.

3.3.3 Efeito de *buffer*

Para uma reação, a modelagem acima pode funcionar, mas com várias reações em um mesmo sistema, a CRN em DNA pode sofrer de um problema chamado de efeito de *buffer*.

Caso X_3, do exemplo ($X_1 + X_2 \xrightarrow{k_i} X_3$), fosse usado como primeiro reagente de outra reação bimolecular j, a modelagem matemática acima não daria a cinética esperada porque parte da concentração de X_3 entraria em equilíbrio rapidamente com o H_j (por meio da reação $H_j + B_j$, a qual compete com $X_3 + L_j$), levando a reação $X_3 + L_j$ a ter uma velocidade menor do que a desejada.

Para exemplificar o problema, o oscilador Lotka-Volterra (2.19) é usado. Esta CRN pode ser reescrita da seguinte forma:

$$X_1 \xrightarrow{k_1} 2X_1; \tag{3.9a}$$

$$X_1 + X_2 \xrightarrow{k_2} 2X_2; \tag{3.9b}$$

$$X_2 \xrightarrow{k_3} \emptyset, \tag{3.9c}$$

em que $\emptyset$ representa um conjunto de espécies que são ignorados e não influenciam no comportamento do mecanismo.

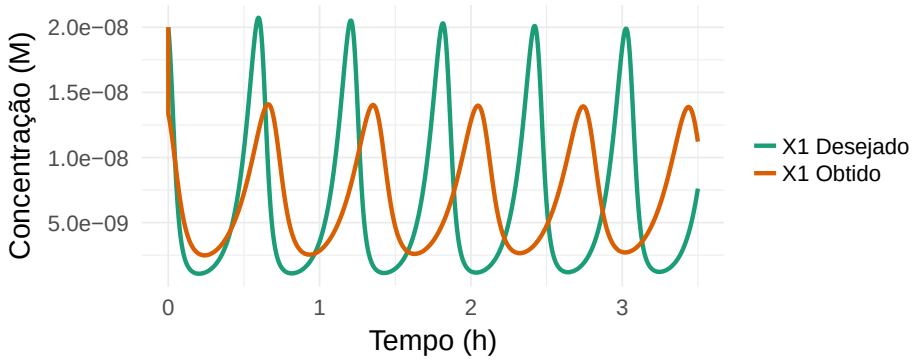
São definidos os parâmetros da CRN como sendo:

- $k_1 = k_3 = {}^1/_{300}\ s^{-1}$;
- $k_2 = 5\text{E}5\ M^{-1}s^{-1}$;
- $[X_1]_0 = 20\text{E}{-}9\ M$;
- $[X_2]_0 = 10\text{E}{-}9\ M$.

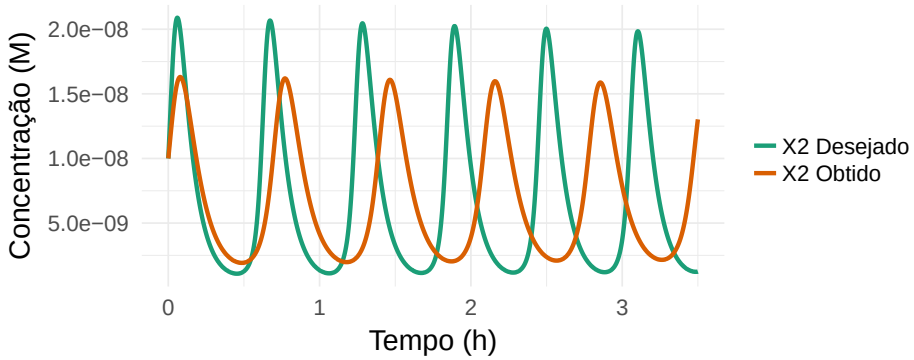
Com isso, são definidos os parâmetros para a CRN de DNA $C_{max} = 1\text{E}{-}5\ M$ e $q_{max} = 1\text{E}6\ M^{-1}s^{-1}$. Por consequência, $q_2 = k_2 = 5\text{E}5\ M^{-1}s^{-1}$ e $q_1 = q_3 = {}^{k_{1,3}}/_{C_{max}} = {}^{1/300}/_{1\text{E}-5} = 333,333\ M^{-1}s^{-1}$.

Com estes parâmetros, tem-se o comportamento mostrado na Figura 3.15. Observando o comportamento de $X1$ na CRN formal (traçado verde) e na de DNA (traçado

laranja), é possível notar diferenças claras. O comportamento de $X1$ em DNA começa com uma queda brusca e depois passa a ter um decaimento de concentração mais brando. Essa variação na velocidade de queda ocorre exatamente quando a primeira reação de (3.6) entra em estado de equilíbrio, ou seja, existe H_1 suficiente no sistema para gerar X_1 de volta na mesma velocidade em que o mesmo é consumido. A partir deste momento, toda a dinâmica de X_1 fica comprometida. E como X_1 influencia no comportamento de X_2 e vice-versa, X_2 também passar a ter um comportamento indesejado, tendo uma variação de concentração mais branda do que o desejado (que é o chamado efeito de *buffer*), impactando na CRN como um todo.



(a)



(b)

Figura 3.15: Efeito de *buffer* ocorrendo no oscilador Lotka-Volterra. As Figuras 3.15a e 3.15b mostram o comportamento das espécies X_1 e X_2, respectivamente. O comportamento da CRN formal é o desejado, enquanto o comportamento da CRN de DNA ainda sem a correção do efeito de *buffer* é o obtido.

Para corrigir este problema, um novo conjunto de reações deve ser adicionado ao mesmo tempo em que um ajuste nos parâmetros da CRN de DNA é feita. A seguir é descrito como isso é realizado.

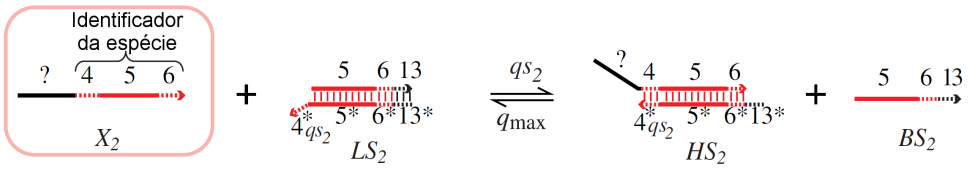


Figura 3.16: Módulo de *buffer* para a espécie X_2. Esta reação pode ser representada por $X_2 + LS_2 \underset{q_{max}}{\overset{qs_2}{\rightleftharpoons}} HS_2 + BS_2$.

É definido $fr(X_j)$ como sendo o conjunto de reações bimoleculares em que X_j é o primeiro reagente. Ou seja, na CRN (3.9), $fr(X_1) = \{X_1 + X_2 \xrightarrow{k_2} 2X_2\}$; $fr(X_2) = \emptyset$, já que X_1 é o primeiro reagente apenas da reação $X_1 + X_2 \xrightarrow{k_2} 2X_2$ e não existe uma reação bimolecular na CRN em que X_2 é o primeiro reagente. Com isso, $\sigma_j = \sum_{i \in fr(X_j)} k_i$ (soma das constantes de velocidade das reações formais em que X_j é o primeiro reagente) e $\sigma = \max_j \sigma_j$. Um novo conjunto de reações (chamado de módulo de *buffer*) é adicionado para cada espécie X_j que possua $\sigma_j < \sigma$. Um módulo de exemplo da espécie X_2 é dado pelo Figura 3.16.

Com os módulos de *buffer* adicionados, basta agora alterar os parâmetros da CRN com base no que é chamado de fator de escala de *buffer*. Este fator de escala é definido como $\gamma^{-1} = q_{max}(q_{max} - \sigma)^{-1}$. Com isto, tem-se que a constante de velocidade da reação do módulo de *buffer* para a espécie X_j é $qs_j = \gamma^{-1}(\sigma - \sigma_j)$. k_i nas equações (3.4) e (3.8) deve ser substituído por $k_i' = \gamma^{-1} k_i$, bem como as concentrações inicias c_j das moléculas de DNA que representam espécies X_j das reações formais devem ser alterados para $c_j' = \gamma^{-1} c_j$.

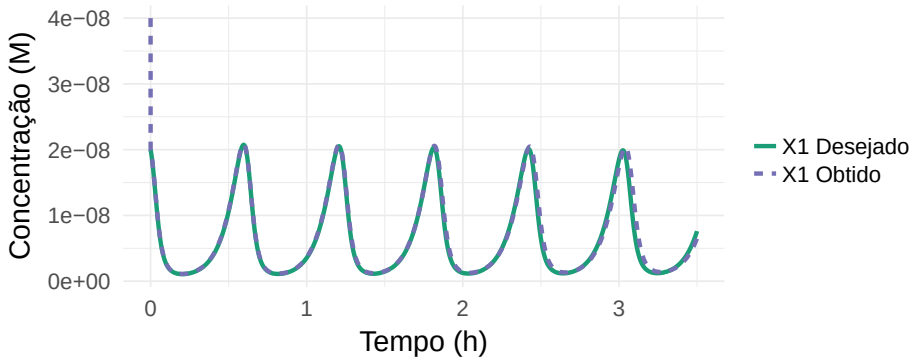
Aplicando o processo descrito acima no exemplo do oscilador Lotka-Volterra, tem-se para as espécies X_1 e X_2, $\sigma_1 = k_2$ e $\sigma_2 = 0$ (já que esta não possuem reações bimoleculares em que a mesma é a primeira reagente). Portanto, $\sigma = \max(\sigma_1, \sigma_2) = \sigma_1$. Já que $\sigma_2 < \sigma$, um módulo de *buffer* é adicionado para X_2.

Com q_{max} = 1E6 $M^{-1}s^{-1}$ e k_2 = 5E5 $M^{-1}s^{-1}$, tem-se que γ^{-1} = 1E6(1E6 − 5E5)$^{-1}$ = 2. Logo, $q_1 = q_3 = {}^{\gamma^{-1}k_{1,3}}/_{C_{max}} = {}^{2 \times 1/300}/_{1\text{E}-5}$ = 666.667 $M^{-1}s^{-1}$ e $q_2 = \gamma^{-1}k_2 = 2 \times$ 5E5 = 1E6 $M^{-1}s^{-1}$.

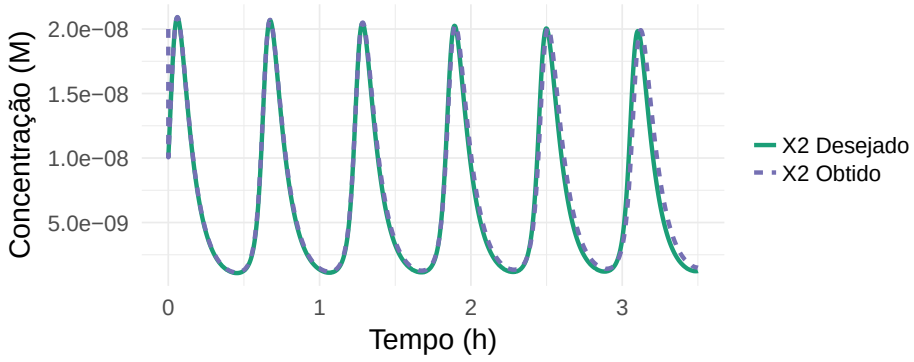
Por fim, tendo $[X_1]_0$ = 20E−9 M e $[X_2]_0$ = 10E−9 M, ajusta-se as concentrações iniciais de X_1 e X_2 multiplicando as mesmas por γ^{-1}, que no caso é 2. Consequentemente, na CRN em DNA $[X_1]_0$ = 40E−9 M e $[X_2]_0$ = 20E−9 M.

Com esta nova modelagem, o comportamento da CRN em DNA é corrigido para emular o comportamento da CRN formal, como é possível ver na Figura 3.17. Existe apenas uma diferença clara no estado inicial do sistema. Isto é devido às concentrações iniciais das espécies alteradas com o processo de adição dos módulos de *buffer*. No

entanto, o comportamento obtido rapidamente converge para o desejado.



(a)



(b)

Figura 3.17: A CRN de DNA com efeito de *buffer* corrigido é mostrado pelo comportamento obtido. O comportamento desejado está sobreposto pelo comportamento obtido, o que indica a equivalência dos mesmos.

Isto concluí a modelagem química para construir uma CRN de DNA que se comporte como uma CRN formal de acordo com o trabalho de Soloveichik et al. [2010], permitindo a implementação física de reações uni e bimoleculares formais.

Já que um mecanismo químico baseado em DNA pode ser modelado como um conjunto de reações químicas como qualquer outro, uma forma de simular seu comportamento é obter a solução numérica do sistema de EDOs obtidos a partir das reações (3.5) e (3.6). No entanto, este tipo de simulação pode ser muito simplista e não mostrar comportamentos imprevistos adversos que podem ocorrer quando se trabalha especificamente com DNA. Para simulações mais detalhadas, uma ferramenta direcionada para este fim se faz necessária. Uma ferramenta que pode ser usada nestas situações é o Microsoft Visual DSD, o qual é descrito com mais detalhes a seguir.

3.4 Microsoft Visual DSD

O Visual DSD (*DNA* Strand Displacement) [Lakin et al., 2011] é uma ferramenta proposta para prototipar e simular dispositivos computacionais implementados em DNA usando a técnica de deslocamento de cadeia. Esta ferramenta, desenvolvida nos laboratórios de pesquisa da Microsoft, é baseada na linguagem de especificação de circuitos de DNA descrita inicialmente por Phillips & Cardelli [2009] e posteriormente expandida por Lakin et al. [2012]. Na Figura 3.18 é possível ver a tela principal da ferramenta.

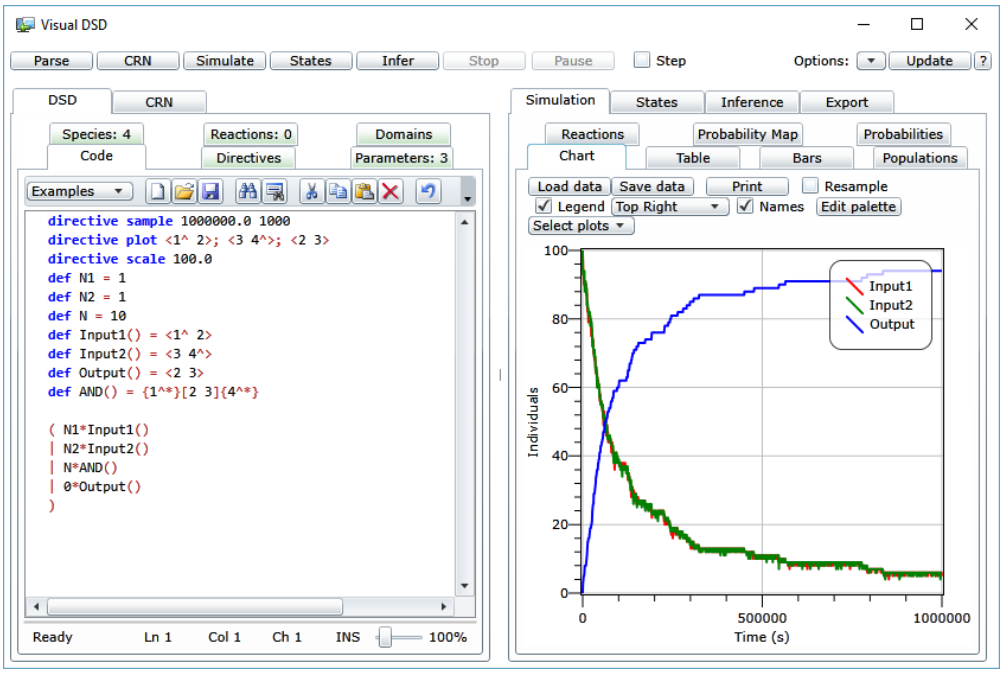


Figura 3.18: Interface principal do Visual DSD. No lado esquerdo é especificado os fragmentos em DNA do sistema, juntamente com os parâmetros de configuração da simulação. Ao lado direito se tem o comportamento simulado do sistema especificado.

Esta ferramenta permite que o usuário faça análises mais avançadas sobre o comportamento de um dispositivo baseado em DNA que opera por meio de reações de deslocamento de cadeia. É possível fazer simulações determinísticas (avaliam o comportamento a nível de variação da concentração das espécies) ou estocásticas (consideram as interações entre cada fragmento de DNA), simular reações não planejadas (não modeladas, mas que podem ocorrer na prática), visualizar a representação gráfica dos fragmentos e complexos de DNA, além de indicar uma possível sequência de nucleotídeos para cada domínio que poderia atender às necessidades do circuito especificado.

3.4.1 Linguagem de especificação

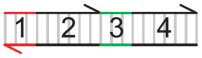
A linguagem de especificação de circuitos de DNA usada pelo Visual DSD permite especificar fragmentos e complexos de DNA, definir seus domínios e quais são seus *toeholds*. Com base na especificação das moléculas, um algoritmo é capaz de gerar automaticamente as reações que irão ocorrer entre elas.

A linguagem suporta a definição não só de fragmentos simples, mas também de estruturas complexas com dois ou mais fragmentos parcialmente ou totalmente ligados. A Figura 3.19 mostra exemplos de como especificar alguns fragmentos e complexos.

(a) `{1^:2}` – Fragmento inferior. O símbolo `^` após um domínio significa que ele é um *toehold*.

(b) `<1^ 2>` – Fragmento superior.

(c) `[1^ 2]` – Dois fragmentos ligados completamente.



(d) `[1^ 2]:[3^ 4]` – Um fragmento inferior ligado a dois superiores. O `:` indica o ponto de corte entre os dois fragmentos superiores.

(e) `[1^ 2]:3^:[4 5^]` – Mesma semântica da Figura 3.19d, apenas com a diferença de que existe um domínio `3` entre os dois fragmentos superiores.

Figura 3.19: Especificando moléculas de DNA utilizando a linguagem do Visual DSD. O subtítulo de cada Figura contém o código na linguagem do Visual DSD que representa a molécula referenciada. Imagens adaptadas de Phillips & Cardelli [2009].

Note que não é possível especificar uma molécula com um fragmento superior ligado a dois inferiores porque isso seria redundante. Basta rotacionar as moléculas das das Figuras 3.19d e 3.19e.

Para adicionar mais de uma molécula no sistema, basta usar o símbolo `|` entre as especificações das moléculas. Assim, estas podem interagir. Ex.: `{1^:2} | <1^ 2>`, o que reage e resulta em `[1^ 2]`. Contudo, é necessária que as moléculas estejam dentro de um escopo delimitado por parênteses `()`.

É possível também definir um conjunto de moléculas de DNA, o que é chamado de módulo. Estes módulos são usadas para nomear um conjunto de moléculas DNA que são usadas simultaneamente. Isso é muito útil para reuso e modularizarão de código. Para definir um módulo, basta usar a sintaxe `m(p1, p2, ...) = (s1 | s2 | ...)`, em que `m` é o nome do módulo, `p1` e `p2` são os parâmetros de configuração do módulo, e `s1` juntamente com `s2` são as espécies (ou outros módulos) contidas no módulo `m`.

3.4.2 Compilação

Com as moléculas de DNA especificadas, o DSD é capaz de derivar as reações possíveis de acontecer e em qual velocidade. O termo "reações possíveis" refere-se a união das reações especificadas na CRN com as reações indesejadas que podem ocorrer no momento da experimentação (e.g.: a separação de um complexo em fragmentos de DNA de forma espontânea). Esta etapa de traduzir o sistema especificado pela linguagem em comportamento em ambiente real é chamada de compilação. É possível definir o nível de detalhamento da compilação, de forma que quanto mais detalhada, mais reações possíveis de ocorrerem na realidade são simuladas. Entretanto, maior o custo de processamento e, portanto, maior o tempo para se completar a simulação.

O nível de detalhamento de compilação é definido por três parâmetros: o nível de abstração, a existência de reações de vazamento e a presença de reações improdutivas.

Existem 4 níveis de abstração (do mais detalhado para o menos): Detalhado, Finito, Padrão, Infinito. Estes níveis indicam um conjunto de regras o qual é utilizado para derivar as reações a serem consideradas na simulação. Para cada estado do sistema, este conjunto de regras indicam reações que o estado atual pode sofrer, os quais levam o sistema a novos estados, e assim o processo se repete. Quanto maior o nível de abstração, mais reações devem ser simuladas e mais estados devem ser considerados.

Caso a simulação tenha reações de vazamento, um conjunto de reações que representam reações indesejadas no sistema é adicionado à CRN que será simulada. Estas reações indesejadas são, por exemplo, uma separação espontânea entre dois domínios os quais não são *toeholds* (podendo resultar na separação de dois fragmentos).

Reação improdutiva é definida como uma reação que não acarreta em um deslocamento de cadeia completo. Um exemplo de reação com esta característica é dada pelas duas moléculas $a =$ `{t^*}[x]` e $b =$ `<t y>`. Os *toeholds* `t` e `t*` são hibridizados mas o fragmento `y` de b não é capaz de separar o fragmento `<x>` de a, já que ele não possui o domínio necessário. A representação gráfica desta reação é mostrada pela Figura 3.20. Caso a configuração de reações improdutivas estiver ativa, este tipo de reação será simulada.

Figura 3.20: Exemplo de reação improdutiva. Imagem retirada de Lakin et al. [2012].

3.4.3 Simulação

No momento de simular um circuito é possível escolher entre dois modos de compilação: saturação (SAT) e o modo *just-in-time* (JIT). Dado o conjunto inicial de espécies, o modo SAT deriva todas as reações que possam existir no sistema antes de começar a simulação de fato.

Já o modo JIT deriva inicialmente apenas as reações que estão diretamente relacionadas com as espécies iniciais. Com isso, de acordo com algum método de simulação estocástico, é feito a escolha de qual reação será simulada. A simulação de uma dada reação consome reagentes e gera produtos, e estas modificações são aplicadas no estado atual do sistema, o qual entra em um novo estado. Um novo conjunto de reações é computado a partir do novo estado do sistema. Portanto, no modo JIT a compilação das reações e a simulação do sistema são feitas de forma intercalada.

Também existem dois métodos de simulação: o determinístico e o estocástico. O método determinístico computa o sistema de EDOs que representa a CRN a ser simulada, assim como foi demonstrado no capítulo anterior.

Já o método estocástico utiliza de um modelo probabilístico para modelar a interação entre moléculas de DNA a nível de indivíduos, e não mais de concentrações. Com isso, se tem uma simulação muito mais detalha da interação entre as moléculas de DNA. O modo de compilação JIT suporta apenas este método de simulação já que o método determinístico precisa que todas as reações tenham sido computadas para se gerar o sistema de EDOs.

Capítulo 4

DNAr – Simulando Redes de Reações de Larga Escala

Como foi mostrado no Capítulo 2, uma das formas de simular CRN é calcular numericamente as equações diferencias obtidas pela lei da ação das massas. Para fazer isto na linguagem R, uma função que representa estas equações deve ser criada. Esta função deve conter um comando para cada equação diferencial. É possível modelar facilmente CRNs mais simples, como é mostrado no algoritmo 2.3 (em que se modela uma reação unimolecular de um produto), ou até mesmo a CRN Lotka-Volterra, como é mostrado no Algoritmo 4.1.

```
1  fx <- function(t, y, pars) {
2      with(pars, {
3          dy  <- numeric(2)    # X_1, X_2
4          X1 = y[1]            # X_1
5          X2 = y[2]            # X_2
6          J1 = k1 * X1 * X2    # J_1 = k_1X_1X_2
7          J2 = k2 * X1         # J_2 = k_2X_1
8          J3 = k3 * X2         # J_3 = k_3X_2
9          dy[1] = -J1 + J2     # dX_1/dt = -J_1 + J_2
10         dy[2] = J1 - J3      # dX_2/dt = J_1 - J_3
11         list(dy)
12     })
13 }
```

Algoritmo 4.1: Função modelo da CRN Lotka-Volterra.

Entretanto, esta abordagem de simulação não é escalável. Para CRNs mais com-

plexas, com dezenas de reações, a função modelo que representa o sistema de equações diferencias começa a se tornar muito grande e suscetível a erros de programação que podem tomar muito tempo para serem corrigidos.

Este problema ainda é agravado quando se deseja implementar uma CRN em DNA. Com a abordagem de transformação de CRNs formais para CRNs baseadas em DNA (Seção 3.3) utilizada neste trabalho, o número de reações totais da CRN dobra (em reações unimoleculares) ou até mesmo triplica (em reações bimoleculares). Observe o Algoritmo 4.2, que mostra a CRN Lotka-Volterra em DNA, para se ter uma ideia de como a função modelo pode crescer quando uma CRN formal é transformada para uma baseada em DNA.

```
fx <- function(t, y, pars) {
    with(pars, {
        X1  <- y[1];   L1  <- y[2];   H1  <- y[3]
        B1  <- y[4];   X2  <- y[5];   O1  <- y[6]
        T1  <- y[7];   LS2 <- y[8];   HS2 <- y[9]
        BS2 <- y[10];  G2  <- y[11];  O2  <- y[12]
        T2  <- y[13];  G3  <- y[14];  O3  <- y[15]
        dy<-numeric(15)
        dy[1]  <- - q1*X1*L1   + qmax*H1*B1
                  - q2*X1*G2   + 2*qmax*O2*T2
        dy[2]  <- - q1*X1*L1   + qmax*H1*B1
        dy[3]  <-   q1*X1*L1   - qmax*H1*B1
                  - qmax*X2*X1
        dy[4]  <-   q1*X1*L1   - qmax*H1*B1
        dy[5]  <- - qmax*X2*H1 + 2*qmax*O1*T1
                  - qs2*X2*LS2 + qmax*HS2*BS2
                  - q3*X2*G3
        dy[6]  <-   qmax*X2*H1 - qmax*O1*T1
        dy[7]  <- - qmax*O1*T1
        dy[8]  <- - qs2*X2*LS2 + qmax*HS2*BS2
        dy[9]  <-   qs2*X2*LS2 - qmax*HS2*BS2
        dy[10] <-   qs2*X2*LS2 - qmax*HS2*BS2
        dy[11] <- - q2*X1*G2
        dy[12] <-   q2*X1*G2   - qmax*O2*T2
        dy[13] <- - qmax*O2*T2
        dy[14] <- - q3*X2*G3
        dy[15] <-   q3*O3
        list(dy)
    })
}
```

Algoritmo 4.2: Função modelo da CRN Lotka-Volterra baseada em DNA.

Portanto, a abordagem de simulação de CRNs deve ser escalável para comportar CRNs com dezenas ou, até mesmo, centenas de reações. Para isso, este trabalho propõe o desenvolvimento de um pacote em R, chamado DNAr, para simulação de reações químicas formais e as baseadas em DNA.

O pacote DNAr[1] foi desenvolvido para simplificar a simulação de reações químicas formais e as baseadas em DNA. No pacote, basta descrever a CRN textualmente, juntamente com seus parâmetros, para simular a mesma. Ou seja, não há a necessidade de especificar a função modelo (com o sistema de equações diferencias – e.g.: Algoritmo 4.2).

Além disso, existem funções no pacote que facilitam a comparação de comportamentos entre CRNs, geração de gráficos e também agrega recursos para exportar e importar CRNs. Suas funcionalidades principais são descritas nas seções seguintes.

4.1 Simulando reações formais

Para simular uma CRN formal no DNAr, os seguintes parâmetros são necessários:

- um vetor contendo as espécies existentes na CRN;
- a concentração inicial de cada espécie;
- vetor com as reações da CRN;
- constante de velocidade de cada reação;
- o intervalo de tempo em que a CRN deve ser simulada.

Utilizando como exemplo a reação $A \xrightarrow{k} B$. Os parâmetros usados nesta reação são os mesmos utilizados no exemplo da subseção 2.3.1. Ou seja, $[A]_0 = 1\text{E}{-5}\ M$, $[B]_0 = 0\ M$, $k = 5\text{E}{-5}\ s^{-1}$ e 72.000 segundos de tempo de simulação. Para simular esta CRN no DNAr, basta utilizar o Algoritmo 4.3.

A variável `b` representa o comportamento das espécies no tempo. Ou seja, é uma tabela em que cada linha possui a concentração de cada espécies em um ponto no tempo. Como somente foram definidos 10 pontos no tempo (igualmente distribuídos entre 0 e 72.000 segundos), a tabela possui 10 linhas.

Para visualizar graficamente o comportamento das espécies, basta utilizar o Algoritmo 4.4, o que resulta no gráfico mostrado na Figura 4.1.

[1]Disponível em `https://github.com/DanielKneipp/DNAr`

```
1  library(DNAr)  # Carregando o DNAr
2
3  # Simulando
4  b <- react(
5      species   = c('A', 'B'),
6      ci        = c(1e-5, 0),
7      reactions = c('A -> B'),
8      ki        = c(5e-5),
9      t         = seq(0, 72000, length.out = 10)
10 )
```

Algoritmo 4.3: Simulando uma reação no DNAr.

```
1  plot_behavior(
2      b,
3      x_label = 'Tempo (s)',
4      y_label = 'Concentração (M)',
5      legend_name = 'Espécies'
6  )
```

Algoritmo 4.4: Simulando uma reação no DNAr.

Para mais informações sobre as funções presentes no DNAr e seus respectivos parâmetros, basta usar o comando `?<nome-da-função>` ou `help(<nome-da-função>)`.

Já para CRNs mais complexas, como a Lotka-Volterra, por exemplo, mais de uma reação (juntamente com suas constantes de velocidade) devem ser especificadas. Seguindo a CRN especificada na subseção 2.2.1, o Algoritmo 4.5 mostra como isso é feito.

Note que na linha 7, três valores são definidos. São estes as constantes de velocidade das reações da CRN, em ordem com as reações definidas nas linhas 4 – 6. Da mesma forma, as concentrações inicias das espécies (linha 3) são definidas seguindo a ordem das espécies na linha 2.

Com o DNAr é possível definir e modificar a CRN apenas alterando sua descrição textual e parâmetros, não sendo mais necessário se preocupar com as equações diferenciais que representam a CRN. Isto é possível por conta da modelagem matemática vetorial que o DNAr usa internamente para representar as CRNs, permitindo que o algoritmo usado não mude com a alteração da CRN a ser simulada. Esta modelagem é descrita com mais detalhes a seguir.

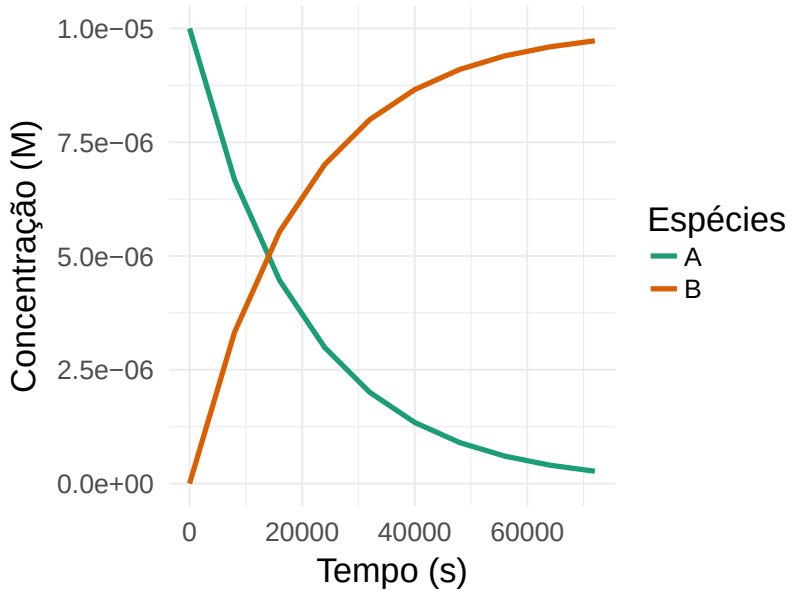


Figura 4.1: Exemplo de visualização gráfica no DNAr utilizando a função `plot_behavior()`.

```
react(
    species   = c('X',   'Y',   'P'),
    ci        = c(20e-9, 10e-9, 0),
    reactions = c('X -> 2X',
                  'X + Y -> 2Y',
                  'Y -> P'),
    ki        = c(1/300, 5e5, 1/300),
    t         = seq(0, 9000, 1)
)
```

Algoritmo 4.5: Simulando o Lotka-Volterra no DNAr.

4.2 Modelagem vetorial das EDOs

Para mostrar a modelagem vetorial, a seguinte CRN é utilizada como exemplo:

$$A \xrightarrow{k_1} 2B + A; \tag{4.1a}$$

$$2B \xrightarrow{k_2} A. \tag{4.1b}$$

Com isso, duas matrizes P e R são definidas. Cada linha representa uma reação e cada coluna uma espécie. O valor na linha i e coluna j de P representa o número de moléculas da espécie associada à coluna j gerada pela reação i. Analogamente, R possui a mesma semântica, mas com o valor de i, j representando o número de moléculas

consumidas. A equação (4.2) mostra estas duas matrizes para o exemplo, sendo $r1$ e $r2$ as reações (4.1a) e (4.1b), respectivamente.

$$R = \begin{matrix} \\ \text{r1} \\ \text{r2} \end{matrix}\!\!\begin{matrix} \begin{matrix} \text{A} & \text{B} \end{matrix} \\ \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix} \end{matrix}, \qquad P = \begin{matrix} \\ \text{r1} \\ \text{r2} \end{matrix}\!\!\begin{matrix} \begin{matrix} \text{A} & \text{B} \end{matrix} \\ \begin{bmatrix} 1 & 2 \\ 1 & 0 \end{bmatrix} \end{matrix}. \tag{4.2}$$

Fazendo $P - R$ tem-se uma matriz chamada de M, em que cada elemento representa a quantidade de moléculas de uma espécie em uma reação. Ou seja,

$$M = P - R = \begin{bmatrix} 0 & 2 \\ 1 & -2 \end{bmatrix}. \tag{4.3}$$

Além de M, um vetor coluna $\vec{v}$ é definido, o qual contém as expressões que representam a dinâmica cinética das reações de acordo com a lei da ação das massas. Para o exemplo, se define $\vec{v}$ da seguinte forma:

$$\vec{v} = \begin{matrix} \\ \text{r1} \\ \text{r2} \end{matrix}\!\!\begin{matrix} \\ \begin{bmatrix} k_1[A]_t \\ k_2[B]_t^2 \end{bmatrix} \end{matrix}. \tag{4.4}$$

Note que, com o intuito de generalização, os expoentes que elevam as concentrações nas expressões em $\vec{v}$ (1 em $[A]_t$ na reação $r1$ e 2 em $[B]_t$ na reação 2) podem ser obtidos diretamente a partir da matriz R.

Com M e $\vec{v}$ definidos, tem-se que as EDOs da CRN podem ser obtidas da seguinte forma:

$$\begin{bmatrix} \frac{d[A]}{dt} \\ \frac{d[B]}{dt} \end{bmatrix} = M^T \times \vec{v} = \begin{bmatrix} 0 & 1 \\ 2 & -2 \end{bmatrix} \times \begin{bmatrix} k_1[A]_t \\ k_2[B]_t^2 \end{bmatrix} = \begin{bmatrix} k_2[B]_t^2 \\ 2k_1[A]_t - 2k_2[B]_t^2 \end{bmatrix}. \tag{4.5}$$

Este tipo de modelagem facilita consideravelmente o algoritmo que calcula os valores das equações diferencias em um ponto no tempo (função modelo `fx()`, e.g.: Algoritmo 4.2) pelo fato de ser generalista e poder ser usado em diversas CRNs.

Também é possível obter um ganho de desempenho utilizando esta modelagem já que o mesmo separa cálculos que precisam ser feitos iterativamente de expressões que precisam ser calculadas apenas uma vez. A matriz M é dependente apenas da especificação da CRN e, portanto, não precisa ser calculada na função modelo. Além

disso, a obtenção dos valores das diferencias por meio de uma multiplicação matricial tira proveito da vetorização de código que R aplica, o que acelera o cálculo em vetores e matrizes [Ross, 2014].

4.3 Simulando CRNs baseadas em DNA

Até o momento, apenas a simulação de CRNs formais no DNAr foi apresentada. Sem a utilização do pacote, para simular CRNs baseadas em DNA seria necessário converter a CRN formal para outra. Esta representaria as reações em DNA que emulariam o comportamento da CRN formal em questão.

Já utilizando o DNAr, esta CRN em DNA não precisa ser especificada, ela é automaticamente gerada a partir da CRN formal (lembrando que a abordagem descrita por Soloveichik et al. [2010] é a utilizada para converter as CRNs formais para reações entre DNA)

Para demonstrar como isso é feito, a reação bimolecular $A + B \xrightarrow{k} C$ é utilizada (CRN em DNA que representa esta reação é mostrada na equação (3.6)). O Algoritmo 4.6 mostra como simular esta reação em DNA com o DNAr.

```
1  r <- react_4domain(
2       species   = c('A', 'B', 'C'),
3       ci        = c(1e-6, 1e-6, 0),
4       reactions = c('A + B -> C'),
5       ki        = c(1e2),
6       qmax      = 1e6, cmax = 1e-4,
7       alpha     = 1,    beta = 1,
8       t         = seq(0, 72000, 10)
9  )
```

Algoritmo 4.6: Simulando uma reação bimolecular em DNA no DNAr.

`r` é um objeto contendo a CRN que de fato foi simulada (CRN de DNA), juntamente com seu comportamento. Para ter acesso às reações de dna, por exemplo, basta usar o comando `r$reactions`. Este vai retornar:

```
"A + L1 -> H1 + W1"    "H1 + W1 -> A + L1"
"B + H1 -> O1"         "O1 + T1 -> C"
```

Os parâmetros `qi` e `qmax` são adicionadas por conta de serem necessários para o uso da abordagem de Soloveichik et al. [2010]. Já `alpha` e `beta` são parâmetros que

reescalam a CRN (constantes de velocidades e concentrações iniciais das espécies) para que esta se torne factível enquanto se mantém o mesmo comportamento ao longo do tempo. `alpha` e `beta` são mais detalhadamente explicados no trabalho de Soloveichik et al. [2010]. Caso não deseje alterar a escala do circuito, basta manter `alpha = 1` e `beta = 1`.

Partindo para um exemplo mais complexo, o Algoritmo 4.7 mostra como simular a CRN Lotka-Volterra (3.9).

```
r <- react_4domain(
    species   = c('X1',  'X2'),
    ci        = c(20e-9, 10e-9),
    reactions = c('X1 + X2 -> 2X2',
                  'X1 -> 2X1',
                  'X2 -> 0'),
    ki        = c(5e5, 1/300, 1/300),
    t         = seq(0, 9000, 1),
    qmax      = 1e6,  cmax = 10e-6,
    alpha     = 1,    beta = 1
)
```

Algoritmo 4.7: Simulando o Lotka-Volterra no DNAr.

As reações que de fato são simuladas (mostradas pelo comando `r$reactions`) são as seguintes:

```
"X1 + L1 -> H1 + W1"    "H1 + W1 -> X1 + L1"    "X2 + H1 -> O1"
"O1 + T1 -> 2X2"        "X1 + G2 -> O2"         "O2 + T2 -> 2X1"
"X2 + G3 -> O3"         "X2 + LS2 -> HS2 + WS2" "HS2 + WS2 -> X2 + LS2"
```

Note que o conjunto de reações que corrigem efeitos de *buffer* são, por padrão, adicionados à CRN a ser simulada. As duas últimas reações corrigem o efeito de *buffer* sofrido por esta CRN.

Para mais exemplos de CRNs de DNA sendo simuladas, acesse a documentação da função com o comando `?react_4domain`.

4.4 Exportando CRNs para o Microsoft Visual DSD

Uma das principais funcionalidades do DNAr é a possibilidade de exportar a CRN formal especificada em uma CRN de DNA para ser simulada no Microsoft Visual DSD.

A função usada para isto é a `save_dsd_script()` e o *script* exportado é ser dividido nas seguintes seções:

- **Cabeçalho**: contém configurações gerais de simulação, como período a ser simulada, método usado (estocástico ou determinístico), além do tipo de compilação e configurações para geração de gráficos. Por padrão, o *script* gerado utiliza a simulação determinística, compilação SAT e nível de detalhamento Infinito.

- **Módulos de reações**: possui a implementação do conjunto de reações mais usadas no formato de módulos. Nesta seção são definidas as espécies e reações de DNA que se comportam como reações formais, seguindo a abordagem de Soloveichik et al. [2010]. A Tabela 4.1 mostra o conjunto de reações formais implementadas em DNA:

  Tabela 4.1: Reações que podem ser exportadas para Visual DSD.

Unimolecular	**Bimolecular**	**Formação**	**Degradação**
$A \longrightarrow B$	$A + B \longrightarrow C$	$\emptyset \longrightarrow A$	$A \longrightarrow \emptyset$
$A \longrightarrow B + C$	$A + B \longrightarrow C + D$	$\emptyset \longrightarrow A + B$	$A + B \longrightarrow \emptyset$
$A \longrightarrow B + C + D$	$A + B \longrightarrow C + D + E$	$\emptyset \longrightarrow A + B + C$	

  Relembrando, reações de degradação são aquelas que geram produtos os quais não influenciam no comportamento da CRN e podem ser ignorados. Já reações de formação servem como uma representação para a formação constante de produtos. Por exemplo, na reação $\emptyset \xrightarrow{k} A + B$, A e B possuem o seguinte comportamento: $d[A]/dt = d[B]/dt = k$. Ou seja, as concentrações das espécies crescem linearmente a uma velocidade constante k.

  Esta seção é sempre incluída nos *scripts* gerados de forma estática, ou seja, não importa quais reações são usadas, todas as reações da Tabela 4.1 serão exportadas. Por isso, mesmo CRNs simples terão um *script* em Visual DSD consideravelmente grande.

- **Parâmetros da CRN**: as configurações específicas da CRN são definidas nesta seção. Parâmetros como concentração inicial das espécies e valores das constantes de velocidade são definidos nesta seção do *script*. Além disso, as moléculas de DNA que representam espécies da CRN formal são renomeadas de acordo com o nome das mesmas na CRN. Isso permite referenciar estas moléculas mais rapidamente utilizando apenas o nome das mesmas (o que é feito na seção de Cabeçalho, para gerar uma gráfico do comportamento das espécies).

- **Especificação da CRN**: esta seção é delimitada por parênteses, onde é especificado quais módulos (definidos na seção de Módulos de reações) serão usados na CRN de DNA simulada. Estes módulos são instanciados (semelhante à funções), passando como parâmetro para os mesmos os domínios de cada molécula de DNA que compõem tais módulos.

Para exemplificar esta funcionalidade do pacote, a CRN formal especificada no Algoritmo 4.3 é utilizada. Para obter a CRN de DNA que representa esta CRN formal, os parâmetros $q_{max} = 1\text{E}6\ M^{-1}s^{-1}$ e $C_{max} = 1\text{E}{-4}\ M$ são definidos, juntamente como $\alpha = \beta = 1$. Lembrando que o comportamento desta CRN formal é mostrado na Figura 4.1.

Além dos parâmetros da função `react_4domain()`, `save_dsd_script()` exige que o nome do arquivo de saída seja especificado. O uso desta função é mostrado no Algoritmo 4.8.

```
1  save_dsd_script(
2      species   = c('A', 'B'),
3      ci        = c(1e-5, 0),
4      reactions = c('A -> B'),
5      ki        = c(5e-5),
6      qmax      = 1e6, cmax = 1e-4,
7      alpha     = 1,    beta = 1,
8      t         = seq(0, 72000, length.out = 10),
9      filename = 'script.dsd'
10 )
```

Algoritmo 4.8: Simulando uma reação bimolecular em DNA no DNAr.

A execução deste Algoritmo cria um arquivo chamado `script.dsd`, o qual possui o seguinte conteúdo:

- **Cabeçalho**:

```
directive duration 72000 points 1000
directive simulation deterministicstiff
directive concentration M
directive compilation infinite
directive plot A(); B()
```

No trecho de código acima são definidos a duração da simulação, o método determinístico para simulação (`deterministicstiff` é um dos métodos determinísticos existentes no Visual DSD) e o tipo de compilação infinita. Além disso,

configurações para geração de gráficos são definidas. São estas a unidade de medida de concentração (M) e as espécies a serem mostradas no gráfico (A e B, no caso).

- **Parâmetros da CRN**:

```
def Cmax = 1e-04
def qmax = 1e+06
def CiA = 1e-05
def CiB = 0.0
def k1 = 0.5
def A() = (Signal(d1, d2, d3, d4))
def B() = (Signal(d5, d6, d7, d8))
def CiA = CiA/1.0
def CiB = CiB/1.0
```

  As 5 primeiras linhas definem as concentrações iniciais e constantes de velocidade usadas. Logo depois, as estruturas que representam as espécies da CRN formal são nomeadas. Estas estruturas são obtidas a partir do módulo `Signal`, o qual é definido como `def Signal(unk, i1, i2, i3) = <unk i1^ i2 i3^>`. `unk`, `i1`, `i2` e `i3` são nomes de domínios (como os nomes `t` e `x` da Figura 3.20). Estes são passados como parâmetros para o módulo no momento que o mesmo é instanciado. Ou seja, `A()` representa a estrutura `Signal()` com os domínios `d1`, `d2`, `d3` e `d4`.

  Já as duas últimas linhas fazem uma alteração nas concentrações das espécies formais já definidas. Isso se faz necessário pelo fato de uma mesma espécie poder participar de diferentes reações ao mesmo tempo. Quando isso ocorre, a espécie é definida em mais de um módulo na seção seguinte e sua concentração é divida igualmente para cada um dos módulos. Caso isso não fosse feito, a concentração inicial da espécie seria $[A] \times nA$, em que $[A]$ é a concentração desejada para A e nA é o número de módulos em que A é usado.

- **Especificação da CRN**:

```
(
A_e_B(
  k1, qmax, CiA, CiB, Cmax,
  d1, d2, d3, d4,
  d5, d6, d7, d8
)
)
```

  Como já foi dito anteriormente, esta é a seção onde as reações que representam a CRN a ser simulada são definidas. Como a CRN formal possui apenas a reação $A \xrightarrow{k} B$, somente o módulo `A_e_B()` (A *equals* B) foi instanciado. Os primeiros parâmetros de qualquer módulo são as configurações do mesmo (concentrações

iniciais das espécies associadas ao módulo, juntamente como a constante de velocidade do mesmo). Logo após os nomes dos domínios são passados, primeiramente os domínios de A, depois os de B.

A seção Módulos de reações não foi mostrada por ser um código extenso que independe da CRN a ser simulada.

A Figura 4.2 mostra o resultado da simulação do *script* gerado. À direita é mostrado o comportamento das duas espécies, em que `Signal` é a espécie A, e `Signal_1` representa B.

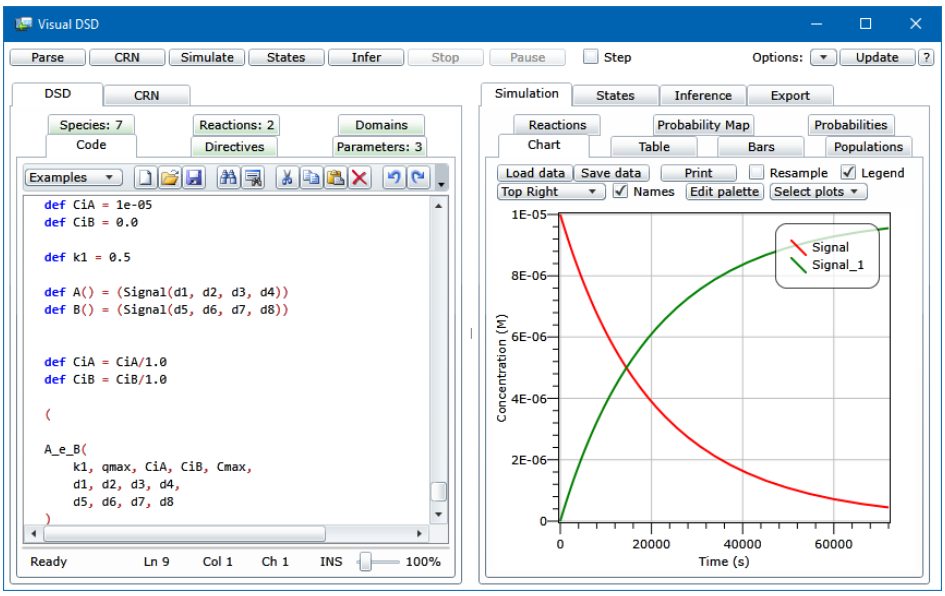


Figura 4.2: Interface inicial do Visual DSD mostrando o resultado da simulação do *script* gerado pelo DNAr para a reação $A \xrightarrow{k} B$.

A Figura 4.3 mostra um painel no Visual DSD em que é apresentado para o usuário as reações simuladas. Veja que não foi apenas a reação formal $A \xrightarrow{k} B$ que foi simulada, mas sim as reações (3.5) em DNA que representam esta reação de acordo com a abordagem de Soloveichik et al. [2010].

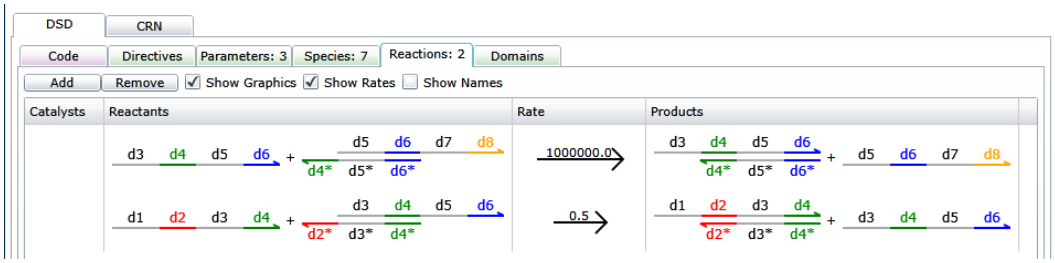


Figura 4.3: Reações simuladas a partir do *script* gerado.

Na Figura 4.4 é possível observar a sequência de nucleotídeos indicada pelo Visual DSD para cada domínio que compõe os fragmentos de DNA.

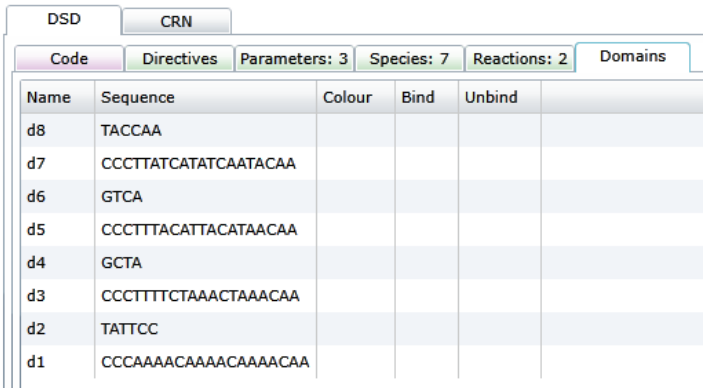


Figura 4.4: Sequência de nucleotídeos dos domínios dos fragmentos.

4.4.1 Limitações

É importante deixar claro que esta funcionalidade possui suas limitações. Não é qualquer CRN que funciona no Visual DSD quando exportada. Algumas CRNs simples foram testadas para validar sua aplicabilidade prática, e uma CRN que falhou ao ser simulada no Visual DSD foi a própria Lotka-Volterra. Ainda por motivos desconhecidos, quando esta CRN é exportada pelo `save_dsd_script()` (utilizando os mesmos parâmetros do Algoritmo 4.7), a mesma possui o comportamento errôneo mostrado na Figura 4.5.

Entretanto, quando os comandos `directive reltolerance 1E-12` e `directive abstolerance 1E-12` são adicionados na cabeçalho do *script*, a simulação passa a ter o comportamento mostrado na Figura 4.6. Os valores padrão dos dois parâmetros são $1E{-}3$ e $1E{-}6$, respectivamente.

Isto demonstra que não há garantias de que o *script* gerado tenha o comportamento esperado no Visual DSD, mas que, para alguns casos, existem formas de contornar a situação problemática.

4.5 Comparação entre DNAr e o Visual DSD

Apesar do DNAr e o Visual DSD serem capazes de simular circuitos baseados em DNA, deve estar claro que o DNAr não foi desenvolvido para substituir o Visual DSD, mas sim para complementá-lo. O propósito destas ferramentas é:

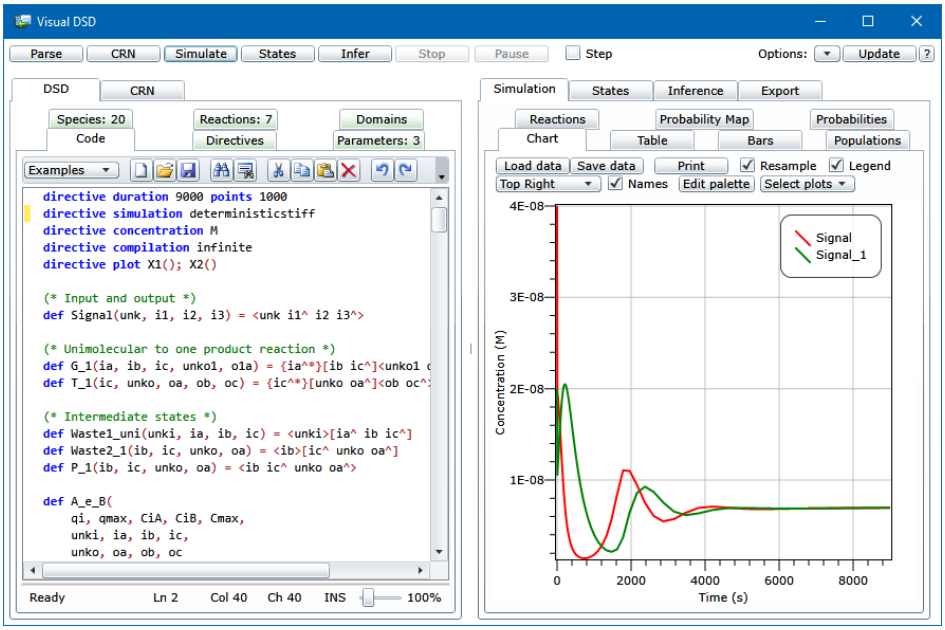


Figura 4.5: Comportamento errôneo do Lotka-Volterra simulado pelo Visual DSD.

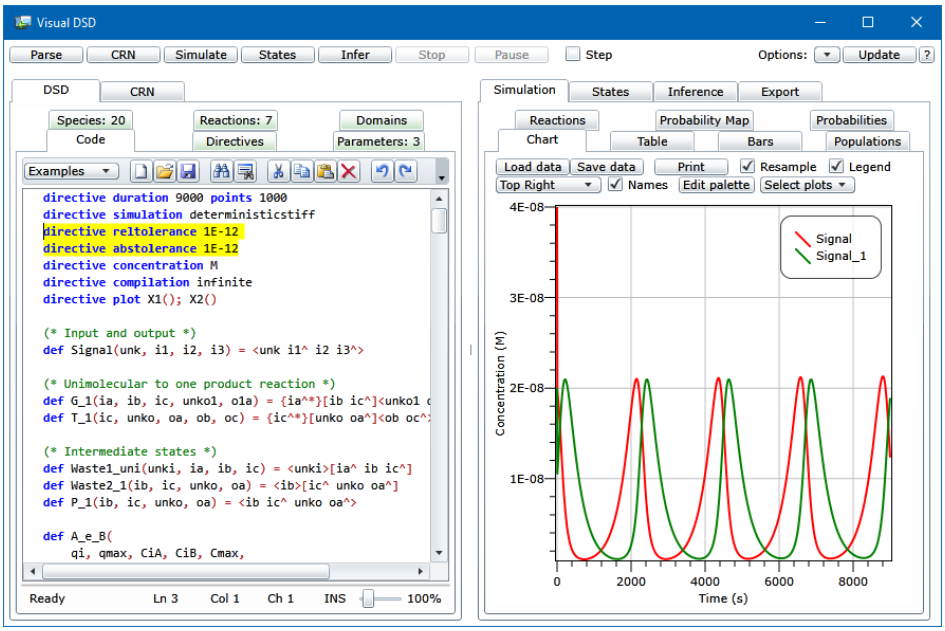


Figura 4.6: Comportamento corrigido do Lotka-Volterra simulado pelo Visual DSD.

- DNAr: facilitar a acelerar a prototipagem e simulação de circuitos baseados em DNA especificados pela linguagem das reações químicas;
- Visual DSD: Permitir a simulação mais detalhada de circuitos de DNA que operam por meio de reações de deslocamento de cadeia.

A interseção entre ferramentas está no fato de que uma das formas de se trans-

formas reações químicas formais em reações entre fragmentos de DNA é por meio de reações de deslocamento de cadeia. Ou seja, existe uma interoperabilidade entre as duas ferramentas.

A Tabela 4.2 comparam as duas ferramentas apresentando alguns pontos que distinguem as mesmas.

Tabela 4.2: Tabela comparativa do DNAr com o Visual DSD.

DNAr	**Visual DSD**
• Prototipagem simples e rápida; • Simula reações químicas formais; • Computacionalmente eficiente; – Capaz de simular circuitos com centenas de reações em questão de minutos. • Possui Ferramentas para análise do comportamento das EDOs; • Código aberto.	• Possui métodos de Simulação mais complexos (simulação estocástica); • Permite obter a sequência de nucleotídeos dos domínios; • Simula reações entre fragmentos de DNA não previstas (reações vazamento e improdutivas).

4.5.1 Fluxo de trabalho com o DNAr e o Visual DSD

A Figura 4.7 mostra um fluxograma que descreve como utilizar o DNAr e o Visual DSD em conjunto. Os passos do fluxo de trabalho são dados pela seguinte sequência:

1. Dada uma CRN, a mesmas é usada como entrada para o DNAr;
2. A função `save_dsd_script()` do DNAr se encarregará de:
   a) Converter a CRN formal para um conjunto de reações entre fragmentos de DNA (recalculando todos os parâmetros necessários);
   b) Gerar um *script* para o Visual DSD com a CRN de DNA obtida.
3. Por fim, *script* gerado pelo DNAr pode ser simulado no Visual DSD.

4.6 Estrutura detalhada do pacote

O pacote é dividido nos seguintes módulos:

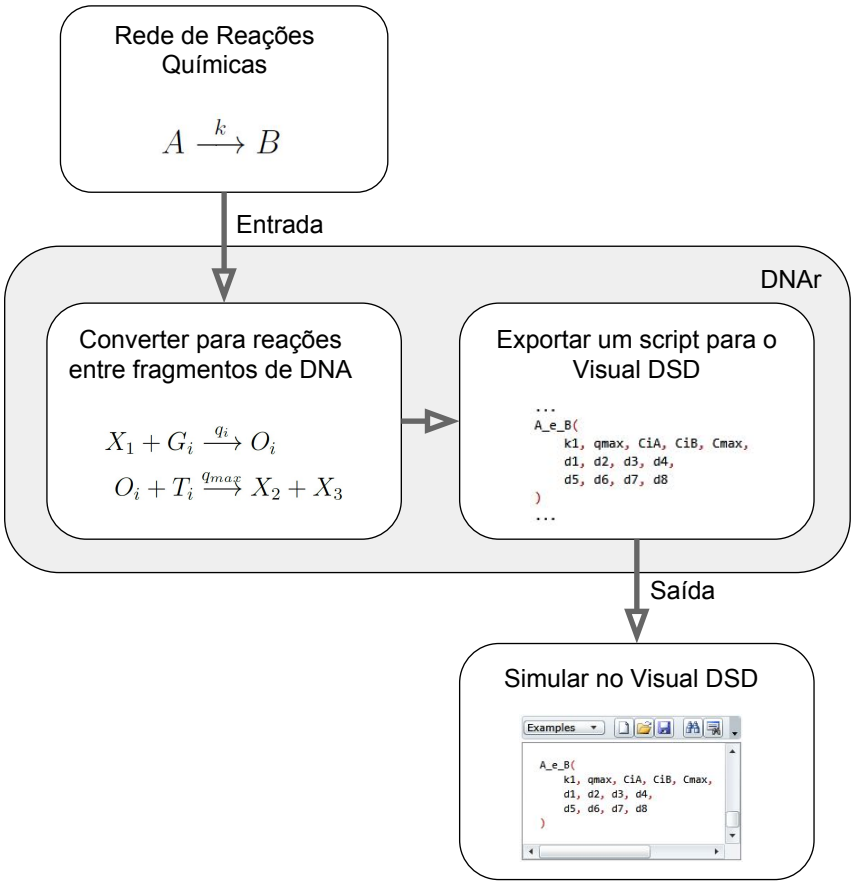


Figura 4.7: Fluxograma de trabalho com o DNAr e Visual DSD

Reator – Composto pelos arquivos `crn_reactor.R` e `4domain_reactor.R`, neste módulo estão contidas a funções diretamente relacionadas às simulações de CRNs (onde estão as funções `react()` e `react_4domain()`).

Parser – Responsável por analisar a CRN passada como parâmetro para as funções de simulação. Para obter algumas informações da CRN, as reações (as quais são passadas como texto) devem ser tratadas. Este módulo contém funções que fazem os tratamentos e manipulações necessárias nos textos, provendo informações como tipo de reações (se a reação é unimolecular ou bimolecular), quantidade de moléculas das espécies em cada reação, além de funções que validam se a CRN foi especificada corretamente. O arquivo `parser.R` contém todas as funções deste módulo.

Análise – Este módulo possui ferramentas criadas para analisar e comparar comportamentos de CRNs. Algumas funções que devem ser destacadas são:

- `analyze_behavior()`: Dada uma CRN, esta função retorna, em formato textual,

as derivadas de cada espécie. Ex.: passando como parâmetro a CRN do Algoritmo 4.3, esta função retorna:

```
d[A]/dt = (-5e-05 * [A])
d[B]/dt = (5e-05 * [A]),
```

o que representa as derivadas de acordo com a lei da ação das massas.

Também é possível passar o comportamento retornado pelas funções de simulação, juntamente com um ponto no tempo. Assim, esta função irá retornar as mesmas derivadas, mas com os valores das concentrações das espécies no ponto especificado em conjunto, facilitando a análise do comportamento da CRN. Após a simulação da CRN do Algoritmo 4.3, estas são as derivadas retornadas no tempo 1:

```
d[A]/dt = (-5e-05 * 1e-05[A])
d[B]/dt = (5e-05 * 1e-05[A]).
```

Note o valor ao lado da notação de concentração das espécies. Já no tempo 2, estas são as derivadas retornadas:

```
d[A]/dt = (-5e-05 * 6.67197603954321e-06[A])
d[B]/dt = (5e-05 * 6.67197603954321e-06[A])
```

- `eval_derivative()`: Passando uma das linhas de texto retornadas por `analyze_behavior()`, esta função retorna o valor da derivada. A derivada passada deve conter os valores das concentrações das espécies. Seguindo o exemplo anterior, para a derivada de `A` no tempo 1, esta função retorna `-5e-10`. Já para o ponto 2, `-3.335988e-10` é retornado.

  Perceba que a derivada de `A` (e `B` também) retornada por `analyze_behavior()` está cercada por parênteses. Caso mais de uma reação influenciasse `A`, sua derivada seria composta por mais sub-expressões separadas por parênteses, de forma que cada sub-expressão representaria a influência que uma reação possui sobre `A`. A função `eval_derivative_part()` calcula o resultado apenas destas sub-expressões, permitindo o usuário analisar o impacto que cada reação tem sobre o comportamento da espécie em um momento no tempo.

- `compare_behaviors_nrmse()`: Com o intuito de comparar os comportamentos de duas CRNs, esta função pode ser usada. Passando como parâmetro dois comportamentos (um chamado de simulado – sim – e outro de observado – obs), ela computa, para cada espécie, o valor de NRMSE (*Normalized Root Mean*

Square Error). NRMSE é definido da seguinte forma:

$$\text{NRMSE} = \frac{\text{RMSE}}{\max\limits_{i\in[n]} [Y_{obs}]_i - \min\limits_{i\in[n]} [Y_{obs}]_i}, \quad [n] = \{1, \ldots, n\}, \tag{4.6}$$

$$\text{RMSE} = \sqrt{\frac{\sum\limits_{i=1}^{n} \left([Y_{obs}]_i - [Y_{form}]_i\right)^2}{n}}, \tag{4.7}$$

em que n é o número de pontos no tempo e i especifica um desses momentos no tempo. $[Y_{obs}]_i$ é a concentração de alguma espécie Y no momento i do comportamento observado, enquanto $[Y_{form}]_i$ é a concentração da espécie Y do comportamento simulado no momento i. Portanto, quanto mais próximo de 0 o valor de NRMSE for, mais próximos os comportamentos estão um do outro.

Utiliza-se uma medida normalizada porque caso uma medida não normalizada (RMSE – *Root Mean Square Error*) fosse utilizada diretamente, a variação da magnitude da concentração inicial das espécies poderia causar uma variação no erro calculado. A utilização de uma medida normalizada torna a comparação entre comportamentos das espécies invariante à magnitude das concentrações das entradas (molar, micro-molar – $M \times 1\text{E}{-6}$, nano-molar – $M \times 1\text{E}{-9}$, etc).

Entrada/Saída – As funções implementadas neste módulo permitem o usuário exportar e importar dados de simulação, sejam elas graficamente ou textualmente. Os arquivos e funções implementadas neste módulo são:

- `io.R`: `plot_behavior()` (já vista anteriormente) mostra graficamente o comportamento da CRN simulada. As funções `save_behavior_csv()` e `save_reactions_txt()` exportam o comportamento em formato `.csv` (*Comma-separated values*) e texto, respectivamente. Já a função `load_behavior_csv()` carrega o arquivo `.csv` exportado por `save_behavior_csv()` para memória;

- `dsd.R`: A função `save_dsd_script()`, a qual exporta um *script* interpretável pelo Microsoft Visual DSD, está implementada neste arquivo.

4.6.1 Principais funções

Em resumo, a Tabela 4.3 descreve sucintamente as principais funções do DNAr que podem ser utilizadas pelo usuário.

Tabela 4.3: Principais funções do DNAr.

Função	Descrição
`react()`	Simula uma CRN formal.
`react_4domain()`	Converte um CRN formal em uma CRN de DNA e simula a mesma.
`analyze_behavior()`	Retorna o sistema de EDOs de uma CRN.
`eval_derivative()`	Calcula uma das derivadas retornadas por `analyze_behavior()`.
`compare_behaviors_nrmse()`	Compara duas simulações usando NRMSE como métrica.
`plot_behavior()`	Gera um gráfico de uma simulação.
`save_behavior_csv()`	Salva o resultado de uma simulação em `.csv`.
`load_behavior_csv()`	Carrega um `.csv` exportado por `save_behavior_csv()` em memória.
`save_dsd_script()`	Exporta um script para o Visual DSD para simular uma CRN de DNA.

Relembrando, o propósito deste pacote é permitir que CRNs de larga escala sejam simuladas rapidamente, bastando apenas que a mesma seja especificada textualmente. Para demonstrar a aplicabilidade deste pacote, a modelagem química teórica de um neurônio desenvolvida por Hjelmfelt et al. [1991] foi escolhida para ser simulada. O próximo capítulo descreve o processo de especificação e simulação de circuitos químicos baseados nesta modelagem.

Capítulo 5

Estudo de Caso: Neurônio Químico

O objetivo deste capítulo é demonstrar, utilizando o pacote DNAr (descrito no capítulo anterior), o estudo e implementação de CRNs complexas. Hjelmfelt et al. [1991] propôs uma série de CRNs que, unidas, se comportam como um modelo de neurônio. Estas CRNs servem como *building blocks* para a construção de circuitos lógicos descritos na Seção 5.3.

Hoje em dia, Rede Neural Artificial (*Artificial Neural Network* – ANN) (rede formada por neurônios artificiais interconectados) é uma modelagem matemática amplamente utilizada nas mais variadas aplicações. Entre elas estão aplicações de classificação de dados [Zhang, 2000; Krizhevsky et al., 2012], reconhecimento de padrões [Han & Cho, 2005; Hoppensteadt & Izhikevich, 2000] e previsão de comportamento baseado em histórico [Zhang et al., 1998; Kim, 2006; Dixon et al., 2016].

Por conta de seu desenvolvimento se basear no conceito de aprendizado, uma ANN não é explicitamente programada para responder a um conjunto específico de dados. Isso permite com que um sistema baseado em rede neural seja capaz de responder de forma esperada à dados de entrada nunca antes vistos (o que é chamado de capacidade de generalização [Jagreet, 2017]). Esta "programação automática" proporcionada pelas redes neurais possibilita o uso da mesma para performar tarefas antes realizadas por humanos [Nielsen, 2015].

O trabalho de Hjelmfelt et al. [1991] modela quimicamente um modelo específico de neurônio, chamado de neurônio de McCulloch–Pitts. Este modelo é descrito com mais detalhes na seção seguinte.

Existem outras abordagens para a construção de neurônios que não utilizam a linguagem das CRNs para serem definidos (especificação direta em DNA) [Qian et al., 2011; Qian & Winfree, 2011a]. No entanto, a camada abstração fornecida pelo uso da linguagem das CRNs permite a especificação de circuitos e dispositivos quaisquer (não

apenas neurônios) baseados em DNA sem a necessidade de um entendimento sobre como os fragmentos de DNA interagem.

5.1 Modelo McCulloch–Pitts

O modelo de neurônio McCulloch–Pitts foi primeiramente introduzido por McCulloch & Pitts [1943], sendo um resultado de um estudo de como o cérebro poderia produzir e identificar padrões complexos utilizando elementos simples interconectados [Marsalli, 2006]. Este elementos, chamados de neurônios, geram um valor de saída com base no que eles recebem como entrada.

O modelo de neurônio de McCulloch–Pitts opera com base em valores binários, ou seja, as entradas I_i e saída o_i de um neurônio i estão contidas no domínio $\{0, 1\}$. A Saída do neurônio é definida com base na soma dos valores das entradas. Caso a soma das entradas seja maior ou igual a um determinado limiar l_i, a saída o_i do neurônio i será igual a 1. No entanto, existe uma entrada especial e_i, chamada de entrada inibitória. Se o valor desta entrada for igual a 1, o_i será 0, independente do valor da outras entradas. Isto é uma das características adotadas do modelo McCulloch–Pitts com o intuito de simplificar o modelo de neurônio humano [Fairhead, 2014]. A Figura 5.1 mostra uma representação gráfica do modelo de neurônio McCulloch–Pitts com 3 entradas.

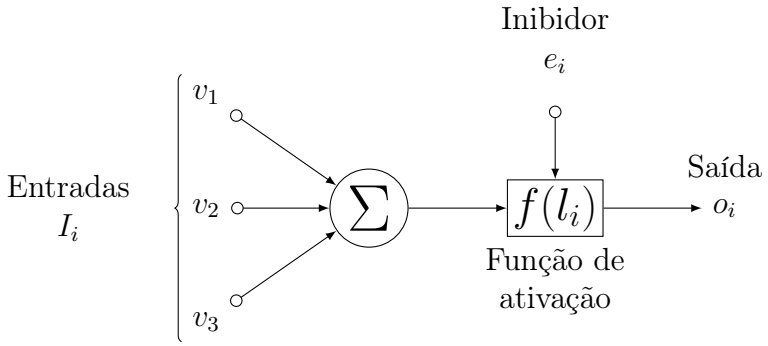


Figura 5.1: Modelo de neurônio McCulloch–Pitts

$f(l_i)$ é definido pela seguinte equação:

$$o_i = f(l_i) = \begin{cases} 1 & : \text{se } \left(\sum_{v \in I_i} v \geq l_i\right) \wedge \left(e_i = 0\right); \\ 0 & : \text{caso contrário.} \end{cases} \tag{5.1}$$

Alguns podem adicionar pesos à modelagem [Beeman, 2001], os quais multiplica as entradas do neurônio. Isso porque duas ou mais entradas podem vir de uma mesma

origem. Por exemplo, v_1 e v_2 da Figura 5.1 podem ser uma replicação de um mesmo valor, o qual pode vir da saída de outro neurônio ou do meio externo à ANN. Isso faz com que v_1 e v_2 possam ser combinadas de forma que em $v_1 + v_2 = 2 \times v_1$.

5.1.1 Construindo Portas Lógicas

Um comportamento interessante que pode ser obtido utilizando este modelo de neurônio é o de portas lógicas. Uma mesma estrutura de neurônio consegue se comportar como uma porta `OR` ou `AND`, bastando apenas alterar o valor do limiar de ativação do neurônio para obter os dois comportamentos.

Para fazer uma porta `AND`, define-se um neurônio com duas entradas e um limiar $l_i = 2$. Os únicos valores de entrada que farão com que v_i seja igual a 1 são $v_1 = v_2 = 1$, já que $v_1 + v_2 = 2 = l_i$. Para construir uma porta `OR`, basta definir $l_i = 1$. Assim, basta apenas uma das entradas ser um para que o neurônio seja ativado.

No entanto, para fazer uma porta `NOT`, um neurônio diferente precisa ser definido. Tendo como objetivo aplicar uma operação de negação sobre uma entrada v_1, defini-se $l_i = 1$, uma entrada no valor de 1 é constantemente aplicada sobre o neurônio, e v_1 é utilizado como entrada inibidora. A Figura 5.2 mostra a representação gráfica do neurônio descrito.

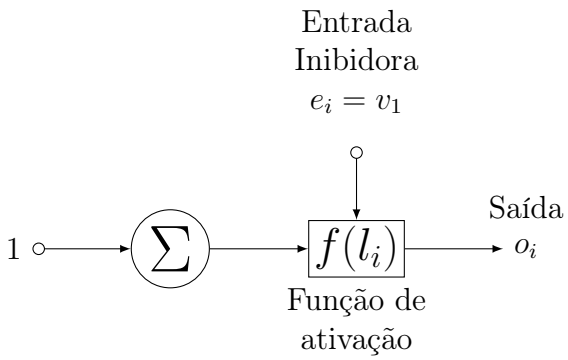


Figura 5.2: Neurônio que opera como uma porta `NOT`.

Caso $v_1 = 0$, o neurônio não é inibido, e seu valor de saída será igual a 1, já que o mesmo possui uma entrada 1 sendo aplicada, o qual é igual ao limiar l_i de 1. Já no caso em que $v_1 = 1$, o neurônio é inibido e sua saída se torna 0.

Com uma porta `AND` seguida de uma porta `NOT` é possível formar uma porta chamada de `NAND`. Esta é conhecida como uma porta universal, o que significa que a mesma pode ser usada para construir quaisquer outras portas [Knight, 2015]. Portanto, com apenas a utilização das estruturas discutidas, é possível modelar comportamentos

complexos descritos com lógica booleana, como somadores e registradores, por exemplo [Fairhead, 2014].

5.1.1.1 *Majority Gate*

A *Majority Gate* – MG (Porta da Maioria) é uma porta lógica com um número de entradas variável e seu comportamento pode ser definido da seguinte forma: esta porta é ativada se e somente se mais da metade das entradas estão ativas. Caso exatamente metade das entradas estejam ativas (quando se tem um número de entradas par), a saída da MG será 0.

Neste trabalho uma MG é definida variando o valor do limiar l_i em função do número de entradas da porta. Supondo uma MG com n entradas, defini-se $l_i = \lfloor n/2 \rfloor + 1$. Assim, para as entradas ativas alcançarem o valor do limiar e, consequentemente, ativarem o neurônio, as mesmas devem ser maioria em relação as entradas inativas.

Exemplo: tendo uma MG com 3 entradas, $l_i = \lfloor 3/2 \rfloor + 1 = 2$, logo, no mínimo duas entradas precisam estar ativas para a saída da MG ser 1. Para 5 entradas, $l_i = 3$, sendo necessário 3 entradas ativas para o acionamento da MG.

5.2 Modelagem Química do Neurônio

Para implementar um neurônio de McCulloch–Pitts, um mecanismo químico precisa ser definido. Ou seja, a CRN deve possuir dois estados, os quais irão representar o valor 0 e 1 (saídas do neurônio).

Um mecanismo que possui esta característica é descrito por Hjelmfelt et al. [1991]. No trabalho dos autores, um neurônio é definido pelas seguintes espécies:

- C: representa a entrada do neurônio. Seu valor vai determinar a saída do mesmo.
- A e B: descrevem o estado do neurônio. Por construção, $A + B = 1$ e, em equilíbrio, o estado das duas espécies são apenas $A = 1,\ B = 0$ e $A = 0,\ B = 1$. São chamadas de espécies de estado.
- X_1 e X_3: reagem diretamente com a entrada e as espécies de estado, intermediando a interação entre as mesmas.
- $I_1^*,\ I_2^*,\ X_2^*,\ X_4^*$: são espécies auxiliares que precisam ter suas concentrações mantidas constantes durante o funcionamento do sistema para que o comportamento do mesmo seja correto.

Com estas espécies, a CRN que possui o comportamento de um neurônio pode ser definida. A primeira reação desta CRN é a que interage com a entrada C. Esta é definida da seguinte forma:

$$I_1^* + C \underset{k_{-1}}{\overset{k_1'}{\rightleftharpoons}} X_1 + C. \tag{5.2}$$

Com esta reação, X_1 passa a representar o valor da entrada no neurônio. Note que C não é consumido na reação. Isto faz com que o circuito construído com a CRN não entre em um estado errôneo onde a entrada estaria igual a 0 (por ter sido consumida) ao mesmo tempo em que o neurônio estivesse ativado (já que entrada estava ativada em um momento anterior).

Com a reação que recebe a entrada definida, agora define-se as reações que computam a entrada com o intuito de gerar a saída desejada. São estas:

$$X_1 + B \underset{k_{-2}'}{\overset{k_2}{\rightleftharpoons}} X_2^* + A; \tag{5.3a}$$

$$X_3 + A \underset{k_{-3}'}{\overset{k_3}{\rightleftharpoons}} X_4^* + B; \tag{5.3b}$$

$$X_3 \underset{k_{-4}'}{\overset{k_4}{\rightleftharpoons}} I_2^*. \tag{5.3c}$$

Pela reação (5.3a), com a existência de X_1, B é consumida para formar A. Ou seja, caso a entrada C do neurônio esteja ativado, A é gerado ao mesmo tempo em que B é consumido, definindo a ligação complementar que as duas espécies tem.

Ao mesmo tempo, A reage com X_3, gerando B novamente (reação (5.3b)). Esta reação garante que, caso a entrada não esteja ativa (não existe X_1) e mesmo assim A esteja presente, a mesma seja consumida ao mesmo tempo em que B é gerado, fazendo com que a CRN entre no estado correto com base no valor atual de sua entrada.

A reação (5.3c) é responsável por gerar X_3. Esta espécie é gerada de forma constante devido ao fato de que a concentração de I_2^* se mantém constante. Note que com esta características que alguma espécies possuem, a CRN (5.3) juntamente com a

reação (5.2) podem ser simplificadas para:

$$C \underset{k_{-1}}{\overset{k_1}{\rightleftharpoons}} X_1 + C; \tag{5.4a}$$

$$X_1 + B \underset{k_{-2}}{\overset{k_2}{\rightleftharpoons}} A; \tag{5.4b}$$

$$X_3 + A \underset{k_{-3}}{\overset{k_3}{\rightleftharpoons}} B; \tag{5.4c}$$

$$X_3 \underset{k_{-4}}{\overset{k_4}{\rightleftharpoons}} \emptyset, \tag{5.4d}$$

de forma que taxas são atualizadas da seguinte forma:

$$k_i = k_i^{'} \times [S^*]; \qquad i \in \{1, -2, -3, -4\}, \tag{5.5}$$

em que S^* representa uma espécie que possui concentração constante. Esta transformação é feita em todas as reações em que S^* participa como reagente.

Os valores das constantes de velocidade das reações definidas por Hjelmfelt et al. [1991] são descritas na Tabela 5.1.

Tabela 5.1: Velocidade das reações. mM significa millimolar, o que representa 1E−3 M.

Constante de Velocidade	Valor
k_1	100 s^{-1}
k_{-1}	1 $mM^{-1}s^{-1}$
k_2	5E4 $mM^{-1}s^{-1}$
k_{-2}	1 s^{-1}
k_3	5E4 $mM^{-1}s^{-1}$
k_{-3}	1 s^{-1}
k_4	1 s^{-1}
k_{-4}	100 mMs^{-1}

Note que as constantes k_{-1}, k_{-2} e k_{-3} e k_4 são consideravelmente menores que k_1, k_2, k_3 e k_{-4}, respectivamente. Isto indica que a influência das reações com estas constantes menores pode ser desprezível. Portanto, uma possível otimização das reações do neurônio seria retirar a característica reversível das reações, mantendo apenas as reações com taxas maiores.

As concentrações iniciais das espécies são dadas na Tabela 5.2.

Com isso, a entrada C ativa o neurônio caso $C > 1,1\ mM$. Com $C < 0,9\ mM$ o neurônio se mantém inativo.

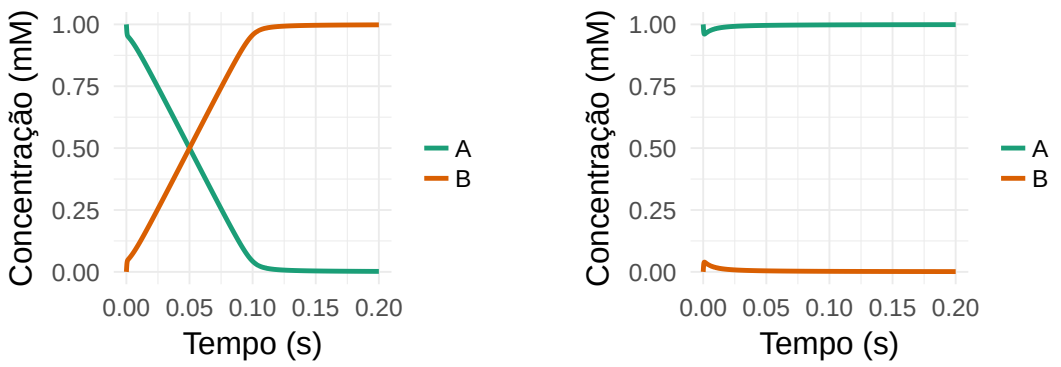
Tabela 5.2: Concentrações inicias das espécies.

Espécie	Concentração inicial
X_1	$0\ mM$
X_3	$0\ mM$
A	$1\ mM$
B	$0\ mM$

Para demonstrar o comportamento desta CRN, a mesma é simulada seguindo o Algoritmo 5.1. Este gera o comportamento mostrado na Figura 5.3.

```
1  b <- reactR(
2      species   = c('C', 'X1', 'X3', 'A', 'B'),
3      ci        = c(input, 0, 0, 1, 0),
4      reactions = c('C -> X1 + C', 'X1 + C -> C',
5                    'X1 + B -> A', 'A -> X1 + B',
6                    'X3 + A -> B', 'B -> X3 + A',
7                    'X3 -> 0', '0 -> X3'),
8      ki        = c(100, 1, 5e4, 1, 5e4, 1, 1, 100),
9      t         = seq(0, 0.04, length.out = 5000)
10 )
```

Algoritmo 5.1: Simulando o comportamento de um neurônio.



(a) Neurônio desativado (`input = 0.9`). (b) Neurônio ativado (`input = 1.1`).

Figura 5.3: Comportamento de um neurônio.

Note que existe uma queda na concentração de A e, por consequência, um aumento na concentração de B, no momento inicial da simulação. Isso é devido ao fato do estado inicial do sistema ser definido com $[A]_0 = 1$ e $[B]_0 = 0$. Isto faz com que a reação (5.4c) cause o consumo de A, ao mesmo tempo em que gera B. Após um

breve instante de tempo, os comportamentos (neurônio ativo e inativo) se divergem por conta da diferença no valor de entrada.

O estado do neurônio é considerado indeterminado quando $0,9\ mM > [C] > 1,1\ mM$ devido ao fato do neurônio não convergir para um estado bem definido nesta situação. A Figura 5.4 mostra o estado de convergência do neurônio (representado pela concentração de A e B) em função da concentração da entrada C. Note que quanto mais próximo $[C]$ está de $1\ mM$, mais próximos $[A]$ e $[B]$ ficam de $0,5\ mM$, fazendo com que o neurônio perca sua característica biestável. $0,9\ mM > [C] > 1,1\ mM$ é considerado um intervalo de transição e, por conta disso, o estado do neurônio não é determinado quando $[C]$ está neste intervalo.

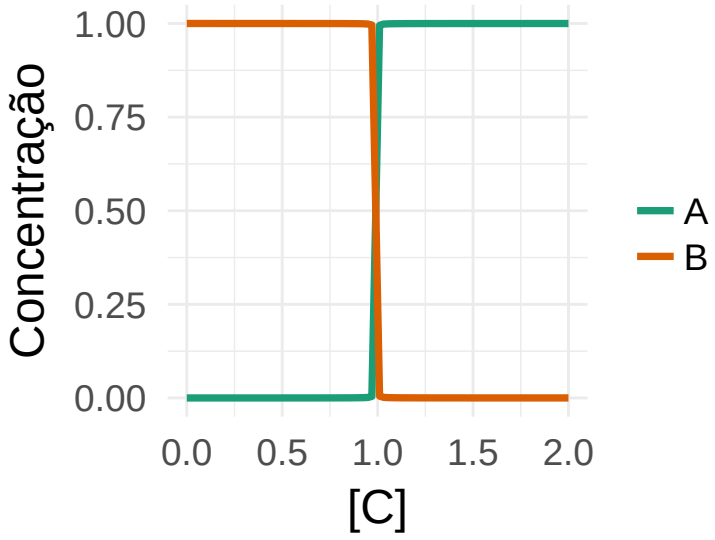


Figura 5.4: Estado de convergência do neurônio em função da entrada $[C]$.

5.2.1 Conectando neurônios

O processo de conexão entre neurônios detalhado por Hjelmfelt et al. [1991] envolve fazer com que um neurônio receba como entrada a saída de um ou mais neurônios. Supondo que se deseja conectar a saída do neurônio j ao neurônio i, sabe-se que A_j ou B_j podem ser usados já que os mesmos representam o estado do neurônio j. Para associar estas espécies de estado com uma entrada do neurônio i, uma etapa intermediária descrita pela reação (5.6) é definida.

$$E_{ij} + A_j \underset{k_{-ij}}{\overset{k_{ij}}{\rightleftharpoons}} C_{ij}, \tag{5.6}$$

em que C_{ij} representa a entrada destinada ao neurônio i vinda do neurônio j. Assume-se que a velocidade da reação (5.6) é mais rápida do que o tempo levado pelos neurônios para entrarem em seu estado de equilíbrio. Ou seja, as taxas k_{-ij} e k_{ij} devem ser consideravelmente maiores que as taxas das reações do neurônio. Também assume-se que a concentração de E_{ij} é menor que a concentração máxima das espécies de estado (A_i ou $B_i = 1$).

Para se determinar as concentrações inicias de C_{ij} e E_{ij}, bem como as constantes de velocidade da reação (5.6), uma equação que relacione estas espécies com A_j (ou B_j) deve ser definida.

É definido

$$E_{ij}^0 = [C_{ij}] + [E_{ij}], \tag{5.7}$$

como sendo a concentração total de C_{ij} somada com E_{ij}. Como E_{ij} é consumido para gerar C_{ij} e vice-versa, este valor permanece constante.

A constante de equilíbrio desta reação é definido como

$$K = \frac{C_{ij}}{[E_{ij}][A_j]}, \tag{5.8}$$

assumindo que A_j esteja sendo associado ao neurônio i, e não B_j. Isolando E_{ij}, tem-se:

$$[E_{ij}] = \frac{[C_{ij}]}{K[A_j]}. \tag{5.9}$$

Combinando as equações (5.7) e (5.9), tem-se que:

$$E_{ij}^0 = \frac{[C_{ij}]}{K[A_j]} + [C_{ij}]; \tag{5.10a}$$

$$E_{ij}^0 = \frac{[C_{ij}] + K[A_j][C_{ij}]}{K[A_j]}; \tag{5.10b}$$

$$E_{ij}^0 = \frac{[C_{ij}](1 \times K[A_j])}{K[A_j]}. \tag{5.10c}$$

Isolando $[C_{ij}]$, é obtida a equação:

$$[C_{ij}] = \frac{E_{ij}^0 K[A_j]}{1 + K[A_j]}; \tag{5.11a}$$

$$[C_{ij}] = \frac{E_{ij}^0}{1 + \dfrac{1}{K[A_j]}}. \tag{5.11b}$$

Esta equação é usada para calcular qual o valor de concentração o neurônio i vai receber como entrada do neurônio j.

A entrada C_i do neurônio i é definida como

$$[C_i] = \sum_{j \in J} [C_{ij}], \tag{5.12}$$

sendo J o conjunto de todos os neurônios que estão conectados ao neurônio i. Uma forma de fazer esta soma de concentrações sem alterar a dinâmica das reações envolvidas é permitir que o neurônio receba diretamente C_{ij} para todos os $j \in J$, fazendo com que, na prática, C_i como uma espécie independente não exista. Em outras palavras, replica-se a reação (5.4a), definindo uma reação $C_j \underset{k_{-1}}{\overset{k_1}{\rightleftharpoons}} X_1 + C_j, \quad \forall j \in J$.

A criação de reações intermediárias deve ser visto com cautela, visto que estas podem alterar o comportamento das reações que precedem e sucedem as mesmas. Por exemplo, a reação $C_{ij} \longrightarrow C_i, \quad \forall j \in J$ transformaria as saídas de todos os neurônios em uma única espécie a qual serviria como entrada para i. O problema é que esta reação é irreversível e consome C_{ij}, alterando o equilíbrio da reação reversível (5.6).

Para se fazer uma conexão de inibição, basta substituir o uso de A_j por B_j. Assim, quando j estiver ativo, a conexão entre i e j estará inativo. Entretanto, quando j estiver inativo, B_j terá uma concentração que representa um sinal ativo, acionando a ligação entre os neurônios.

Perceba que este comportamento é diferente do comportamento de inibição descrito pelo modelo de McCulloch–Pitts. O modelo original descreve que um neurônio possui apenas uma ligação inibitória. Quando a mesma está ativa, o neurônio deve ser desativado, não importando o valor das outras entradas. No modelo químico cada entrada de um neurônio pode ser uma entrada inibitória. Além disso, uma conexão ser inibitória apenas significa que esta possui um comportamento inverso, ou seja, a conexão entre i e j só é ativada quando o neurônio j está desativado, e vice-versa. A conexão inibitória não possui o caráter absoluto de poder inibir a saída do neurônio i no modelo químico, apenas de não contribuir com sua ativação.

5.2.2 Construindo Portas Lógicas

A seguir é descrito como construir uma porta lógica `AND` [Hjelmfelt et al., 1991]. Esta porta recebe duas entradas e gera uma saída. Para construí-la utilizando a modelagem de neurônio químico, é necessário um neurônio que receba duas entradas. O mesmo só pode ser ativado quando estas duas entradas estão ativadas (possuem valor lógico 1).

i é definido como o neurônio que representa a porta `AND`, enquanto que j e k são os neurônios conectados às entradas de i. Portanto, seguindo a equação 5.12, C_i é definido como $C_i = C_{ij} + C_{ik}$. Esta equação pode ser expandida (utilizando a equação (5.11b)) para:

$$[C_i] = \frac{1}{1 + \frac{1}{2[A_j]}} + \frac{1}{1 + \frac{1}{2[A_k]}}, \tag{5.13}$$

com $E^0_{ij} = E^0_{ik} = 1$ e $K = 2$. Note que $E^0_{ij} = 1$ não significa que $[E_{ij}]_0 = 1$. De acordo com a equação (5.7), a única afirmação que pode ser feita é que $[C_{ij}]+[E_{ij}] = 1$. Perceba que com estes parâmetros, em equilíbrio, C_i possui o seguinte comportamento:

- $C_i = 0$ quando nenhuma das entradas está ativa;
- $C_i = 2/3$ quando somente uma das entradas está ativa (o que não é o suficiente para ativar i);
- $C_i = 4/3$ quando as duas entradas estão ativas.

Existem inúmeras possibilidades de valores para $[C_{ix}]_0$ e $[E_{ix}]_0$ que atendam a restrição $[C_{ix}]_0 + [E_{ix}]_0 = 1$ (com $x \in \{j, k\}$). Neste trabalho, o valor estipulado para C_{ix} (construção de uma porta `AND`) é $2/3\ mM$, ou seja, o estado inicial da conexão é ativo. Por consequência, $[E_{ix}]_0 = 1/3\ mM$, respeitando a restrição $[E_{ix}] < 1\ mM$.

Como a constante de equilíbrio K foi definida como 2, as taxas k_{ix} e k_{-ix} presentes na reação (5.6) (lembrando que $x \in \{i, j\}$) devem respeitar a condição $k_{ix}/k_{-ix} = 2$. Ao mesmo tempo, como já foi dito anteriormente, a reação (5.6) deve ser mais rápida que as reações dos neurônios. Para respeitar estas duas condições, as constantes de velocidades foram definidas como $k_{ix} = 2\text{E}5\ mM^{-1}s^{-1}$ e $k_{-ix} = 1\text{E}5\ mM^{-1}s^{-1}$.

A versatilidade do modelo de neurônio aparece quando se faz necessário construir uma porta `OR`. Para tal, basta apenas alterar o valor de E^0_{ij}. As reações e nenhum dos outros parâmetros precisam ser alterados. Com $E^0_{ij} = 2$, tem-se que caso apenas uma conexão esteja ativa, $[C_i] = 4/3\ mM$, ativando o neurônio i. Caso as duas conexões estejam ativas, $[C_i]$ será igual a $8/3\ mM$.

Utilizando conexões inibitórias é possível criar portas com entradas negadas, como é o caso da `AND NOT`. Esta porta funciona como uma `AND`, no entanto, uma de suas entradas é negada (seu valor é invertido). Perceba que este é o mesmo comportamento de uma ligação inibitória no modelo de neurônio químico. Portanto, para se fazer uma porta `AND NOT`, basta conectar a espécie de estado B (ao invés de A) no neurônio i.

Fazendo isso com a entrada vinda do neurônio k, tem-se que o valor de $[C_i]$ é definido por:

$$[C_i] = \frac{1}{1 + \frac{1}{2[A_j]}} + \frac{1}{1 + \frac{1}{2(1 - [A_k])}}, \tag{5.14}$$

em que $[B_k] = 1 - [A_k]$, lembrando que a espécie B é sempre o complemento de A, de forma que $[A] + [B] = 1\ mM$.

Isso faz com que $[C_i]$ seja maior que $1\ mM$ apenas quando o neurônio j está ativo e k está inativo (valores lógicos 1 e 0, respectivamente).

5.2.2.1 *Majority Gate*

Na modelagem teórica da MG em neurônio artificial o limiar de ativação do neurônio é alterado em função do número de entradas. Como isso é mais complexo de se fazer na modelagem química do neurônio artificial, neste trabalho optou-se por manipular a contribuição que uma entrada tem sobre a ativação de um neurônio com limiar constante (lembrando que o neurônio gera uma saída 0 quando possui a entrada $i < 0,9\ mM$ e 1 quando $i > 1,1\ mM$).

Analisando a conexão de entrada j de um neurônio i, é definido $[C_{ij}]$ (presente na equação (5.11b)) como:

$$[C_{ij}] = \frac{2}{n}, \tag{5.15}$$

com n sendo o número de entradas da porta. Este é o valor de uma entrada ativa do neurônio i. Fazendo com que todas as entradas possuam o mesmo valor quanto estão ativas, o neurônio será ativado quando $\lfloor 2/n \rfloor + 1$ entradas estiverem ativas. Esta conclusão pode ser obtida avaliando a situação em que apenas metade das entradas está ativa, ou seja, quando $[C_i] = n/2 \times 2/n = 1$. Caso apenas uma entrada mude de estado, $[C_i]$ irá mudar para um valor menor ou maior que 1 de acordo com o novo estado da entrada alterada. O novo estado se tornará maioria e a saída do neurônio responderá de acordo.

Duas observações devem ser feitas sobre esta abordagem:

1. empate nos valores de entrada fazem com que o neurônio entre em um estado indefinido. Por conta disso, apenas uma quantidade de entradas ímpar é permitida;

2. a medida que n cresce, $[C_{ij}]$ diminui. Isso faz com que a diferença que uma entrada causa no comportamento do neurônio é reduzido. Consequentemente, caso um estado esteja em maioria por apenas uma entrada, é mais fácil detectar o estado em maioria em uma porta com 3 entradas do que uma com 53. Dependendo da quantidade de entradas, algumas configurações de entrada podem fazer com que $[C_i]$ entre em um estado indeterminado ($0,9\ mM \leq [C_i] \leq 1,1\ mM$).

Tendo definido $[C_{ij}]$, tem-se que:

$$E_{ij}^0 = [C_{ij}] \times \left(1 + \frac{1}{2 \times 1}\right), \tag{5.16}$$

seguindo a equação (5.11b), com $K = 2$ e assumindo que a conexão j está ativada.

5.2.2.2 Implementação em DNA

Utilizando o pacote DNAr, uma forma de realizar a implementação em DNA das reações do neurônio químico é descrever textualmente as reações e utilizar a função `react_4domain()`. Esta gera a CRN em DNA automaticamente com base na CRN formal do neurônio. No entanto, mesmo utilizando a descrição textual para facilitar o trabalho, existe um número consideravelmente grande de reações e parâmetros para serem especificados, isto considerando apenas um neurônio. Logo, para vários neurônios conectados, se torna inviável especificar todos os parâmetros e reações manualmente. Por isso, o pacote DNAr fornece funções que facilitam a especificação de neurônios e portas lógicas formadas por neurônios.

O algoritmo 5.2 mostra como simular uma porta `AND` feita por neurônios químicos no DNAr.

```
1  g      <- get_neuron_AND_gate_hje(g_name, c(1.5, 1.5))
2  circ   <- get_circuit_from_neuron_gates_hje(list(g))
3  crn    <- get_crn_from_neuron_circuit_hje(circ)
4  crn$t <- seq(0, 2, length.out = 100)
5  do.call(react_4domain, update_crn_4domain(crn))
```

Algoritmo 5.2: Especificando e simulando uma porta lógica `AND` construida usando um neurônio químico.

A linha 1 define uma CRN de uma porta `AND` formada por neurônios com as entradas $1,5\ mM$ especificadas como parâmetros. Esta função retorna três neurônios conectados, dois deles recebem cada uma das entradas e o outro recebe a saída dos

dois neurônios. A saída deste último neurônio é a saída da porta lógica. Isto é feito para normalizar os valores de entrada recebidos pelo neurônio de saída. Os neurônios de entrada recebem as concentrações de entrada, discretizando-as para valores $\in 0, 1$ (valores possíveis de $[A]$ e $[B]$ em equilíbrio). Caso as entradas fossem conectadas diretamente no neurônio de saída (substituindo $[A_j]$ ou $[A_k]$ na equação (5.13)), apenas uma das entradas poderia colocar o neurônio em um estado indeterminado (lembrando que em uma porta `AND`, deve ser necessário que duas entradas estejam ativadas para tirar a porta do estado de desativada). Utilizando apenas uma entrada como exemplo, suponha que uma entrada $[I]$ substitua $[A_j]$, fazendo uma conexão direta com o neurônio de saída, Quando $[I] \to \infty$, tem-se que:

$$\lim_{[I]\to\infty} \frac{1}{1 + \frac{1}{2[I]}} = 1. \tag{5.17}$$

É seguro dizer que neurônio é ativado quando o mesmo recebe uma entrada de $1,1\ mM$, e é desativado quando a entrada é menor que $0,9\ mM$. Uma concentração muito grande em uma das conexões de um neurônio pode fazer com que a entrada que ele receba seja próxima de $1\ mM$, fazendo com que o mesmo entre em um estado indeterminado.

A Figura 5.5 mostra uma representação gráfica da porta `AND` utilizando os neurônios de entrada E_1 e E_2, juntamente com o neurônio de saída S. As espécies recebidas pelos neurônios de entradas são X_1 e X_2.

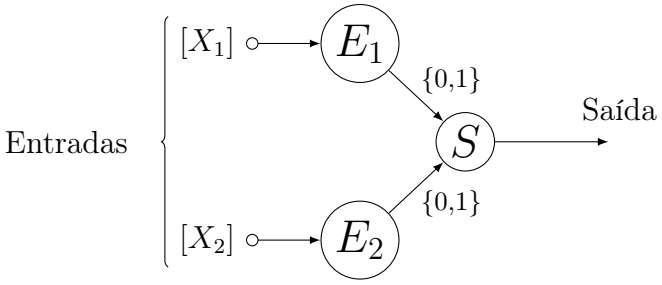


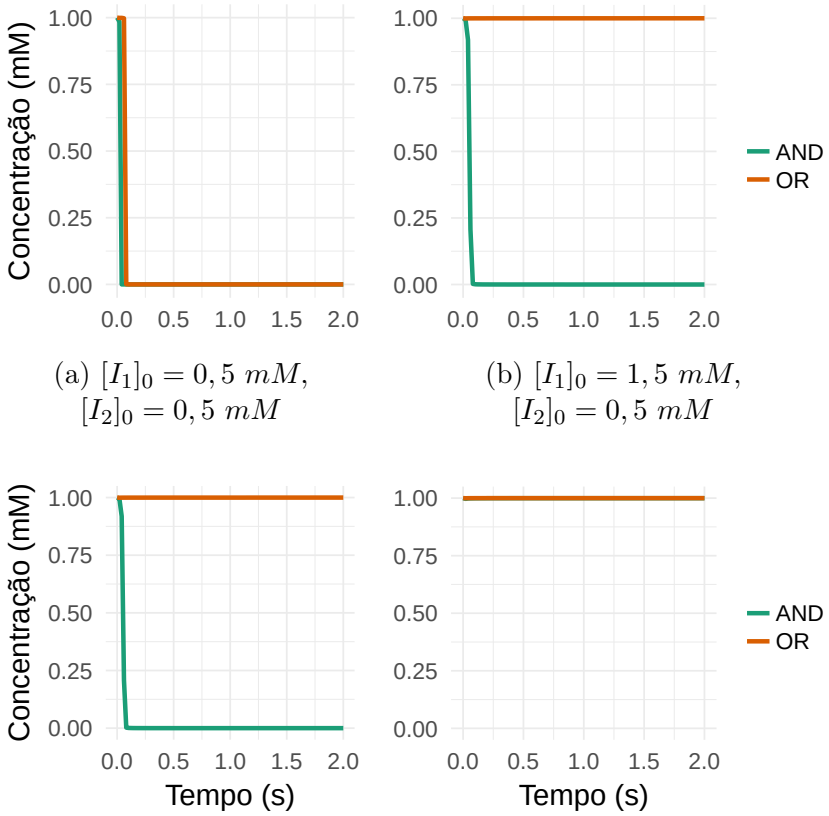
Figura 5.5: Porta `AND` formada por neurônios.

Continuando com a explicação do Algoritmo 5.2, a linha 2 define um circuito formado por uma ou mais portas lógicas. No caso, o circuito é formado por apenas uma porta. A partir de um circuito, uma CRN formal pode ser gerada, e isto é feito na linha 3, juntamente com a linha 4, em que é definido o tempo de simulação.

A função `get_crn_from_neuron_circuit_hje()` retorna a CRN formal do circuito. Caso deseje simular a CRN em DNA, é necessário que os parâmetros α, β, q_{max} e c_{max} sejam definidos. É isso que a função `update_crn_4domain()` faz. Nela é definido

$\alpha = \beta = 1$, $q_{max} = \max_i(k_i) \times 1E4$ e $c_{max} = \max_{s \in S}([s]_0) \times 1E4$, sendo S o conjunto de todas as espécies da CRN. Com isso, a função `react_4domain()` é capaz de gerar a CRN de DNA e simula-la, o que é feito na linha 5.

Para simular uma porta `OR` ao invés de uma `AND`, basta substituir a função `get_neuron_AND_gate_hje()` pela `get_neuron_OR_gate_hje()`. A Figura 5.6 mostra o comportamento simulado das portas `AND` e `OR` com neurônios.



(a) $[I_1]_0 = 0,5\ mM$, $[I_2]_0 = 0,5\ mM$

(b) $[I_1]_0 = 1,5\ mM$, $[I_2]_0 = 0,5\ mM$

(c) $[I_1]_0 = 0,5\ mM$, $[I_2]_0 = 1,5\ mM$

(d) $[I_1]_0 = 1,5\ mM$, $[I_2]_0 = 1,5\ mM$

Figura 5.6: Comportamento das portas lógicas `AND` e `OR` formadas por 3 neurônios (indicado pela concentração de A do neurônio de saída de cada porta). I_1 e I_2 representam as duas entradas das portas lógicas. Note que nas Figuras 5.6a e 5.6d o comportamento das duas portas estão sobrepostos.

Esta porta lógica simples com apenas 3 neurônios gera 30 reações formais, o que em DNA, culmina em 113 reações (isto para a porta `AND` e `OR`, já que as duas possuem a mesma estrutura). Isto comprova a falta de escalabilidade da especificação manual de CRNs e justifica o uso de funções que as geram programaticamente.

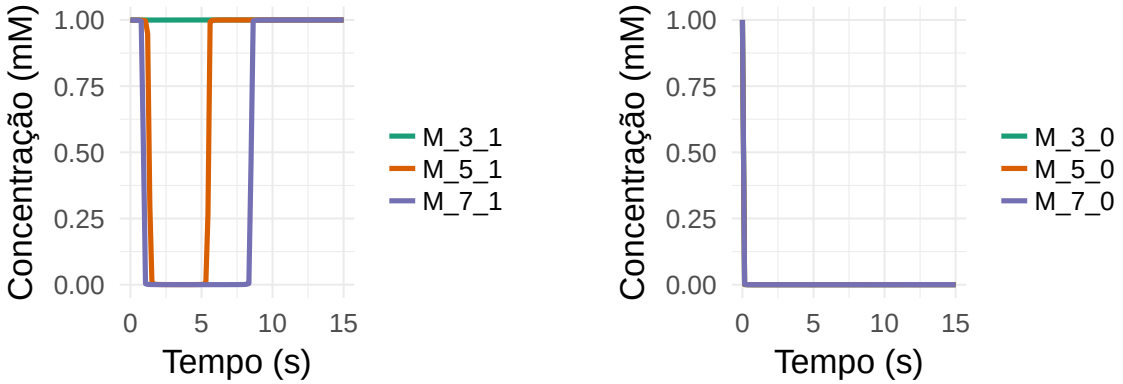
A simulação de uma MG segue o mesmo processo das portas anteriores. Para simular uma MG com 3 entradas basta substituir a primeira linha do Algoritmo 5.2

pela mostrada no Algoritmo 5.3. Esta linha instancia uma MG com 3 entradas, com duas já no estado ativado.

```
1  g <- get_neuron_majority_gate('_MAJ_', 3, c(1.5, 1.5, 0))
```

Algoritmo 5.3: Instanciando e uma MG feita de um neurônio artificial.

A Figura 5.7 mostra o comportamento de uma MG em DNA com 3, 5 e 7 entradas.



(a) Maioria das entradas estão ativadas.

(b) Maioria das entradas estão desativadas.

Figura 5.7: Comportamento de uma MG com 3, 5 e 7 entradas (`M_3`, `M_5` e `M_7`, respectivamente). Em todos os casos o estado em maioria possui apenas uma entrada a mais que o outro. Note na Figura 5.7a o aumento no tempo de resposta para ativação a medida que o número de entradas aumenta. Isto é reflexo da segunda observação feita na subseção 5.2.2.1. Na Figura 5.7b os traçados de `M_3_0` e `M_5_0` estão ocludidos pelo traçado de `M_7_0`.

5.3 Estudo de Caso: Detecção de Células Específicas de Câncer

As subseções anteriores discutiram como implementar um modelo de neurônio por meio de redes de reações químicas. Com isso, uma forma de construir portas lógicas conectando neurônios foi descrita. Esta seção tem como objetivo demostrar um uso das portas lógicas feitas com neurônios químicos. Um circuito lógico formado por estas portas capaz de detectar células cancerígenas é descrito a seguir.

O trabalho de Xie et al. [2011] discute marcadores biológicos que podem ser usados para indicar a presença de células cancerígenas. Caso estes marcadores possuam um perfil de concentração (conjunto de valores de concentração para cada espécie) em uma

célula, é dito que esta célula é do tipo HeLa. HeLa é uma linhagem celular (população de células) humana, e células que participam desta linhagem são classificadas como cancerígenas [Lucey et al., 2009].

Os marcadores usadas para a classificação de células HeLa são: miR-21, miR-17-30a (um marcador comporto pelos miR-17 e miR-30a), miR-141, miR-142(3p) e miR-146a. Estes são referenciados pelas seguintes letras no decorrer do trabalho, respectivamente: A, B, C, D e E.

Com estes marcadores, o circuito lógico para detecção do perfil de concentração dos mesmos é definido como:

$$A \texttt{ AND } B \texttt{ AND NOT}(C) \texttt{ AND NOT}(D) \texttt{ AND NOT}(E). \tag{5.18}$$

A Figura 5.8 apresenta a representação gráfico do circuito (5.18)

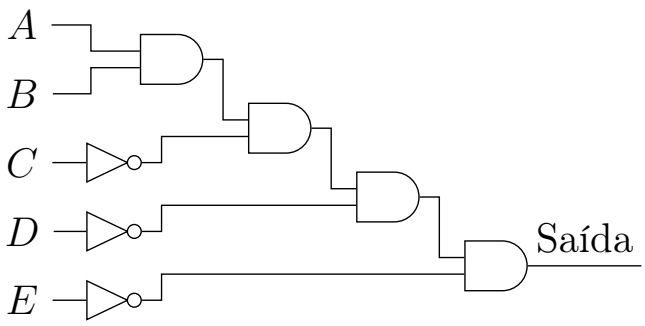


Figura 5.8: Circuito lógico para detecção do perfil de concentração desejado.

Isto significa que o circuito classifica uma célula como sendo uma HeLa caso ela possua uma concentração significativa dos marcadores A e B, ao mesmo tempo em que não existe uma concentração significativa de C, D e E.

Estes marcadores reagem com fragmentos de DNA, logo, é possível utilizá-los como entrada de alguma rede de reações químicas baseada em DNA. Com as portas lógicas baseadas em neurônios químicos, é possível implementar o circuito (5.18). No caso, o limiar de 1 mM define se um valor de concentração é significativo ou não. No entanto, note que as concentrações em que os neurônios operam, bem como suas constantes de velocidade, podem ser alteradas e reescaladas. Ou seja, este limiar de 1 mM pode ser alterado.

Este circuito pode ser definido utilizando o Algoritmo 5.4 [1]. Nas 4 primeiras linhas, as portas lógicas são definidas. O primeiro parâmetro que função `get_neuron_AND_gate_hje()` é o nome da porta (para nomear as espécies da mesma).

[1]Para mais detalhes do processo de implementação dos neurônios químicos com o circuito lógico em questão, veja `https://rpubs.com/danielkneipp/dna-cancer-detector`

O segundo contém as concentrações inicias das entradas (algumas portas estão com uma das entradas com concentração igual a 0 devido ao fato que esta entrada será conectada com a saída de outro neurônio, portanto este valor não influencia o circuito). O terceiro parâmetro especifica quais conexões (entre E_1 e E_2 com S na Figura 5.5) devem se tornar conexões inibitórias (com comportamento inverso). Definindo o parâmetro como `(FALSE, TRUE)` faz com que a segunda entrada seja inibitória. Por padrão, nenhuma conexão é inibitória.

```
1  g1 <- get_neuron_AND_gate_hje('_AND1_', c(A, B))
2  g2 <- get_neuron_AND_gate_hje('_AND2_', c(0, C), c(FALSE, TRUE))
3  g3 <- get_neuron_AND_gate_hje('_AND3_', c(0, D), c(FALSE, TRUE))
4  g4 <- get_neuron_AND_gate_hje('_AND4_', c(0, E), c(FALSE, TRUE))
5
6  g2 <- define_neuron_gate_binding_hje(g1, g2, 1)
7  g3 <- define_neuron_gate_binding_hje(g2, g3, 1)
8  g4 <- define_neuron_gate_binding_hje(g3, g4, 1)
9
10 circ <- get_circuit_from_neuron_gates_hje(list(g1, g2, g3, g4))
11 crn <- get_crn_from_neuron_circuit_hje(circ)
```

Algoritmo 5.4: Definindo uma CRN com base no circuito lógico (5.18).

As linhas 6 – 8 conectam as portas lógicas. A linha 6 conecta a saída porta `g1` (`_AND1_`) à primeira entrada da porta `g2` (`_AND2_`). Com isso, o circuito é definido com base nas portas e, ao final, a CRN é gerada a partir do circuito. `A`, `B`, `C`, `D` e `E` representam os valores de concentrações dos marcadores de mesmo nome.

A CRN de DNA é obtida a partir da CRN do Algoritmo 5.4 (utilizando a mesma função `update_crn_4domain()` mencionada anteriormente), e esta é simulada utilizando as configurações da Tabela 5.3. A critério de informação, a CRN formal possui 126 reações, enquanto que 476 reações constituem a CRN de DNA [2].

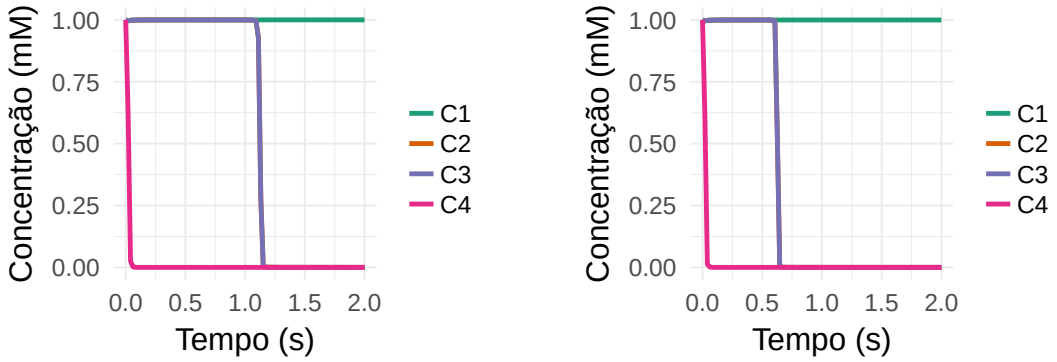
Com estas configurações, os comportamentos obtidos para a saída da última porta (`_AND4_`) do circuito são mostrados nas Figuras 5.9a e 5.9b. A configuração C2 não está visível na figura devido ao fato da mesma ter o mesmo comportamento de C3, fazendo com que o traçado de C3 ocultasse o de C2. Note que a única configuração a manter a saída do circuito ativa foi C1, como previsto já, que esta é a única configuração com o perfil de concentração que classifica uma célula como HeLa (cancerígena).

Perceba também que as configurações C2 e C3 demoram mais para convergirem para 0 do que C4. Isso porque apenas as duas primeiras entradas (que estão na primeira

[2] As reações formais e de DNA estão disponíveis em `https://goo.gl/FLweRS` e `https://goo.gl/8tZuqn`, respectivamente.

Tabela 5.3: Configurações em que o circuito em DNA baseado em neurônios químicos foi simulado. Cada configuração especifica um perfil diferente de concentrações dos marcadores.

Configuração	$[A]_0$	$[B]_0$	$[C]_0$	$[D]_0$	$[E]_0$
C1	$1,5\ mM$	$1,5\ mM$	$0,0\ mM$	$0,0\ mM$	$0,0\ mM$
C2	$0,0\ mM$	$1,5\ mM$	$0,0\ mM$	$0,0\ mM$	$0,0\ mM$
C3	$1,5\ mM$	$0,0\ mM$	$0,0\ mM$	$0,0\ mM$	$0,0\ mM$
C4	$0,0\ mM$	$0,0\ mM$	$1,5\ mM$	$1,5\ mM$	$1,5\ mM$



(a) Comportamento da CRN formal.

(b) Comportamento da CRN de DNA.

Figura 5.9: Resultados dos testes da CRN de DNA nas diferentes configurações definidas na Tabela 5.3.

porta do circuito) não estão dentro do perfil de concentração classificado como positivo. Por conta disso, apenas a primeira porta é gera uma saída 0, e esta alteração de sinal leva tempo até percorrer todas as portas do circuito.

Pode haver diferenças de comportamento entre a CRN formal e a de DNA já que são essencialmente CRNs distintas (em que uma tenta emular o comportamento da outra). Por conta disso, uma métrica interessante de ser avaliada é esta divergência entre a CRN formal e a de DNA (neste caso, gerada com base na abordagem de Soloveichik et al. [2010]) utilizando o cálculo de NRMSE (métido descrito na subseção 4.6). Assim, é possível quantificar o quão preciso a CRN de DNA emula o comportamento da CRN formal.

A Tabela 5.4 mostra o valor de NRMSE da espécie de saída do circuito para cada configuração. Os valores mostram que existe uma diferença significativa de comportamento das configurações C2 e C3. A causa desta diferença pode ser vista na Figura 5.9, em que há um maior atraso do decaimento da concentração de saída da CRN formal em comparação com a CRN de DNA. O motivo deste atraso ainda necessita de uma maior investigação.

Tabela 5.4: NRMSE calculado comparando o comportamento da CRN formal baseada nas portas lógicas de neurônios com a CRN de DNA gerada a partir da anterior.

	C1	C2	C3	C4
NRMSE	1.97E-04	2.46E-01	2.46E-01	1.03E-03

5.3.1 Comparação com outra abordagem

A abordagem tomada para construir o circuito lógico (5.18) foi utilizar um modelo de neurônio artificial químico como o elemento base. Conectando-os são formadas portas lógicas que operam em cascata. Existem outras formas de implementar portas lógicas as quais não é preciso utilizar uma modelagem intermediária como é feito na abordagem apresentada por meio de neurônios.

Uma das formas de se implementar portas lógicas químicas diretamente foi descrita na Seção 2.4. O trabalho de Jiang et al. [2013] define CRNs para representar as portas lógicas e os sinais que trafegam entre as mesmas. As portas possuem um comportamento bem semelhantes ao do neurônio, tendo característica de biestabilidade, além de como o estado da porta é representado. Cada porta possui duas espécies complementares (chamadas de Z_0 e Z_1) que representam o estado da mesma. Para conectar as portas, basta utilizar uma destas espécies de estado como entrada para outra porta.

Para definir se um valor de concentração representa um valor lógico 1, é utilizado um limiar C. Quando uma porta lógica expressa uma concentração de Z_1 (espécie de estado que define o valor de saída da porta) igual ou maior que C, a mesma possui valor lógico 1 e é considerado que ela está presente no sistema. No caso, define-se $C = 1\text{E}{-7}\ M$. Uma entrada é considerada com valor lógico 1 pelas portas lógicas caso a concentração das mesmas seja maior que $C/2$. Ou seja, dado uma porta `AND` com as entradas X e Y, caso $[X_1] > C/2$ e $[Y_1] > C/2$, $[Z_1]$ será igual a C. Caso contrário, $[Z_1] = 0$.

Relembrando, a construção de uma porta `AND` se dá por intermédio de 3 sinais (representados por duas espécies e 3 reações cada – veja as reações (2.26)), dois sinais de entrada X e Y, juntamente com um sinal de saída Z. Além das reações dos sinais, existem as reações que define o comportamento da porta, que são as reações (2.27) e (2.28). As concentrações iniciais das espécies são as seguintes: $[Z_1]_0 = [Z_1']_0 = 0$, $[S_X]_0 = [S_Y]_0 = [S_Z]_0 = 0$, e por fim, $[Z_0]_0 = C$ (o que faz com que $[Z_1]_0 = 0$).

De acordo com os autores, esta abordagem tem como objetivo ser uma construção robusta de portas lógicas. Por isso, não é necessário ter uma valor específico de

constante de velocidade para cada reação da CRN. Basta apenas um valor grosseiro de constante para todas as reações para garantir que a computação ocorra corretamente [Jiang et al., 2013]. Neste trabalho, foi adotado a constante $k = 1E4\ M^{-1}s^{-1}$.

Partindo para a construção da CRN (5.18), a qual classifica células HeLa, é necessária a negação (`NOT`) de uma das entradas de 3 das 4 portas `AND`. Da mesma forma como é feito na modelagem baseada em neurônio, já que todo sinal é representado por duas espécies (X_1 e seu complemento X_0), basta utilizar como entrada para a outra porta o complemento do sinal (X_0 no caso do sinal X).

A CRN formal que implementa o circuito lógico (5.18) foi definida com base na abordagem de construção de portas lógicas de Jiang et al. [2013], utilizando o pacote `DNArDigitalLogic` [3] implementado por Renan Marks (atual Doutorando em Ciência da Computação e membro do Laboratório de NanoComputação da Universidade Federal de Minas Gerais). O Algoritmo 5.5 mostra como definir uma CRN que implementa o circuito lógico.

A primeira seção do Algoritmo define as portas lógicas. O primeiro parâmetro determina o nome da porta. O segundo e terceiro parâmetros atribuem os valores de concentrações inicias às entradas das portas. O último parâmetro que aparece em 3 das 4 portas (`input2negate`) faz com que a segunda entrada seja negada, ou seja, a espécie que representa o complemento do valor do sinal é conectado como entrada da porta (e.g.: supondo X como o sinal de entrada, X_0 passar ser a espécie usada como entrada, ao invés de X_1). É possível negar a primeira entrada usando o parâmetro `input1negate`.

A segunda seção cria o circuito baseado nas portas lógicas definidas. O circuito é definido para ser simulado por 25000 segundos (6,95 horas, aproximadamente). E por fim, as portas lógicas do circuito são conectadas nas últimas linhas do Algoritmo.

A estrutura `circ` contém todos os parâmetros que definem a CRN, bastando passá-las para função `react()` (ou `react_4domain()`) para iniciar o processo de simulação.

Esta CRN formal foi então traduzida em uma CRN de DNA seguindo o processo descrito na Seção 3.3. Os parâmetros α, β, q_{max} e c_{max} foram definidos também utilizando a função `update_crn_4domain()`. Os perfis de concentrações iniciais das entradas simulados foram os mesmos da Tabela 5.3, apenas alterando o valor de um sinal ativo de $1,5\ mM$ para $1E{-}7\ M$, com o intuito de reescalar as entradas para a nova CRN. Com isso, tem-se que o comportamento obtido é mostrado na Figura 5.10.

Como é possível observar, apenas a configuração C1 gerou uma saída com valor

[3] Disponível em `https://gitlab.com/DanielKneipp/DNArDigitalLogic`

```
g1 <- make_and_gate('AND1', input1value, input2value)
g2 <- make_and_gate('AND2', 0, input3value, input2negate=T)
g3 <- make_and_gate('AND3', 0, input4value, input2negate=T)
g4 <- make_and_gate('AND4', 0, input5value, input2negate=T)

circ <- make_circuit(seq(0, 25000, 100))
circ <- circuit_add_gate(circ, g1)
circ <- circuit_add_gate(circ, g2)
circ <- circuit_add_gate(circ, g3)
circ <- circuit_add_gate(circ, g4)

circ <- circuit_link_gate_signals(
    circ,
    g1$species$output,
    g2$species$input1
)
circ <- circuit_link_gate_signals(
    circ,
    g2$species$output,
    g3$species$input1
)
circ <- circuit_link_gate_signals(
    circ,
    g3$species$output,
    g4$species$input1
)
```

Algoritmo 5.5: Definindo uma CRN baseada em portas lógicas químicas para a implementação do circuito lógico (5.18).

lógico 1 do circuito. Uma diferença clara do comportamento gerado por esta CRN em comparação com a CRN baseada em neurônios é que o valor lógico das portas desta CRN começam em 0, diferentemente dos neurônios, que possuem o estado inicial da saída no valor lógico 1.

Como esta abordagem faz uma implementação direta de portas lógicas em reações químicas, espera-se que esta possua um número menor de reações que a CRN baseada em neurônio. Isto de fato é confirmado. A CRN formal de portas lógicas possui 62 reações, enquanto a transformação para CRN de DNA gerou 272 [4], pouco mais da metade do número de reações que a CRN baseada em neurônios. No entanto, lembre-se que lógica booleana é apenas uma das possibilidades que a modelagem das redes neurais artificiais oferece.

[4] As reações formais e de DNA estão disponíveis em `https://goo.gl/5NHez3` e `https://goo.gl/1AepDb`, respectivamente.

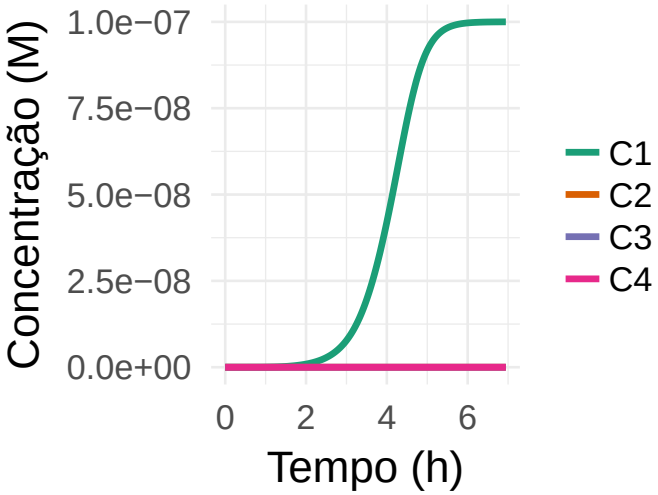


Figura 5.10: Comportamento da CRN de DNA obtida a partir da abordagem de implementação de portas lógicas químicas.

A Tabela 5.5 mostra que a divergência entre o comportamento da CRN formal e da CRN de DNA é desprezível. Isso é um ponto importante para a abordagem de construção de portas lógicas de Jiang et al. [2013], tendo em vista que com uma divergência menor, há maiores chances de que a CRN de DNA emule corretamente o comportamento da CRN formal *in-vivo*.

Tabela 5.5: NRMSE calculado comparando o comportamento da CRN formal baseada nas portas lógicas de Jiang et al. [2013] com a CRN de DNA gerada a partir da anterior.

	C1	C2	C3	C4
NRMSE	4.88E-11	1.05E-26	0	0

Algo que pode chamar atenção é a diferença no tempo de simulação desta CRN (mais de 6 horas) com a CRN baseada em neurônios (2 segundos). Isto não deve ser levado em conta, tendo em vista que as concentrações das espécies bem como as constantes de velocidade podem ser reescaladas para que o comportamento se ajuste em um determinado intervalo de tempo.

Neste capítulo foram discutidas duas CRNs capazes de diagnosticar células cancerígenas do tipo HeLa. Cada uma baseada em uma abordagem diferente para construção de circuitos lógicos utilizando reações químicas. Uma baseada no modelo de neurônio de McCulloch–Pitts [Hjelmfelt et al., 1991], enquanto a outra descreve uma forma direta de implementar e cascatear portas lógicas [Jiang et al., 2013].

O pacote DNAr foi utilizado para gerar as CRNs baseadas em DNA e simular as

mesmas. Por conta do pacote operar por meio de algoritmos que acessam interfaces públicas, *softwares* (e.g.: outros pacotes) podem ser desenvolvidos para definir CNRs programaticamente, o que caracteriza um dos pontos positivos mais fortes de se usar o DNAr.

Um conjunto de funções foi criado para cada abordagem tratada. Algumas destas funções são usadas nos Algoritmos 5.4 e 5.5, os quais mostram que com apenas algumas linhas de código é possível definir CRNs compostas por centenas de reações, além de simula-las no DNAr de forma fácil e ágil. Isto mostra que o DNAr foi feito para ser expandido, de forma que o mesmo trabalhe com diferentes conjuntos de funções que definem CRNs baseadas nas mais variadas abordagens de construção de circuitos e arquiteturas químicas.

Capítulo 6

Conclusões e Trabalhos Futuros

Com o transistor baseado em silício chegando ao limite de sua miniaturização, novas tecnologias vem sendo pesquisadas para substituir a atual arquitetura em determinados casos. Uma das alternativas atualmente pesquisadas é a computação com DNA (*deoxyribonucleic acid* – ácido desoxirribonucleico), um dos campos da computação molecular.

DNA vem sendo usado como matéria prima nas mais variadas aplicações, entre elas a robótica, armazenamento de dados e computação em si. A computação com DNA não é feita para substituir arquiteturas baseadas em silício em sua totalidade. Reações entre moléculas de DNA podem demorar horas, muito superior à frequência de *clock* dos processadores convencionais da atualidade (que podem operar a 2Ghz, ou seja, 2 bilhões de ciclos por segundo).

Os pontos positivos da computação com DNA são explorados quando:

- a solução de um problema pode ser obtida por meio de processamento massivamente paralelo, já que é possível ter bilhões de reações entre moléculas de DNA simultâneas ocorrendo em um tubo de ensaio;

- quando se faz necessário criar dispositivos que operam *in-vivo* (funcionam em células ou organismos vivos), uma vez que dispositivos baseado em DNA reagem de forma previsível com componentes que existem nestes contextos (geralmente outras moléculas de DNA ou RNA).

Uma arquitetura de computação baseada em DNA (como qualquer outra) necessita de uma linguagem capaz de descrever seus dispositivos. A linguagem utilizada para expressar redes de reações químicas (CRNs) é empregada para suprir esta demanda. Esta linguagem já vem sendo estudada de forma independente há vários anos, con-

templando um arcabouço teórico com inúmeros dispositivos e mecanismos teorizados [Zechner & Khammash, 2016; Napp & Adams, 2013]. Conseguir utilizar a linguagem das CRNs para especificar reações entre moléculas de DNA significa ter a possibilidade de implementar na prática utilizando DNA vários destes dispositivos que antes eram apenas teóricos.

Além de dispositivos e mecanismos, existem abordagens da construção de arquiteturas complexas de uso genérico teorizadas por meio da linguagem das CRNs. Alguns exemplos discutem como construir redes neurais artificiais [Hjelmfelt et al., 1991] e portas lógicas [Jiang et al., 2013]. Isto adiciona camadas de abstração sobre o desenvolvimento de dispositivos baseados em DNA, facilitando e acelerando a criação dos mesmos.

Uma das formas de simular comportamento de mecanismos especificados por meio da linguagem das CRNs é resolver numericamente um sistema de equações diferencias. Estas EDOs representam o comportamento do dispositivo, ou seja, elas expressam a variação da concentração de cada espécie no tempo. Tradicionalmente, para realizar esta simulação, estas EDOs devem ser especificadas como um modelo, e um método de solução numérica deve ser empregado para resolver este modelo.

No entanto, com a adição destas camadas de abstração, as redes de reações se tornam cada vez maiores e mais complexas. Com isso, simula-las de forma tradicional se torna inviável, ou dependendo do caso, até mesmo especifica-las textualmente.

Nesta dissertação, um pacote de *software* chamado de DNAr foi desenvolvido com o intuito de simular mecanismos de larga escala especificados por meio da linguagem das CRNs. O usuário pode acessar suas funcionalidades através de interfaces públicas as quais podem ser usadas por *scripts* ou outros pacotes R. R foi a linguagem de programação escolhida por ser de código aberto, não gerar custo financeiro ao usuário, já possuir métodos numéricos eficientes implementados em pacotes externos e já ser utilizada amplamente em pesquisas de cunho científico.

Além de simular redes de reações teóricas (chamadas de CRNs formais), o DNAr é capaz de gerar redes de reações entre moléculas de DNA (referenciadas como CRNs de DNA) que reproduzem o comportamento desejado das reações formais. Isto é feito de forma automática utilizando a abordagem de Soloveichik et al. [2010] como esquema de tradução, sendo necessário apenas definir alguns parâmetros específicos da abordagem.

Esta CRN de DNA pode ser simulada pelo próprio DNAr, ou exportada para outro simulador chamado de Visual DSD, da Microsoft Corp. Este é feito para simular de forma mais detalhada a interação entre moléculas de DNA, possuindo vários parâmetros de configuração e níveis de detalhamento para a simulação. O Visual DSD também permite alterar o método de simulação, entre as opções existem a simulação

determinística (baseada em sistema de EDOs, o mesmo método usado pelo DNAr), e a estocástica (modelagem probabilística baseada em cadeias de Markov). Nele é possível inclusive verificar os domínios dos fragmentos de DNA que estão interagindo. É importante salientar que este recurso (exportar CRNs de DNA para o Visual DSD) não funciona para qualquer CRN, sendo ainda um trabalho futuro expandir o escopo de funcionamento deste recurso no DNAr.

Para demonstrar o uso do DNAr, o desenvolvimento de um detector de células cancerígenas foi usado como estudo de caso. O trabalho de Xie et al. [2011] descreve um circuito lógico capaz de classificar se uma célula é do tipo HeLa ou não (células que participam desta linhagem são classificadas como cancerígenas). Este circuito utiliza marcadores biológicos para fazer a classificação. Caso a célula possua um perfil de concentração específico para estes marcadores, a classificação deve retornar positivo, caso contrário, negativo.

Para construir uma CRN que tenha o comportamento deste circuito lógico, duas abordagens foram usadas: uma modelagem de neurônio químico inspirado no modelo de McCulloch–Pitts [Hjelmfelt et al., 1991], e outra que descreve como implementar portas lógicas diretamente em CRNs [Jiang et al., 2013]. Cada abordagem gera uma CRN formal, a qual é usada para obter a CRN de DNA. O comportamento das CRNs de DNA das duas abordagens foram comparados com os comportamentos desejados, ditados pelas CRNs formais. Os resultados mostraram que as CRNs de DNA das duas abordagens alcançaram o comportamento desejado com um erro desprezível. Apesar disso, a abordagem de Jiang et al. [2013] (implementação direta de portas lógicas) possui um erro menor, ao mesmo tempo que tem um número consideravelmente menor de reações (próximo da metade), comparando com a abordagem de neurônios químicos.

As duas abordagens geraram CRNs com mais de 200 reações, o que inviabiliza a simulação das mesmas de forma tradicional, ou até mesmo descreve-las de forma textual. Com o uso de pacotes de funções implementados para operarem em conjunto com o DNAr, simular CRNs de larga escala como estas se torna possível com apenas algumas linhas de código. Isto demonstra não somente a praticidade de se simular sistemas químicos no DNAr, mas também sua permissibilidade em ser expandido. Como foi mostrado, é possível implementar conjuntos de funções (como pacote externo ou integrados ao DNAr) usadas para definir mecanismos químicos, ou CRNs baseadas nas mais variadas abordagens.

6.1 Trabalhos Futuros

Como já foi deixado claro anteriormente, o DNAr foi feito para ser expandido por meio de funções e pacotes externos. Ou seja, esta dissertação deu apenas o primeiro passo em direção a um conjunto de ferramentas feitas para facilitar e automatizar o processo de simulação de redes de reações químicas e entre moléculas de DNA. A seguir são discutidas algumas oportunidades para trabalhos futuros que contribuiriam em direção da criação de um conjunto de ferramentas robusto para este fim:

- Remover a normalização das entradas feitas nas portas lógicas feitas com neurônios químicos. Quando a concentração máxima das entradas que cada porta lógica pode receber é limitada, a normalização que é feita pelos neurônios E_1 e E_2 (mostrados na Figura 5.5) não é necessária. A remoção dos mesmos acarreta em uma redução considerável no número de reações, e pode até mesmo diminuir a diferença do comportamento da CRN de DNA se comparado com a CRN formal;

- Estender o suporte ao Visual DSD. Como já foi dito, o Visual DSD é uma ferramenta capaz de simular reações entre moléculas de DSD com várias configurações de detalhamento e métodos de simulação. O DNAr consegue gerar códigos compatíveis com o DSD para CRNs mais simples, mas para CRNs mais complexas os comportamentos simulados nas duas ferramentas podem divergir. Investigar o motivo destas discordâncias aumentaria a interoperabilidade entre elas;

- O único método de simulação implementado no DNAr é o determinístico baseado em sistemas de equações diferencias. Este é um modelo de simulação mais superficial. Uma contribuição interessante seria implementar métodos mais complexos, como métodos estocásticos baseados em cadeias de Markov. Isto possibilitaria o DNAr simular a cinética de espécies com dezenas ou centenas de moléculas, permitindo avaliar o comportamento do sistema em termos da quantidade de moléculas (indivíduos) das espécies no tempo, e não apenas a nível de concentrações;

- Implementar outras abordagens de construção de CRNs e mecanismos químicos. Existe uma vasta gama de sistemas químicos teorizados. Implementar alguns deles (da mesma forma que foi feito com os neurônios químicos e portas lógicas) facilita o uso dos mesmos por outros pesquisadores e aumenta a gama de extensões suportadas pelo DNAr. Alguns exemplos de mecanismos interessantes de serem implementados são as CRNs que implementam o algoritmo de propagação de

crenças [Napp & Adams, 2013], e o estimador de mínimos quadrados [Zechner & Khammash, 2016].

Referências Bibliográficas

Adams, R. L. P.; Knowler, J. T. & Leader, D. P. (1986). *The biochemistry of the nucleic acids*. Chapman and Hall, 10 edição.

Adleman, L. M. (1994). Molecular computation of solutions to combinatorial problems. *Nature*, 369:40.

Amos, M.; Păun, G.; Rozenberg, G. & Salomaa, A. (2002). Topics in the theory of dna computing. *Theoretical computer science*, 287(1):3--38.

Anderson, D. F. (2014). Tutorial: chemical reaction network theory for both deterministic and stochastic models. *Programming with Chemical Reaction Networks: Mathematical Foundations*.

Arkin, A. & Ross, J. (1994). Computational functions in biochemical reaction networks. *Biophysical journal*, 67(2):560--578.

Arnon, S.; Dahan, N.; Koren, A.; Radiano, O.; Ronen, M.; Yannay, T.; Giron, J.; Ben-Ami, L.; Amir, Y.; Hel-Or, Y. et al. (2016). Thought-controlled nanoscale robots in a living host. *PloS one*, 11(8):e0161227.

Beeman, D. (2001). Some specific models of artificial neural nets. `http://ecee.colorado.edu/~ecen4831/lectures/NNet2.html`. Acessado em 21 jan. 2018.

Benenson, Y. (2012). Biomolecular computing systems: principles, progress and potential. *Nature Reviews Genetics*, 13(7):455--468.

Benenson, Y.; Gil, B.; Ben-Dor, U.; Adar, R. & Shapiro, E. (2004). An autonomous molecular computer for logical control of gene expression. *Nature*, 429(6990):423--429.

Brown; LeMay; Busten; murphy & Woodward (2017a). Avogadro's number and the mole. `https://goo.gl/C6Vop6`. Acessado em 11 nov. 2017.

Brown; LeMay; Busten; murphy & Woodward (2017b). Concentration and rates (differential rate laws). `https://goo.gl/brWp3G`. Acessado em 14 nov. 2017.

Brown; LeMay; Busten; murphy & Woodward (2017c). Reactions in aqueous solution. `https://goo.gl/epxQdn`. Acessado em 11 nov. 2017.

Brown, P. N.; Byrne, G. D. & Hindmarsh, A. C. (1989). Vode: A variable-coefficient ode solver. *SIAM journal on scientific and statistical computing*, 10(5):1038--1051.

Butcher, J. C. (1987). *The numerical analysis of ordinary differential equations: Runge-Kutta and general linear methods*. John Wiley & Sons. ISBN 9780471910466.

Butcher, J. C. (2001). Numerical methods for ordinary differential equations in the 20th century. Em *Numerical analysis: Historical developments in the 20th century*, pp. 449--477. Elsevier.

Carlson, R. (2009). The changing economics of dna synthesis. *Nature biotechnology*, 27(12):1091.

Cavin, R. K.; Lugli, P. & Zhirnov, V. V. (2012). Science and engineering beyond moore's law. *Proceedings of the IEEE*, 100(Special Centennial Issue):1720--1749.

Chellaboina, V.; Bhat, S.; Haddad, W. & Bernstein, D. (2009). Modeling and analysis of mass-action kinetics. *IEEE Control Systems Magazine*, 29(4):60--78.

Chemistry LibreTexts (2015). The rate law. `https://goo.gl/3hBPnP`. Acessado em 13 nov. 2017.

Chemistry LibreTexts (2017). The equilibrium constant. `https://goo.gl/8i8dro`. Acessado em 28 jan. 2018.

Chen, Y.-J.; Dalchau, N.; Srinivas, N.; Phillips, A.; Cardelli, L.; Soloveichik, D. & Seelig, G. (2013). Programmable chemical controllers made from dna. *Nature nanotechnology*, 8(10):755--762.

Chen, Y.-J.; Groves, B.; Muscat, R. A. & Seelig, G. (2015). Dna nanotechnology from the test tube to the cell. *Nature nanotechnology*, 10(9):748--760.

Dixon, M.; Klabjan, D. & Bang, J. H. (2016). Classification-based financial markets prediction using deep neural networks. *Algorithmic Finance*, (Preprint):1--11.

Dolnik, M.; Banks, A. S. & Epstein, I. R. (1997). Oscillatory chemical reaction in a cstr with feedback control of flow rate. *The Journal of Physical Chemistry A*, 101(28):5148--5154.

Douglas, S. M.; Bachelet, I. & Church, G. M. (2012). A logic-gated nanorobot for targeted transport of molecular payloads. *Science*, 335(6070):831--834.

Eddelbuettel, D.; François, R.; Allaire, J.; Ushey, K.; Kou, Q.; Russel, N.; Chambers, J. & Bates, D. (2011). Rcpp: Seamless r and c++ integration. *Journal of Statistical Software*, 40(8):1--18.

Ewing, M. B.; Lilley, T. H.; Olofsson, G. M.; Ratzsch, M. T. & Somsen, G. (1994). Standard quantities in chemical thermodynamics. fugacities, activities and equilibrium constants for pure and mixed phases (IUPAC recommendations 1994). *Pure and Applied Chemistry*, 66(3):533--552.

Fairhead, H. (2014). The mcculloch-pitts neuron. `http://www.i-programmer.info/babbages-bag/325-mcculloch-pitts-neural-networks.html`. Acessado em 21 jan. 2018.

Feinberg, M. (1979). Lectures on chemical reaction networks. *Notes of lectures given at the Mathematics Research Center, University of Wisconsin*, p. 49.

Hairer, E.; Norsett, S. & Wanner, G. (1993). *Solving Ordinary Differential Equations I*, volume 8 of *Springer Series in Computational Mathematics*. Springer Berlin Heidelberg. ISBN 978-3-540-78862-1.

Hairer, E. & Wanner, G. (1991). *Solving Ordinary Differential Equations II*, volume 14 of *Springer Series in Computational Mathematics*. Springer Berlin Heidelberg. ISBN 978-3-662-09947-6.

Han, S.-J. & Cho, S.-B. (2005). Evolutionary neural networks for anomaly detection based on the behavior of a program. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 36(3):559--570.

Hastings, P. (2001). Branch migration. Em *Encyclopedia of Genetics*, p. 237. Elsevier.

Hindmarsh, A. C. (1983). Odepack, a systematized collection of ode solvers, rs stepleman et al.(eds.), north-holland, amsterdam,(vol. 1 of), pp. 55-64. *IMACS transactions on scientific computation*, 1:55--64.

Hjelmfelt, A.; Weinberger, E. D. & Ross, J. (1991). Chemical implementation of neural networks and turing machines. *Proceedings of the National Academy of Sciences*, 88(24):10983--10987.

Hoppensteadt, F. C. & Izhikevich, E. M. (2000). Pattern recognition via synchronization in phase-locked loop neural networks. *IEEE Transactions on Neural Networks*, 11(3):734--738.

Jagreet (2017). Overview of artificial neural networks and its applications. `https://goo.gl/1P17mE`. Acessado em 19 jan. 2018.

Jiang, H.; Riedel, M. D. & Parhi, K. K. (2013). Digital logic with molecular reactions. Em *Computer-Aided Design (ICCAD), 2013 IEEE/ACM International Conference on*, pp. 721--727. IEEE.

Kim, K.-j. (2006). Artificial neural networks with evolutionary instance selection for financial forecasting. *Expert Systems with Applications*, 30(3):519--526.

Knight, D. (2015). Universal logic gates. `https://www.allaboutcircuits.com/technical-articles/universal-logic-gates/`. Acessado em 21 jan. 2018.

Kotz, J. C. & Treichel, P. M. (2017). Chemical reaction. `https://www.britannica.com/science/chemical-reaction`. Acessado em 08 nov. 2017.

Krizhevsky, A.; Sutskever, I. & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. Em *Advances in neural information processing systems*, pp. 1097--1105.

Laidler, K. J. (2017). Chemical kinetics. `https://www.britannica.com/science/chemical-kinetics`. Acessado em 12 nov. 2017.

Lakin, M. R.; Youssef, S.; Cardelli, L. & Phillips, A. (2012). Abstractions for dna circuit design. *Journal of The Royal Society Interface*, 9(68):470--486.

Lakin, M. R.; Youssef, S.; Polo, F.; Emmott, S. & Phillips, A. (2011). Visual dsd: a design and analysis tool for dna strand displacement systems. *Bioinformatics*, 27(22):3211--3213.

Lucey, B. P.; Nelson-Rees, W. A. & Hutchins, G. M. (2009). Henrietta lacks, hela cells, and cell culture contamination. *Archives of pathology & laboratory medicine*, 133(9):1463--1467.

Marquardt, R.; Meija, J.; Mester, Z.; Towns, M.; Weir, R.; Davis, R. & Stohner, J. (2018). Definition of the mole (IUPAC recommendation 2017). *Pure and Applied Chemistry*, 90(1):175--180.

Marsalli, M. (2006). Mcculloch-pitts neurons. Em *The 2008 Annual Meeting of the consortium on cognitive science instruction (ccsi)*, pp. 172--179.

Mayer, C.; McInroy, G. R.; Murat, P.; Van Delft, P. & Balasubramanian, S. (2016). An epigenetics-inspired dna-based data storage system. *Angewandte Chemie International Edition*, 55(37):11144--11148.

McCulloch, W. S. & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115--133.

MeSH (2017). Nucleic acid hybridization. `https://www.ncbi.nlm.nih.gov/mesh?term=Nucleic+Acid+Hybridization`. Acessado em 1 dez. 2017.

Mills, I.; Cvitas, T.; Homann, K. & IUPAC (1993). *Quantities, Units and Symbols in Physical Chemistry (IUPAC Chemical Data)*. Wiley-Blackwell. ISBN 0-632-03583-8.

Mohr, P. J.; Newell, D. B. & Taylor, B. N. (2016). CODATA recommended values of the fundamental physical constants: 2014. *Reviews of Modern Physics*, 88(3).

Napp, N. E. & Adams, R. P. (2013). Message passing inference with chemical reaction networks. Em *Advances in neural information processing systems*, pp. 2247--2255.

National Institutes of Health (2016). Talking glossary of genetic terms. `https://www.genome.gov/glossary/`. Acessado em 08 nov. 2016.

Nielsen, M. A. (2015). Neural networks and deep learning. `http://neuralnetworksanddeeplearning.com/`. Acessado em 21 jan. 2018.

Noyes, R. M. (1989). Some models of chemical oscillators. *Journal of Chemical Education*, 66(3):190.

Petzold, L. (1983). Automatic selection of methods for solving stiff and nonstiff systems of ordinary differential equations. *SIAM journal on scientific and statistical computing*, 4(1):136--148.

Phillips, A. & Cardelli, L. (2009). A programming language for composable dna circuits. *Journal of the Royal Society Interface*, 6(Suppl 4):S419--S436.

Poshusta, R. (2016). Oscillating chemical reactions. `http://www.idea.wsu.edu/OscilChem/#Lotka-VolterraModel`. Acessado em 17 nov. 2017.

Qian, L. & Winfree, E. (2011a). Scaling up digital circuit computation with dna strand displacement cascades. *Science*, 332(6034):1196--1201.

Qian, L. & Winfree, E. (2011b). A simple dna gate motif for synthesizing large-scale circuits. *Journal of the Royal Society Interface*, p. rsif20100729.

Qian, L.; Winfree, E. & Bruck, J. (2011). Neural network computation with dna strand displacement cascades. *Nature*, 475(7356):368--372.

R Core Team (2017). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

Ross, N. (2014). Vectorization in r: Why? `http://www.noamross.net/blog/2014/4/16/vectorization-in-r--why.html`. Acessado em 29 dez. 2017.

Senum, P. & Riedel, M. (2011). Rate-independent constructs for chemical computation. *PloS one*, 6(6):e21414.

Soetaert, K.; Petzoldt, T. & Setzer, R. W. (2010). Solving differential equations in r: Package desolve. *Journal of Statistical Software*, 33(9):1--25. ISSN 1548-7660.

Soloveichik, D.; Seelig, G. & Winfree, E. (2010). Dna as a universal substrate for chemical kinetics. *Proceedings of the National Academy of Sciences*, 107(12):5393--5398.

Sterner, R. W.; Small, G. E. & Hood, J. M. (2011). The conservation of mass. `https://goo.gl/Ps5o92`. Acessado em 3 mai. 2018.

Vance, A. (2009). Data analysts captivated by r's power. *New York Times*, 6(5.4).

Watson, J. D.; Crick, F. H. et al. (1953). Molecular structure of nucleic acids. *Nature*, 171(4356):737--738.

Xie, Z.; Wroblewska, L.; Prochazka, L.; Weiss, R. & Benenson, Y. (2011). Multi-input rnai-based logic circuit for identification of specific cancer cells. *Science*, 333(6047):1307--1311.

Zechner, C. & Khammash, M. (2016). A molecular implementation of the least mean squares estimator. Em *Decision and Control (CDC), 2016 IEEE 55th Conference on*, pp. 5869--5874. IEEE.

Zhang, D. Y. & Seelig, G. (2011). Dynamic dna nanotechnology using strand-displacement reactions. *Nature chemistry*, 3(2):103--113.

Zhang, G.; Patuwo, B. E. & Hu, M. Y. (1998). Forecasting with artificial neural networks:: The state of the art. *International journal of forecasting*, 14(1):35--62.

Zhang, G. P. (2000). Neural networks for classification: a survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 30(4):451--462.