Poznan Monographs in Computing and Its Applications

The *Poznan Monographs in Computing and Its Applications* series publishes the very latest, high-quality research results in computing and its practical applications. It includes primarily doctoral and postdoctoral (habilitation) dissertations written by scientists from Poland, from the Poznan scientific environment. The most valuable dissertations are selected from the range of monographs published by scientists from the Faculty of Computing and the Faculty of Electronics and Telecommunications at the Poznan University of Technology (PUT) and from the Poznan Supercomputing and Networking Center (PSNC). The series goal is to promote interesting research results and spread them to broader audience. Before the series was established such monographs were shared with very limited groups of people and it was almost impossible for them to reach interested readers. Each position in the series is carefully reviewed by experienced professors and then corrected according to their remarks.

POLITECHNIKA POZNAŃSKA

Bioinformatyczne modele i algorytmy infekcji wirusowych

Bioinformatics models and algorithms of viral infections

ROZPRAWA DOKTORSKA

mgr inż. Szymon Wąsik

Promotor

prof. dr hab. inż. Jacek Błażewicz

Promotor pomocniczy

dr inż. Piotr Łukasiak

Instytut Informatyki

Wydział Informatyki

2012

Cover design: Agata Łożykowska
Scientific review: Jacek Błażewicz

Edition 1

Volume 10
ISBN: 978-83-89529-98-5

Publisher: Wydawnictwo NAKOM
ul. Wielka 10, 61-774 Poznań, Poland
tel./fax +48 61 852 83 82, +48 61 852 97 47
e-mail: info@nakom.com.pl, wydawnictwo@nakom.com.pl
www.nakom.com.pl

Print: TOTEM
www.totem.com.pl
tel. +48 52 35 400 40

Człowiek stworzony jest na to,
by szukać prawdy, a nie by ją posiadać.

Blaise Pascal

Mało wiedzy oddala od Boga.
Dużo wiedzy sprowadza do Niego z powrotem.

Ludwig Pasteur

Podziękowania

Serdecznie dziękuję

prof. dr. hab. inż. Jackowi Błażewiczowi
za powierzenie ciekawej tematyki badawczej,
wsparcie oraz nieocenione wskazówki w trakcie prowadzenia badań

prof. dr. hab. Markowi Figlerowiczowi
za pomoc dotyczącą biologicznych aspektów pracy
oraz nieocenione wskazówki w trakcie prowadzenia badań

żonie Agacie, córkom Julii i Łucji oraz pozostałym najbliższym,
za wsparcie i cierpliwość

Oświadczam, że jestem stypendystą w ramach projektu pt.: „Wsparcie stypendialne dla doktorantów na kierunkach uznanych za strategiczne z punktu widzenia rozwoju Wielkopolski”, Poddziałanie 8.2.2 Programu Operacyjnego Kapitał Ludzki, współfinansowanego ze środków Unii Europejskiej w ramach Europejskiego Funduszu Społecznego.

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Spis treści

Spis rysunków

Spis tablic

Spis algorytmów

Spis listingów

Abstract

Viral infections are one of the most dangerous sources of diseases affecting humans and animals. For example, Hepatitis C virus (HCV) infections are among major global health problems and concern 3% of the human population [Wor00]. Most of these infections become chronic and lead to liver failure, including brosis, cirrhosis and hepatocellular carcinoma [CM06]. Another common virus is HIV (Human immunodeficiency virus) which infects approximately 34 million people worldwide and kills over 1 million each year [oH11].

Mathematical models of viral infections have been valuable tools in addressing biologically important questions concerned with crucial features of viral dynamics. This thesis describes in details consecutive steps that have to be completed to prepare the valuable model of such an infection. This is achieved by fulfilling following goals:

1. Reviewing already existing modelling techniques and analysing the classical model of viral infection (Chapter 4).
2. Designing an intuitive, easily understandable by biologists language for describing mathematical models of viral infections (Chapter 5).
3. Designing simulation framework and algorithms for analysing viral infections (Chapter 6).
4. Designing better statistical methods for analysing therapy efficiency in the human population (Chapter 7).

The following is a short summary of chapters describing results corresponding to each of the goals listed above.

Existing models of viral infections

There are several groups of methods dedicated to modelling biological systems. Probably the most popular group of methods use systems of differential equations [Per89, WN02, DGPL11, SH08]. The application of these methods requires both advanced knowledge and experience with mathematical analysis. The example system of differential equations is presented in equations 4.1-4.3.

The second group of modelling methods involves designing dedicated formal languages. This group of tools includes for example XMLlab [MP04] and SBML [HFS$^+$03] languages based on XML, JiST based on Java [BHvR05], SimPy based on Python [MV03], Antimony [SBCS09] or the immune system simulation frameworks MSI[MLF$^+$08] and CAFISS [TJ05]. Also rule-based approaches [Fae11, MRU11] that have recently been quite popular fall into this group. All of them require the user to know a programming language. An example SBML model is presented in appendix A.1, an example Antimony model is presented in listing 4.2 and an example set of rules is presented in equations 4.4-4.12.

The third group of methods consists of higher level software that have some graphical user interface (GUI) that can be used for easy visual construction of biological system models . Unfortunately all the existing packages either have very limited functionality or require preparation of several scripts written in formal programming language to supplement the graphically designed part. Example tools in this group are Brahms [SCvH03], AndroMeta [And12], NetLogo [Wil99] or SBGN [LHM$^+$09]. The example SBGN model is presented in figure 4.2 and its formal description in section A.2.

The last group of methods is composed of programs that help to construct models from blocks using graphical user interface, but in this case the researcher has to have perfect understanding of how to formally model biological interactions (for example using differential equations). These are programs like Mathworks Simulink [KA11] and WinFACT [Ber12]. Example Simulink models are presented in figures 4.1 and A.1.

All modelling methods described above usually consists of some software used to prepare a description of the model using a computer and some simulation framework that can be used to analyse them. Most of these methods were extensively used to model viral infections. However, even using these very well known and commonly applied tools and algorithms, some important analyses

can be omitted. For example, usually during analysis of an HCV infection, the available clinical data are limited to the level of HCV RNA in serum. Acquisition of more detailed data, such as the number of uninfected and infected hepatocytes, requires invasive procedure of liver biopsy, followed by sophisticated analyses. This probably explains why authors of mathematical models of HCV kinetics usually pay attention only to the equation that describes the level of the viral RNA in serum and use this equation to fit their model results to data. This approach can give rise to misleading conclusions, because one cannot state that a model's solution correctly fits to data unless equations modelling the number of hepatocytes give at least reasonable results. In most of the models it is assumed that the number of hepatocytes is constant during first days of therapy and this fact is not verified in simulation results. However, solution of the equation that models the number of virions can fit perfectly, but unless the solution of equations that model the number of hepatocytes fits data, the model does not have to explain data correctly. Such a case is described in section 4.1.1 and using agents based model is investigated. More details about this analysis can be found in [WJFB11, WPB12].

Formal description of biological models

Computational modelling in biology connects two different groups of disciplines – computational mathematics and computer science with biology, with a mutual benefit. Mathematicians and computer scientists help to process large amount of data and to find regularities in them, and in turn, biology provides challenging problems that are not found in other disciplines [dVHL$^+$06]. However, at the point of contact of such different disciplines, communication problems arise. Biologists often have insufficient knowledge of mathematics and computer science to describe biological systems in a formal way and when mathematicians and computer scientists do this on their behalf, biologists have difficulties to understand and verify such descriptions. Moreover, mathematicians themselves have a problem with gathering coherent and precise biological knowledge required to define models of biological systems.

None of the methods presented in the previous section provides an intuitive modelling tool for a person without high mathematical or programming skills. Time consuming and expensive cooperation with a mathematician or computer scientist is usually required to apply them. To solve this problem the ModeLang language was designed (modelang.cs.put.poznan.pl). It is a new formal language that can be used to describe viral infection models. It can be a useful tool

facilitating cooperation between researchers from different disciplines working on models of viral infections. To make its use easy, it was designed to be as similar as possible to a natural language [CH04]. Finally, thanks to the open and free architecture it can be used as an input for many different modelling techniques.

ModeLang is an innovative, expert-friendly language. The results of its tests clearly show that a description written in ModeLang is much easier to understand, modify or even create from scratch for biologists than for example a definition of model based on differential equations. As a result the use of ModeLang can significantly speed up computational verification of formulated biological hypotheses. This will make the process of designing new therapies and medicines much easier, faster and cheaper, and as a result it can save many lives and help to improve health of many patients.

To make sure that ModeLang is biologist-friendly during the research phase it was consulted with biologists from the Institute of Bioorganic Chemistry in Poznan. This cooperation helped to improve ModeLang and to make it easier to use. ModeLang can certainly be an interesting tool for all experts who analyse viral infections and it can become an important link between biologists and mathematicians specializing in a modelling software. Additionally, after preparation of suitable keywords dictionaries, it can be successfully used in other disciplines.

Details of the implementation of the language and results of its evaluation can be found in [WPB12].

Multi-agent model of HCV infection

Although differential equations constitute a very well-known and versatile methodology, they have some disadvantages. For example they do not allow to model space, add custom attributes, and their analysis involves advanced mathematical theory. That is why some other modelling techniques have been recently designed [MAW12, WJK$^+$10].One of them is an approach based on multi-agent simulations [AMDMV09]. There are a few authors that tried to utilize agents to model mainly HIV infection [ZKC05, MLF$^+$08, IKI$^+$10], but only one research group [IKI$^+$10] assumes that multi-agent simulations can be used for HCV. Moreover all of the existing work is rather only a basic example that this type of models can be successfully used to simulate viral infection. The proposed approaches lack clearly defined goals and algorithms for setting values of unknown parameters in a process similar to data fitting. Only [ZKC05]

presents an algorithm for verification of some commonly assumed theories about HIV infection in a general case.

To prove the usefulness of this type of modelling a comprehensive algorithm for construction and verification of multi-agent model of HCV infection was designed. To achieve this, first an example model designed by means of analogy with an already existing model [DMZ$^+$05] was prepared. Then a procedure for performing some more complex analyses using multi-agent simulation was designed. Finally, the algorithm for finding values of parameters existing in the model was proposed, which can be used to tune and verify the model for a specific patient.

The use of multi-agent simulation instead of differential equations has many advantages (see also [AMDMV09]), such as:

1. Interaction rules are written using biological terms instead of mathematical variables and parameters. Consequently, the rules better reflect the actual complexity of interaction network. In addition, they can be easily understood and defined by users without expert mathematical knowledge.
2. It is easier to modify rules. Usually after introducing a modification only a simple change in simulation's definition is required instead of a repetition of a complex mathematical analysis.
3. Objective function and constraints can be more complex. They do not have to be limited to data fitting but for example can define expected or maximal rates of some processes or change in defined moments.
4. It is possible do define precise spatial dependencies and distinguish between different cells of the same type. In differential equations all cells of the same type are assumed to behave identically, whereas in multi-agent systems their behaviour may depend on certain attributes (for example genetic mutations).
5. There are greater possibilities to analyse the results because each cell is simulated separately. Thereby, it can be analysed how a single cell or a group of cells affects the results of the experiment.
6. It is easy to model randomness by introducing random variables.

Some of the above advantages were demonstrated during the computational experiment which proved that it is trivial to define custom objective function and use them to find values of model's parameters. The experiment also proved that the genetic algorithm can be successfully used to optimize this function and draw valuable results.

Moreover the inverted simulation method was proposed. This method solves the problem of finding values of parameters used in models for specific patients.

Using this method anyone can analyse viral infections using techniques similar to models based on differential equations but making the use of all advantages of a multi-agent simulation. This method is an important tool that was not described in any earlier work related to the use of multi-agent systems in viral infections modelling.

Details of the implementation of the language and results of its evaluation can be found in [WJFB11, WJFB12].

Modelling HCV therapy efficiency

At present, all patients with chronic hepatitis C who meet certain standard inclusion criteria are subjected to the interferon-ribavirin treatment. Since the therapy is effective only for about 60%-80% of them, it would be advantageous to modify and restrict the criteria, in order to improve the treatment efficiency, i.e. to limit it to patients in whom the therapy benefits outbalance the risks. This appears to be of special importance in view of the fact that the administration of currently used medicines is often connected with significant side effects, including flulike gastrointestinal and psychiatric symptoms [MH00].

The aim of the designed algorithms was to propose a method for early stage hepatitis C virus (HCV) patients' assessment, under which predictions can be made about treatment efficiency. In consequence, a method of HCV population analysis was proposed that is useful for predicting CHC treatment outcomes. This method was then used to propose a mathematical approach that could support the process of patients' qualification for treatment.

Earlier it was proposed in [KFF+05] that phylogenetic trees and the mean Hamming distance of HCV populations can be useful in predicting a CHC treatment outcome. Starting from this fact, a linear regression analysis was employed to test the dependency between the genetic variability of the virus and its accumulation reflected in the RNA level in blood. R^2 coefficient for regression, confirmed by the F and t tests, turned out to be reasonably high (0.39). This allowed to use the level of viral RNA accumulation as a proxy for genetic variability. Then the types of responses to treatment were associated with the three patient groups (N, M, H), separated by the different virus RNA levels. Next, matrices describing transitions between the groups were constructed. Finally the therapeutic efficiency was analysed using unprocessed data and results of the algorithm that uses transition matrices. The results indicated that RNA accumulation below $5.25\frac{\log \text{ IU}}{\text{ml}}$ can be regarded as a threshold separating patients with high probability to develop a sustained response from

those whose response was null or temporary. Gathering more data from a larger group of patients could contribute to improving (*i.e.*, "tuning" the threshold) the current criteria of patient qualification for CHC therapy.

Detailed description of algorithms and results of computational experiments can be found in [WJK+10].

1 Wstęp

1.1 Rozwój technologii informatycznych a modelowanie infekcji wirusowych

Infekcje wirusowe są jednym z najgroźniejszych źródeł chorób u ludzi, zwierząt oraz roślin. Powodują wiele chorób o różnorodnych objawach oraz potrafią łatwo rozprzestrzeniać się w obrębie populacji. Pomimo wieloletnich badań wielu infekcji nadal nie można wyleczyć i w rezultacie mogą one prowadzić do ciężkich powikłań, a nawet śmierci. Mimo że pierwsze wirusy zostały odkryte i opisane już ponad sto lat temu [DEL07], mechanizm działania i reprodukcji wielu z nich jest nadal nieznany lub nie do końca rozumiany. W celu opracowania skutecznych terapii oraz szczepionek zabezpieczających przed infekcją ważne jest dokładne zbadanie i opisanie tych procesów. Dlatego też wirusologia jest w ostatnich dziesięcioleciach bardzo intensywnie rozwijającą się gałęzią biologii.

Jednak rozwój wirusologii, tak samo jak wielu innych dyscyplin naukowych, nie mógłby być tak szybki, gdyby nie był wspierany przez informatykę, która jest bezdyskusyjnie najszybciej rozwijającą się obecnie dyscypliną. Od lat czterdziestych minionego wieku, gdy skonstruowano pierwsze komputery [Gol80], do czasów obecnych, dokonał się w tej dyscyplinie niewyobrażalny postęp. Kom-

putery, o których sto lat temu wspominali co najwyżej wizjonerzy, obecnie towarzyszą nam na każdym kroku w postaci inteligentnych telefonów, potrafią dość dobrze tłumaczyć języki naturalne, prowadzić samochody i samoloty, a w najmocniejszych konfiguracjach instalowanych w centrach obliczeniowych, pod względem liczby wykonywanych operacji, przewyższają kilkukrotnie ludzki mózg. Towarzyszący rozwojowi komputerów rozwój technologi sieciowych oraz technik składowania, wymiany i analizy informacji doprowadził do tego, że aktualnie informacja stała się dobrem niematerialnym równie cennym jak dobra materialne, a społeczeństwo, w życiu którego stała się ona kluczowa, nazwano społeczeństwem informacyjnym.

Dzięki gwałtownemu rozwojowi technologii informatycznych opracowywane narzędzia komputerowe stały się nie tylko dobrem wykorzystywanym przez większość społeczeństwa w celach konsumpcyjnych, aby uczynić ich życie wygodniejszym i przyjemniejszym. Innowacyjne technologie informatyczne znalazły również ważne zastosowanie w praktycznie każdej dziedzinie nauki. Stwierdzenie to prawdziwe jest również w przypadku opisywania oraz analizowania infekcji wirusowych, gdzie znaczącą rolę odgrywają modele matematyczne i informatyczne, pozwalające lepiej zrozumieć zachowanie systemów biologicznych oraz prognozować ich zachowanie pod wpływem zmieniających się czynników zewnętrznych, takich jak zmiany zachodzące w organizmie lub aplikowana terapia. Aby zrozumieć poziom zaawansowania opisywanych systemów biologicznych wystarczy zdać sobie sprawę, że liczba interakcji i procesów, które występują w ludzkim organizmie może być liczona w dziesiątkach tysięcy, a gdyby mierzyć złożoność organizmu ludzkiego wyłącznie za pomocą skomplikowania jego kodu genetycznego, to zawarta w nim informacja byłaby porównywalna do 1 GB danych komputerowych. Takiej liczby procesów oraz ilości danych nie da się zanalizować ręcznie, nawet przy udziale wieloosobowego zespołu światowej klasy ekspertów. Dlatego też komputery stały się nieodłącznym narzędziem pracy każdego biologa, w szczególności wirusologa, a rozwój technik modelowania systemów biologicznych doprowadził do zdefiniowania nowego działu biologii nazywanego biologią systemową.

Modelowanie matematyczne i informatyczne w biologii to proces wieloetapowy. Najpierw należy zdefiniować model tak, aby odzwierciedlał obserwowaną rzeczywistość, na poziomie wystarczająco szczegółowym w prowadzonych badaniach. Ten proces zazwyczaj wymaga bliskiej współpracy biologów posiadających wiedzę dziedzinową oraz informatyków (matematyków). Następnie model jest analizowany, czasami analitycznie, a coraz częściej z użyciem oprogramowania symulacyjnego. Na etapie analizy modelu musi nastąpić jego weryfikacja w oparciu o rzeczywiste, zaobserwowane i zmierzone wartości, jego ewentualna korekta oraz dostrojenie występujących w nim parametrów. Ostatecznie

na podstawie opracowanego i zweryfikowanego modelu można wyciągnąć wnioski biologiczne. Jest to kluczowy etap całego procesu, gdyż dokonanie nowych obserwacji dotyczących analizowanego systemu jest głównym celem każdego procesu modelowania w biologii.

Przy realizacji procesu modelowania występują zazwyczaj dwa główne problemy. Po pierwsze problem komunikacji pomiędzy informatykami, a biologami. Eksperci specjalizujący się w modelowaniu potrafią doskonale wykorzystać istniejące narzędzia informatyczne i matematyczne, natomiast często mają duży problem ze zrozumieniem procesów biologicznych występujących w modelowanym systemie. Biolodzy natomiast bardzo dobrze znają i rozumieją działanie systemu, natomiast nie potrafią przełożyć tego na ścisłe i konkretne reguły, które mogliby przekazać zespołowi wspierającemu ich w modelowaniu. Powoduje to, że często współpraca przy modelowaniu systemu biologicznego ciągnie się miesiącami, a nawet latami, zanim uda się zbudować model wiernie odwzorowujący rzeczywistość.

Drugim problemem występującym w procesie modelowania jest niezgodność opracowanego modelu z rzeczywistością. Ponieważ systemy biologiczne są niezmiernie złożone, często opisane za pomocą kilkudziesięciu różnych zależności, które nie zawsze są dobrze poznane, a czasem w ogóle nieznane, może okazać się, że skonstruowany model nie będzie właściwie odzwierciedlał rzeczywistych zachowań. W takim przypadku warto, aby modyfikacje i ponowna weryfikacja były łatwe do przeprowadzenia, co niestety nie zawsze jest możliwe.

Pomimo następującego w ostatnich latach gwałtownego rozwoju technik modelowania informatycznego i matematycznego w biologii, w tym w wirusologii, opisane powyżej problemy pozostają w wielu przypadkach cały czas nierozwiązane. Co prawda zazwyczaj nie uniemożliwia to skonstruowania ostatecznego, poprawnego modelu, nie mniej zdecydowanie ten proces opóźnia, czyni go bardziej kosztownym i w przypadku infekcji wirusowych blokuje proces szybkiego opracowywania terapii i szczepionek. Zaproponowane w niniejszej pracy metody i algorytmy bioinformatyczne mają za zadanie pomóc w rozwiązaniu tych problemów, a przez to umożliwić zaoszczędzenie czasu i pieniędzy oraz ocalenie zdrowia i życia wielu pacjentów.

1.2 Cel i zakres pracy

W niniejszej pracy opisane zostały szczegółowo kolejne etapy procesu modelowania infekcji wirusowych oraz zdefiniowane i przetestowane metody i algorytmy, które mogą ten proces usprawnić. Większość opisywanych metod weryfi-

kowana była w pierwszej kolejności na podstawie wirusa HCV, ponieważ dzięki współpracy z Instytutem Chemii Bioorganicznej PAN w Poznaniu autor posiadał dobry dostęp do danych z badań klinicznych pacjentów zarażonych tym wirusem. Aby jednak zagwarantować, że metody będą wystarczająco elastyczne oraz umożliwić modelowanie infekcji innymi wirusami, większość z nich zweryfikowano również w oparciu o wirus HIV. Korzystając z konsultacji z biologami upewniono się, że będą one stosowalne również w przypadku innych wirusów. Wybór wirusów HIV i HCV ma również tą zaletę, że są to dwa wirusy, które stanowią obecnie bardzo duży problem medyczny na całym świecie.

Praca rozpoczyna się opisem aktualnie stosowanych podejść do formalnej definicji modeli biologicznych. Spektrum stosowanych w tym celu metod jest bardzo szerokie. Zaczyna się od definicji matematycznych, wykorzystujących wzory, równania i twierdzenia, przez formalne języki komputerowe służące do zapisania modelu w sformalizowany sposób, po oprogramowanie, które pozwala zbudować model w sposób wizualny wykorzystując gotowe komponenty. Niestety żadna z powyższych metod nie jest prosta do zrozumienia dla osoby nieposiadającej zaawansowanej wiedzy matematycznej i informatycznej, dlatego zaproponowany został nowy język nazwany ModeLang, służący do opisywania modeli biologicznych.

Gdy model zostanie zdefiniowany należy przeprowadzić jego analizę i weryfikację. W przypadku modeli opisujących infekcję wirusową w pojedynczym organizmie najpopularniejszym stosowanym w tym celu narzędziem jest analiza oparta o równania różniczkowe. Jest to dobrze zbadana i znana metoda matematyczna, jednak posiadająca pewne ograniczenia. Dlatego też zaproponowano i eksperymentalnie zweryfikowano nową metodę informatyczną opartą o symulację opartą na agentach. Jednocześnie pokazano jak ważne jest dokładne weryfikowanie opracowywanych modeli i przykład fałszywych wniosków, do których może doprowadzić zbyt pobieżnie zanalizowany model.

Praca kończy się definicją i analizą modelu przedstawiającego odpowiedź na terapię dla całej populacji pacjentów, która została jej poddana. Jest to temat dość rzadko poruszany w literaturze, gdyż zazwyczaj zakłada się, że w celu ocenienia skuteczności terapii wystarczy zastosować dobrze znane, nieskomplikowane metody statystyczne. Jak jednak pokazały prowadzone badania, na tym polu można cały czas opracowywać nowe, przydatne metody, które znacząco wspomogą proces projektowania nowych leków i terapii.

Głównymi celami niniejszej pracy były zatem:

1. Przegląd aktualnie stosowanych technik modelowania oraz analiza, weryfikacja i zaproponowanie ulepszeń do klasycznego modelu infekcji wirusowej.
2. Opracowanie intuicyjnego, zrozumiałego dla biologów języka opisu infek-

cji wirusowych, który mógłby być użyty do łatwego definiowania przez nich modeli systemów biologicznych, ich analizowania i w razie potrzeby modyfikowania.

3. Opracowanie środowiska symulacyjnego oraz uzupełniających je algorytmów do definiowania i analizowania modeli infekcji wirusowych, wspomagającego biologów na każdym etapie procesu modelowania.
4. Opracowanie ulepszonych metod pozwalających weryfikować metody terapii oraz warunki kwalifikacji do konkretnego sposobu leczenia na podstawie danych z populacji leczonych pacjentów.

Realizacja postawionych powyżej celów głównych wymagała realizacji następujących zadań szczegółowych:

- Zaprojektowanie składni języka służącego do opisu modeli infekcji wirusowych. Projekt powinien powstać w oparciu o konsultacje z biologami, aby powstały język był dla nich intuicyjny i wygodny w użyciu.
- Zaimplementowanie oraz przetestowanie parsera zaprojektowanego języka oraz umożliwienie wykorzystania sparsowanych opisów modeli w aplikacjach służących do ich analizy.
- Analiza i weryfikacja metod aktualnie stosowanych do modelowania infekcji wirusowych.
- Zaprojektowanie, zaimplementowanie i przetestowanie symulatora infekcji wirusowych opartego o systemy wieloagentowe, dla którego wejściem będą modele zdefiniowane w zaprojektowanym języku.
- Opracowanie metod i algorytmów weryfikacji, dostrajania i modyfikacji modeli infekcji wirusowych symulowanych w opracowanym symulatorze.
- Analiza przydatności opracowanego symulatora w konkretnych zastosowaniach biologicznych poprzez próbę zaproponowania ulepszeń klasycznych modeli infekcji wirusowych.
- Weryfikacja i zaproponowanie nowych metod modelowania infekcji wirusowych na poziomie populacyjnym.

Podział pracy jest następujący. W rozdziale 2 przedstawione zostały podstawy biologiczne niezbędne do zrozumienia motywacji i zastosowań wirusologicznych prezentowanych metod. W rozdziale 3 znajdują się podstawowe definicje oraz opis stosowanego aparatu matematycznego. Rozdział 4 zawiera obszerny przegląd aktualnie stosowanych metod modelowania infekcji wirusowych wraz z przykładowymi analizami przeprowadzonymi przez autora w oparciu o te narzędzia. Rozdział 5 przedstawia projekt i testy języka, który używany jest to

opisu modeli infekcji wirusowych. Język ten służy jako narzędzie do wprowadzania danych wejściowych do symulatora infekcji wirusowych, opartego o systemy wieloagentowe, opisanego w rozdziale 6. Rozdział 6 zawiera również opis metod opartych o algorytmy genetyczne służące do weryfikacji symulowanych modeli oraz eksperymentalną weryfikację opracowanych narzędzi. Rozdział 7 zawiera opis metod weryfikacji skuteczności terapii na poziomie populacyjnym w oparciu o opracowany algorytm wykorzystujący stochastyczne macierze przejść. Rozdział 8 zawiera podsumowanie całej pracy i weryfikację osiągnięcia jej celów. Praca kończy się dwoma dodatkami. W dodatku A przedstawiono kod kluczowych procedur i algorytmów oraz schematy opracowanych modeli. Do dodatku tego przeniesiono te elementy, które były zbyt długie i zbyt mało istotne, aby umieszczać je wewnątrz pracy. Umieszczenie ich w dodatku ma na celu poprawienie jej czytelności. Dodatek B zawiera tablice z wykorzystywanymi zbiorami danych.

2
Podstawy biologiczne

Rozdział ten zawiera podstawy biologiczne niezbędne do zrozumienia dalszej części pracy. Rozpoczyna się od opisu w sekcji 2.1 dyscypliny, którą jest biologia systemowa. Jest to stosunkowo nowy obszar biologii, do którego klasyfikowane są problemy opisane w tej pracy, ważne jest więc jego dokładne zdefiniowanie, w celu precyzyjnego umiejscowienia, gdzie w całym obszarze biologii znajdują zastosowanie prowadzone badania bioinformatyczne. Następnie w sekcji 2.2 znajduje się omówienie podstawowych terminów i zagadnień biologicznych wykorzystywanych w pracy. W sekcji 2.3 znajduje się dokładny opis infekcji i terapii osób zarażonych wirusem HCV. Informacje te zostały wyselekcjonowane tak, aby umożliwić łatwe zrozumienie dalszych rozdziałów, w których przedstawione są modele oraz metody analizy infekcji HCV. Ponieważ niektóre z tych modeli działają na bardzo szczegółowym biologicznie poziomie, konieczne było precyzyjne przedstawienie wykorzystywanych aspektów infekcji. Ostatnia sekcja 2.4 zawiera opis wirusa HIV, który wraz z HCV był używany do weryfikacji prezentowanych modeli.

2.1 Biologia systemowa

Przez setki lat rozwój biologii napędzany był poprzez podejście redukcjonistyczne, zgodnie z którym, aby zrozumieć działanie organizmów jako całości, należało wyodrębnić podstawowe procesy życiowe, które w nich występują, rozdzielić od siebie i zrozumieć każdy z nich osobno. Naukowcy byli przekonani, że jeżeli uda się dokładnie opisać każdy z procesów dostarczy to kompletnej wiedzy, z której będzie można bezpośrednio wyciągnąć wnioski dotyczące działania całego organizmu. Ciągły rozwój metod obserwacji umożliwiał obserwowanie organizmów na coraz większym poziomie szczegółowości poczynając od obserwacji działania całych organów, przez tkanki, do pojedynczych komórek i zachodzących w niej procesów, a nawet dzięki biochemii i biofizyce na poziomie pojedynczych cząstek i atomów. Doprowadziło to do precyzyjnego opisania setek procesów zachodzących w organizmach biologicznych, jednak mimo to nie dało odpowiedzi na wiele ważnych pytań dotyczących funkcjonowania tych organizmów. Dobrym przykładem mogą być komórki nerwowe w mózgu. Bardzo dobrze rozumiana jest zasada transmisji impulsów elektrycznych umożliwiająca komunikowanie się neuronów i przesyłanie informacji. Jednak w żaden sposób nie wyjaśnia to w jaki sposób powstaje myśl i jak przebiega proces myślowy.

Powyższe problemy zaczęły być dostrzegane w drugiej połowie minionego wieku, kiedy zauważono, że oprócz analizowania poszczególnych procesów konieczne jest spojrzenie bardziej globalne. Dało to początek nowej dziedzinie biologii nazywanej biologią systemową lub biologią systemów (ang. *systems biology*), która zajmuje się badaniem złożonych oddziaływań występujących w systemach biologicznych. Jednym z głównych celów biologii systemowej jest analizowanie tego, czego zabrakło w podejściu redukcjonistycznym, czyli analizowania własności objawiających się dopiero w kontekście globalnym (ang. *emergent properties*). W związku z tym biologia systemowa może być uznana za najważniejszego przedstawiciela podejścia holistycznego w biologii [Kit01].

Formalnie termin biologia systemowa został wprowadzony w 1966 roku przez Mihajlo Mesarovica poprzez organizację międzynarodowego sympozjum na temat „Systems Theory and Biology" [Mes68]. Jednak za pierwsze badania w tej dziedzinie można uznać opracowany już w 1952 roku model propagacji sygnału wzdłuż aksonów komórek nerwowych, który łączył wiedzę na temat zachowania cząsteczek sodu i potasu występujących w neuronach w celu zaobserwowania bardziej globalnego procesu transmisji impulsów [HH52]. Od tego czasu biologia systemowa była systematycznie rozwijana, z gwałtownym wzrostem zainteresowania tą tematyką w latach dziewięćdziesiątych minionego wieku spowodowanym pojawieniem się dużej liczby danych w obszarze ge-

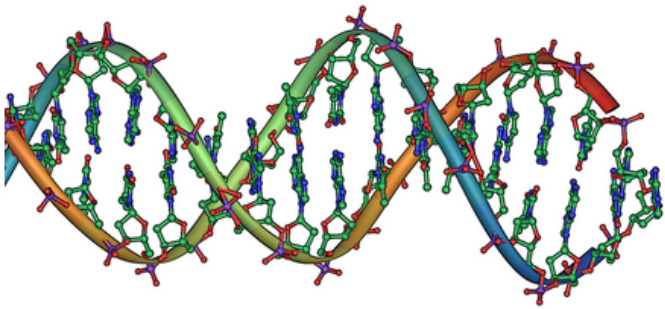
nomiki funkcjonalnej oraz pojawieniem się bardziej wydajnych komputerów. To właśnie szybko rozwijane i ulepszane komputery ułatwiły symulowanie i analizowanie definiowanych modeli. Jednym z większych osiągnięć tego okresu było zamodelowanie w 1997 roku metabolizmu całej, hipotetycznej komórki [THT+97]. Obecnie biologia systemowa to silnie interdyscyplinarna dziedzina, która łączy osiągnięcia dziedzin takich jak biologia, matematyka, informatyka, chemia i biochemia, fizyka i biofizyka, a nawet psychologia i socjologia. Natomiast badania prowadzone w ramach biologii systemowej wykorzystywane są na tak różnorodnych polach jak genomika, transkryptomika, proteomika, metabolomika i wiele innych.

2.2 Podstawowe zagadnienia biologiczne

2.2.1 DNA i RNA

Kwas deoksyrybonukleinowy, w skrócie DNA (ang. *Deoxyribonucleic acid*) stanowi podstawowy nośnik informacji w organizmach żywych. DNA zostało po raz pierwszy wyizolowane przez Friedricha Mieschera w 1869 roku [Dah08], jednak dopiero w 1953 roku James D. Watson i Francis Crick przedstawili rzeczywisty model budowy DNA [WC53]. Większość DNA przechowywana jest w jądrach komórkowych, a u organizmów prokariotycznych, które go nie posiadają, bezpośrednio w cytoplazmie. Cząsteczki DNA przechowują informację genetyczną zakodowaną za pomocą czterech nukleotydów – guaniny, adeniny, cytozyny i tyminy oznaczanych za pomocą wielkich liter G, A, C, T. Zazwyczaj DNA zbudowane jest z dwóch równoległych nici, w których szkielet zbudowany jest z cukru (deoksyrybozy) oraz grup fosforanowych połączonych z jedną z czterech powyższych zasad azotowych (G, A, C, T). W każdej nici wyróżnić można koniec 3' oraz 5' w zależności od tego, który węgiel deoksyrybozy posiada wolną grupę hydroksylową lub fosforanową. Układ nici określa się jako antyrównoległy, czyli każda cząsteczka DNA zaczyna się od końca 3' jednej nici oraz końca 5' drugiej. Nici DNA łączą się za pomocą wiązań wodorowych pomiędzy zasadami, przy czym guanina łączy się zawsze z cytozyną, a adenina z tyminą, co nazywane jest komplementarnością. W przestrzeni trójwymiarowej łańcuchy owijają się wokół wspólnej osi i tworzą tak zwaną podwójną helisę [AJL+02].

W komórkach DNA pogrupowane jest w długie, liniowe struktury zwane chromosomami. Dla przykładu u człowieka występuje 46 chromosomów, na



Rysunek 2.1: Struktura przestrzenna cząsteczki DNA. Źródło: wikimedia [http://commons.wikimedia.org/wiki/File:DNA˙double˙helix˙horizontal.png].

które składa się około 3 miliardy par zasad. Informacja zawarta w chromosomach przechowywana jest w sekwencjach zwanych genami. Geny organizmów eukariotycznych zawierają część kodującą określającą jak zbudować białko oraz część regulatorową określającą w jakich okolicznościach białko powinno być produkowane, z jaką częstotliwością, jak długo oraz w których komórkach. Część kodująca białka w celu opisania struktury białka używa kodu genetycznego, który wykorzystuje trójki nukleotydów zwane kodonami. Każdy kodon koduje jeden z dwudziestu możliwych aminokwasów występujących w białkach lub koniec sekwencji kodującej białko.

Kwas rybonukleinowy, w skrócie RNA (ang. *Ribonucleic acid*) ma budowę chemiczną zbliżoną do DNA. Jedynymi różnicami jest występowanie w szkielecie cukru rybozy oraz niewystępowanie tyminy, zamiast której pojawia się uracyl (oznaczany przez U). RNA powstaje zazwyczaj w procesie transkrypcji DNA, w czasie którego powstaje komplementarna do DNA nić RNA. W przeciwieństwie do DNA, RNA zazwyczaj występuje w postaci jednoniciowej i może tworzyć skomplikowane struktury przestrzenne. Istnieje wiele typów RNA, które mogą pełnić w organizmie różnorodne funkcje. Najistotniejsze z nich to RNA informacyjne (mRNA), które przenosi informację o strukturze białka z sekwencji DNA do rybosomów, w których białko jest syntezowane, RNA transferowe (tRNA), które transportuje do rybosomów aminokwasy oraz RNA rybosomalne (rRNA), które łączy poszczególne aminokwasy w białka. Oprócz powyższych funkcji różne rodzaje RNA pełnią również ważną rolę w procesach regulacji oraz ekspresji genów [BC99].

2.2.2 Wirusy

Nazwa wirus pochodzi od łacińskiego słowa *virus* oznaczającego truciznę. Wirus to mały mikroorganizm, który nie może rozwijać się i rozmnażać samodzielnie poza organizmem gospodarza [Sho08]. Wirusy zostały po raz pierwszy opisane przez holenderskiego mikrobiologa Martinusa Beijerincka w 1898 roku. Aktualnie szczegółowo opisane jest ponad 5000 gatunków wirusów oraz zidentyfikowane kilka milionów ich podtypów [DEL07]. Charakterystycznym dla wirusów sposobem funkcjonowania jest infekowanie przez wolną cząsteczkę wirusa, zwaną wirionem, pewnej komórki gospodarza i zmuszenie jej do produkowania tysięcy kopii wiriona. W przeciwieństwie do większości organizmów żywych wirusy nie składają się z komórek, które mogłyby się dzielić, tak więc są w pełni zależne od swojego gospodarza. Istnieje wiele sposobów przenoszenia się wirusów, takie jak kontakt fizyczny, transmisja poprzez powietrze lub przy pomocy innego organizmu zwanego wektorem, który nie jest zarażany, a tylko wykorzystywany do przekazania wirusa do innego organizmu. Ludzkie choroby wywoływane przez wirusy to na przykład grypa, świnka, AIDS, Ebola i SARS [DEL07].

Każdy wirus posiada informację o genach zakodowaną za pomocą długiej cząsteczki DNA lub RNA. Cząsteczka ta przechowywana jest w białkowej otoczce zwanej kapsydem, której zadaniem jest ochrona genów. Część wirusów posiada dodatkowo otoczkę lipidową – dodatkową błonę, której zadaniem jest ochrona wirusa przed wybranymi enzymami i innymi substancjami chemicznymi. Może ona pełnić również funkcję pomocniczą przy infekowaniu komórek. Większość wirusów ma długość średnicy zawartą między 10 i 300 nanometrów i nie może być obserwowana za pomocą mikroskopu optycznego. Aby zbadać ich strukturę wykorzystywane są mikroskopy elektronowe.

Często wirusy mogą być kompletnie wyeliminowane za pomocą układu immunologicznego danego organizmu. Po ich usunięciu z organizmu, zazwyczaj zyskuje on trwającą całe życie odporność na ten konkretny wariant wirusa. Cecha ta wykorzystywana jest przy tworzeniu szczepionek, które zawierają osłabione lub martwe postacie wirusa i pozwalają organizmowi wykształcić odporność bez potrzeby infekowania go. W przypadku zaistnienia infekcji podaje się leki, które można podzielić na trzy główne grupy:

- wzmacniające układ immunologiczny w walce z infekcją,
- utrudniające lub blokujące rozwój wirusa, na przykład poprzez blokowanie białek wirusa lub zwiększanie liczby mutacji genetycznych,
- blokujące białka receptorowe, uniemożliwiające przez to doczepienie się wirionów do błony komórkowej komórek gospodarza.

2.2.3 Drzewa filogenetyczne

Drzewo filogenetyczne to struktura, która wizualizuje relacje ewolucyjne w pewnym zbiorze bytów (na przykład gatunków biologicznych, sekwencji DNA lub języków) [KW01, BG07]. W drzewie tym byt (ang. *entity*), o którym zakłada się, że w procesie ewolucji bezpośrednio poprzedzał inny, jest jego ojcem. Struktura taka może być wykorzystana do ukazania relacji ewolucyjnej pomiędzy gatunkami. Drzewa filogenetyczne konstruowane są odkąd sformułowana została teoria Darwina. Początkowo były one stosunkowo proste. Pierwsze obszerne i rozbudowane drzewo filogenetyczne zostało skonstruowane w 1887 roku przez Englera i Prantla i zawierało klasyfikację większości roślin, które były w tamtym czasie znane [EP87]. Obecnie drzewa filogenetyczne są zazwyczaj konstruowane na podstawie kodu genetycznego z wykorzystaniem specjalistycznego oprogramowania, takiego jak na przykład MEGA3 [KTN04].

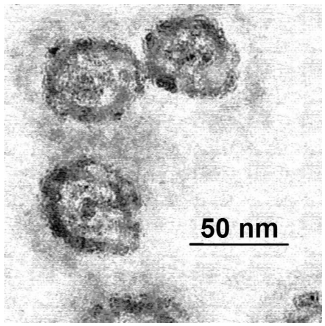
Specjalnym typem drzewa filogenetycznego jest drzewo bez wyróżnionego korzenia (ang. *unrooted tree*). Drzewo to ukazuje relację pomiędzy wszystkimi bytami bez wyróżniania jednego z nich jako przodka wszystkich pozostałych. Drzewo takie może być wykorzystane, aby graficznie reprezentować pewną populację. Jeżeli dwa osobniki z tej populacji są podobne, to w drzewie znajdą się blisko siebie. W rozdziale 7 drzewo takie zostało wykorzystane do wizualizacji populacji wirusów.

2.3 Wirus HCV

2.3.1 Budowa wirusa i przebieg infekcji

Wirus zapalenia wątroby typu C (WZW C, ang. *Hepatitis C Virus*, *HCV*) to otoczkowy, jednoniciowy wirus RNA należący do rodziny flawiwirusów (*Flaviviridae*) rodzaju *Hepacivirus*. Zakażenie wirusem następuje wskutek kontaktu z krwią nosiciela, dlatego zazwyczaj przyczyną zarażenia są operacje i zabiegi medyczne. Średni czas rozwoju choroby od momentu wniknięcia do krwiobiegu nosiciela do pojawienia się poważnych objawów jest bardzo długi i wynosi od 5 do 35 lat. Dlatego też większość nosicieli nie jest świadoma obecności wirusa. Docelowo infekcja HCV może prowadzić do marskości i raka wątroby oraz towarzyszących im innych poważnych schorzeń [ACB99, See02].

Wirus HCV to jeden z najszerzej rozpowszechniony ludzkich patogenów występujących na świecie. Według szacunków WHO zarażonych jest nim ponad

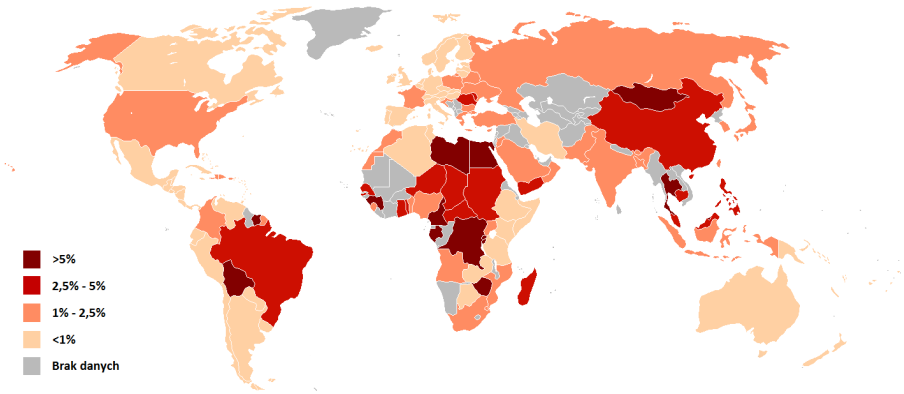


Rysunek 2.2: Wirus zapalenia wątroby typu C widoczny w mikroskopie elektronowym. Źródło: wikipedia [http://pl.wikipedia.org/w/index.php?title=Plik:Em˙flavavirus-HCV˙samp1c.jpg].

170 milionów ludzi [Wor00], a według Polskiej Grupy Ekspertów HCV zarażonych jest nim około 700 tysięcy polaków (1,5% populacji) [Pol07]. Obszar występowania wirusa HCV w 1999 roku, kiedy opracowane zostało dokładne zestawienie dla całego świata, prezentowany jest na rysunku 2.3.

HCV ma budowę sferyczną o średnicy cząstki wynoszącej ok. 50 nm. Zbudowany jest z rdzenia zawierającego kod RNA oraz otoczki [OD03]. Genom HCV stanowi jednoniciowy, liniowy RNA o polaryzacji dodatniej, zbudowany z 9600 nukleotydów. Białka kodowane przez genom dzieli się na strukturalne (rdzeniowe C oraz otoczkowe E1 i E2), niestrukturalne (biorące udział w namnażaniu wirusa: NS2, NS3, NS4a, NS4b, NS5a, NS5b) oraz białko p7 [De 99, DPF09]. Istotnym faktem jest, iż tempo mutacji wirusa jest bardzo szybkie (ok. 1000 razy większe niż u człowieka), co u pojedynczego pacjenta może w ciągu doby generować ok. 10^9-10^{12} wariantów wirusa. Dlatego też w przypadku wirusa HCV u zainfekowanego człowieka mówimy o heterogennej genetycznie populacji [CR11].

Na podstawie badania filogenetycznego genomu wirusa HCV w obszarach kodujących białka NS5 i 5'UTR wyodrębnione zostało 7 rodzajów jego genotypów [NLL$^+$12, CR11]. Genotypy różnią się pomiędzy sobą rodzajem około 30%-35% nukleotydów. W obrębie genotypów wyodrębniane są subtypy o zmienności około 20%-25%, oznaczane małymi literami, a w ich obrębie tak zwane izolaty o zmienności 10%-15%. Znajomość genotypu wirusa jest szczególnie istotna na etapie terapii. Wiadomo na przykład, że genotypy 1a i 1b są odpowiedzialne za około 60% infekcji, występują na całym świecie i jednocześnie są stosunkowo słabo podatne na standardowo stosowaną terapię [SBC$^+$05, YC09].



Rysunek 2.3: Obszar występowania wirusa HCV w 1999 roku. Na mapie zaznaczono procentowy udział osób zainfekowanych w całkowitej populacji kraju. Źródło: wikimedia [http://commons.wikimedia.org/wiki/File:HCV˙prevalence˙1999.png].

W pojedynczym zainfekowanym organizmie wirus HCV występuje jako zbiór pseudotypów (ang. *quasispecies*), które są ze sobą skorelowane filogenetycznie, ale przy tym ich kod genetyczny jest zauważalnie różny (do 5% zmienności) [CR11, FAKKF03, Sim04].

2.3.2 Diagnostyka

W celu zdiagnozowania choroby, a następnie oszacowania postępu zakażenia i szans na jego wyleczenie zastosować można kilka technik. Najprostszym i najtańszym badaniem jest sprawdzenie obecności przeciwciał anty-HCV. Brak przeciwciał wyklucza infekcję, jednak ich obecność nie jest warunkiem wystarczającym do stwierdzenia zakażenia. Przeciwciała te mogą pojawić się również gdy wirus został skutecznie, samodzielnie wyeliminowany przez organizm lub w przypadku niektórych chorób wewnętrznych. Aby potwierdzić obecność wirusa należy zweryfikować obecność jego RNA we krwi metodą PCR.

Powyższe testy nie umożliwiają zdiagnozowania jak poważna jest u pacjenta choroba spowodowana infekcją. Aby ocenić szkody wyrządzone w organizmie przeprowadza się badanie poziomu aminotransferazy alaninowej we krwi (w skrócie ALT lub alaty), czyli enzymu biorącego udział w przemianach białek. Na tej podstawie można prognozować na ile zaawansowana jest infekcja. Badania wykonywane tą metodą są niestety obarczone dużym błędem, a jedynym

skutecznym sposobem diagnostyki jest biopsja wątroby, która niesie ze sobą pewne ryzyko komplikacji [Ros11, Sto01]. Biopsja zazwyczaj wymaga również hospitalizacji, przez co jest dużo bardziej uciążliwa dla pacjenta.

Dużo droższa i bardziej skomplikowana jest analiza oparta o kod genetyczny wirusa. Z tego powodu w praktyce klinicznej stosowana jest ona rzadko, a wykorzystywana głównie w celach badawczych. Obszerny przykład wykorzystania tych danych opisany jest w rozdziale 7. Aby skorzystać z tej metody należy pobrać próbkę złożoną z co najmniej kilkudziesięciu kopii RNA wirusa pochodzących z populacji wirusa u tego samego pacjenta, a następnie przeprowadzić jego sekwencjonowanie. Na podstawie kodu genetycznego można zdefiniować następujące charakterystyki:

- Złożoność pseudotypów (ang. *quasispecies complexity*), czyli liczbę różnych sekwencji genetycznych występujących w sekwencjonowanej populacji. W szczególności można zdefiniować minimalną liczbę zasad, którą muszą różnić się sekwencje, aby uznać je za różne.
- Średnia odległość Hamminga (ang. *Mean Hamming Distance*, *MHD*), czyli średnia liczba różnic pomiędzy każdą parą sekwencji. Formalnie, jeżeli $a_{i,j}$ będzie oznaczało j-ty aminokwas w i-tej sekwencji, mamy:

$$MHD = \frac{2}{n(n-1)} \sum_{i=1}^{n} \sum_{k=i+1}^{n} \sum_{j} H(a_{i,j}; a_{k,j}) \tag{2.1}$$

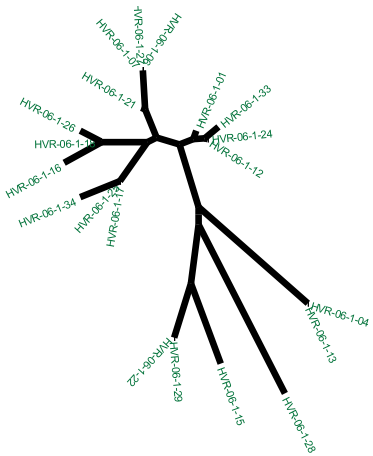
  gdzie:

$$H(a;b) = \begin{cases} 1 & \text{dla } a \neq b \\ 0 & \text{dla } a = b \end{cases} \tag{2.2}$$

- Drzewo filogenetyczne, czyli analiza struktury drzewa utworzonego z pobranych z organizmu kopii wirusa. Przykładowe drzewo filogenetyczne utworzone na tej podstawie prezentuje rysunek 2.4.

2.3.3 Terapia

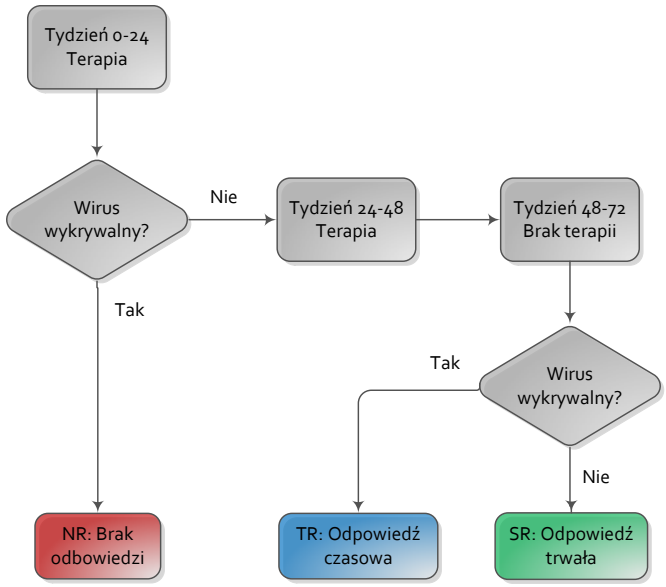
Standardowa terapia infekcji HCV polega na podawaniu dwóch leków. Interferon alfa lub interferon pegylowany zwiększa prawdopodobieństwo mutacji wirusa w celu wygenerowania większej liczby błędnych genomów, natomiast rybawiryna wspomaga odpowiedź immunologiczną organizmu. Przebieg terapii w czasie prezentuje schemat 2.5. Przez pierwsze 24 tygodnie podawane są leki i jeżeli nie spowoduje to usunięcia wirusa z organizmu terapia jest przerywana,



Rysunek 2.4: Przykładowe drzewo filogenetyczne dla populacji HCV pobranej od jednego pacjenta.

a pacjent klasyfikowany jako niereagujący na terapię (ang. *no response*, *NR*). Jeżeli wirus jest niewykrywalny, terapia kontynuowana jest przez kolejne 24 tygodnie. Po jej zakończeniu czeka się kolejne 24 tygodnie na weryfikację wyleczenia. Jeżeli mimo wstępnie pozytywnego wyniku wirus ponownie zostanie wykryty pacjent klasyfikowany jest jako osoba z przejściową odpowiedzią (ang. *transient response*, *TR*) [MP99]. Sytuacja ta może zajść, jeżeli po 24 tygodniu terapii poziom wirusa spadł tak dramatycznie, że nie dało się go wykryć dostępnymi metodami, natomiast nie został w pełni wyeliminowany i mimo kontynuacji leczenia uodpornił się na terapię. W przeciwnym wypadku pacjent klasyfikowany jest jako wyleczony (ang. *sustained response*, *SR*) [PH01].

Skuteczność terapii zależna jest od genotypu obecnego we krwi. Dla genotypu 1 i 4 wynosi ok. 65%-70%, dla genotypu 2 i 3 ok. 90%, a dla genotypu 6 ok. 80% [YC09]. Cały czas prowadzone są badania w celu poprawy tej skuteczności, szczególnie wobec najtrudniejszych do wyleczenia genotypów. Aktualnie badanych jest około 100 różnych, nowych lekarstw [EKWR12]. Aktywnie poszukiwana jest również szczepionka przeciw wirusowi HCV, która dotychczas nie została wprowadzona na rynek. Nie mniej część prowadzonych badań wygląda bardzo obiecująco [TJW11].

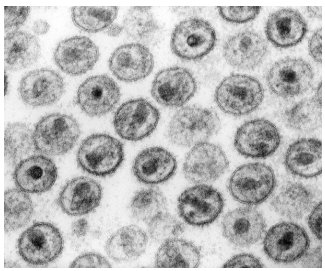


Rysunek 2.5: Schemat terapii infekcji HCV.

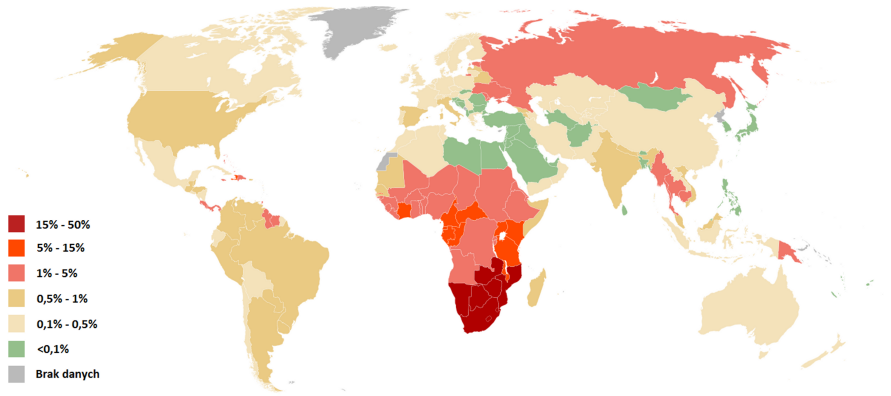
2.4 Wirus HIV

Ludzki wirus niedoboru odporności (ang. *human immunodeficiency virus*, *HIV*) to otoczkowy wirus o kulistej budowie należący do rodzaju lentiwirusów (*Lentivirus*) z rodziny retrowirusów (*Retroviridae*) [DRK09, Wei93]. Wirus HIV przenoszony jest drogą płciową oraz poprzez kontakt z zakażoną krwią, a infekuje komórki układu immunologicznego takie jak pomocnicze limfocyty T $CD4^+$, makrofagi i komórki dendrytyczne [CDH^+10]. Wirus HIV wywołuje zespół nabytego niedoboru odporności, czyli AIDS (ang. *Acquired Immunodeficiency Syndrome*), w czasie którego coraz poważniejsze uszkadzanie układu immunologicznego uniemożliwia obronę przez zagrażającymi życiu infekcjami oraz ułatwia rozwinięcie się raka. Okres inkubacji (okres do pojawienia się pierwszych objawów choroby) wynosi zazwyczaj od pół do trzech lat.

Struktura HIV dość znacząco odróżnia go od innych retrowirusów. Jest kulistego kształtu, ma średnicę 120 nm, co czyni go 60 razy mniejszym niż czerwone krwinki, ale cały czas dość dużym w porównaniu z innymi wirusami [MCGS02, RD02]. Genom HIV stanowi jednoniciowy, liniowy RNA o polaryzacji dodatniej, kodujący 9 genów i zamknięty w otoczce kapsydowej. Podobnie jak w przypadku wirusa HCV wirus HIV cechuje się bardzo dużą zmiennością genetyczną. W zainfekowanym organizmie w ciągu doby powstaje około 10^{10} nowych wirionów, przy czym prawdopodobieństwo mutacji każdego nukleotydu wynosi około $3 \cdot 10^{-5}$ [RHS95, RPCH04]. Znane są dwa typy wirusa HIV. Występujący na całym świecie HIV-1 oraz mniej zakaźny i wolniej rozwijający się,



Rysunek 2.6: Zdjęcie wirusów HIV-1 wykonane mikroskopem elektronowym. Źródło: wikipedia [http://pl.wikipedia.org/w/index.php?title=Plik:HIV-1˙Transmission˙electron˙micrograph˙AIDS02bbb˙lores.jpg].



Rysunek 2.7: Rozpowszechnienie HIV wśród dorosłych na koniec roku 2005. Na mapie zaznaczono procentowy udział osób zainfekowanych w całkowitej populacji kraju. Źródło: wikimedia [http://pl.wikipedia.org/w/index.php?title=Plik:AIDS˙and˙HIV˙prevalence.svg].

występujący głównie w Afryce Zachodniej, wirus HIV-2 [GME+03].

Pierwszym testem wykonywanym w celu zdiagnozowania infekcji HIV jest badanie przesiewową metodą immunoenzymatyczną (ELISA). Jeżeli test wykaże obecność przeciwciał anty-HIV-1, w celu jego weryfikacji jest on powtarzany. Jeżeli wynik obu testów jest pozytywny, pacjent klasyfikowany jest jako powtarzalnie dodatni i przeprowadzone zostają dokładniejsze testy potwierdzające infekcję na przykład za pomocą testu Western blot [Cen01]. Współczesna diagnostyka infekcji HIV jest prawie nieomylna – szacuje się, że wynik fałszywie pozytywny zdarza się raz na około 250 tysięcy przypadków [CCL+91]. Dotychczas nie udało się opracować skutecznej szczepionki przeciw wirusowi HIV, a leczenie polega na podawaniu pacjentowi kilku różnych leków mających zablokować różne etapy rozwoju wirusa. Jest to tak zwana intensywna terapia antyretrowirusowa (HAART), która zdecydowanie spowalnia rozwój AIDS [MI11].

Wirus HIV został po raz pierwszy zaobserwowany w Stanach Zjednoczonych w 1981 roku [TS10], a wywoływana przez niego choroba AIDS nazwana i opisana rok później [SC88]. W roku 1983 udało się laboratoryjnie wyizolować wirus [GSG+83], a obecnie jest to jeden z lepiej poznanych wirusów. Podejrzewa się, że źródłem wirusa HIV jest zmutowany wirus występujący u afrykańskich małp – dla wirusa HIV-1 u szympansów [GBR+99], natomiast dla HIV-2 u mangaby szarej [GME+03]. Szacuje się, że wirus rozwinął się na początku XX wieku [Wot01], choć pierwszy dobrze udokumentowany przypadek infekcji HIV

wystąpił w Kongu dopiero w roku 1959 [McN10].

3 Podstawy matematyczne i informatyczne

3.1 Modelowanie matematyczne

Celem modelowania matematycznego procesów biologicznych jest sformułowanie równań, które będą w stanie w sposób obliczeniowy przewidzieć dynamiczne zachowanie się systemów biologicznych. Ze względu na złożoną strukturę rzeczywistości jest to zadanie, którego nigdy nie da się zrealizować z pełną skutecznością. Jednakże w wielu przypadkach można osiągnąć zadowalającą dokładność, zgodnie ze spostrzeżeniem Rodneya Brooksa z 1990 roku [Bro90]:

> *„...świat sam w sobie jest swoim najlepszym modelem. Zawsze jest idealnie aktualny. Zawsze zawiera każdy szczegół, który powinien być znany. Sztuką natomiast jest monitorować go wystarczająco dokładnie i często."*

W odróżnieniu od modeli typu czarna skrzynka, w modelu matematycznym byty (ang. *entities*) występujące w systemie mają swoją bezpośrednią reprezentację [MT08]. Modelowanie służy zazwyczaj pogłębieniu rozumienia pewnego procesu i składa się z następujących etapów:

1. Formułowanie hipotezy badawczej na podstawie przeprowadzonych obserwacji.

2. Matematyczne definiowanie modelu stanowiącego opis modelowanego systemu.
3. Weryfikacja modelu w oparciu o zebrane dane eksperymentalne.
4. W razie potrzeby weryfikacja i formułowanie kolejnej hipotezy badawczej, jeżeli wyniki symulowania modelu są niezgodne z zachowaniami i danymi zaobserwowanymi eksperymentalnie.

Modelowanie matematyczne jest stosowane w wielu różnych dziedzinach wiedzy takich jak biologia, chemia, medycyna, fizyka, ekonomia i wiele innych. Głównymi celami stosowania modelowania w biologii i medycynie są:

- przetworzenie i analiza wielu serii skomplikowanych danych, często pochodzących z wielu eksperymentów, których nie da się łatwo zinterpretować w oparciu o surowe wyniki lub ich prostą wizualizację komputerową,
- wstępna weryfikacja nowych metod leczenie lub innego oddziaływania na organizmy,
- znajdowanie nowych, obiecujących kierunków badawczych i ich wybór w taki sposób, aby zaoszczędzić czas, pieniądze oraz liczbę wymaganych do testów zwierząt,
- komunikowanie innym badaczom w precyzyjny i czytelny sposób opracowanych wyników dotyczących zachowań organizmów żywych.

W każdym modelu matematycznym, niezależnie w jaki sposób jest on zapisany, można wyróżnić następujące elementy składowe:

- zmienne, które modelują pewne cechy systemu rzeczywistego,
- warunki początkowe, czyli wartości zmiennych, którymi inicjalizuje się je przed rozpoczęciem analizowania modelu,
- parametry umożliwiające dopasowanie jednego modelu do wielu systemów tego samego typu, różniących się tylko jakąś charakterystyką (na przykład prędkością zachodzenia pewnych reakcji),
- reguły zapisane w charakterystyczny dla danej techniki modelowania sposób, definiujące relacje pomiędzy zmiennymi i parametrami.

Opracowanych zostało wiele różnorodnych metod modelowania matematycznego. Modele oparte na regresji wykorzystują równania algebraiczne [SW03], modele Boolowskie zbudowane są z bramek logicznych [SRAE+09], sieci Baysowskie [Jen96] oraz modele stochastyczne [TSB04, Gil07] badają rozkłady prawdopodobieństwa, natomiast modele mechanistyczne wykorzystują zwyczajne równania różniczkowe [ABLS06, MT08], bądź gdy konieczne jest zamodelowa-

nie przestrzeni równania różniczkowe cząstkowe [Mur03].

Z powyżej wymienionych technik modelowania praca ta koncentruje się głównie na modelach opartych o zwyczajne równania różniczkowe, które szczegółowiej opisane zostały w kolejnej sekcji. W pracy wykorzystano również stosunkowo nową technikę modelowania opartego o symulację wieloagentową. Ten temat dokładniej opisany jest w sekcji 3.6.

W przypadku modelowania matematycznego w biologii ważne jest, aby rozumieć na jakim poziomie hierarchii organizacyjnej życia operują projektowane modele. W zależności od autora i szczegółowości przyjętego rozgraniczenia hierarchia ta definiowana jest różnie, jednak zawsze można wyodrębnić następujące podstawowe poziomy organizacji [FCP+11]:

1. Pojedyncze komórki i zachodzące w nich zjawiska, w tym procesy genetyczne i proteomiczne.
2. Tkanki jako zorganizowane struktury komórek określonego typu lub typów.
3. Organy i układy znajdujące się w ciele wykorzystujące tkanki i komórki do pełnienia bardziej zaawansowanych funkcji.
4. Całe, w pełni funkcjonalne i samodzielne organizmy.
5. Populacje wielu organizmów żyjących na zbliżonym obszarze i ich wzajemne oddziaływanie ze środowiskiem życia oraz między sobą.

Ponieważ infekcje wirusami HIV i HCV modelowane w tej pracy dotyczą głównie pojedynczego organu lub układu praca ta koncentruje się głównie na poziomie trzecim, gdyż poziom czwarty byłby zbyt skomplikowany i niepotrzebnie utrudniał modelowanie, a poziom drugi zbyt prosty. W rozdziale 7 omówiony jest natomiast poziom piąty – populacyjny. W przypadku omawianych wirusów poziom pojedynczych cząsteczek jest mniej interesujący ponieważ same mechanizmy infekcji pojedynczych komórek są dość dobrze znane, tak więc poziom ten został pozostawiony do rozważenia w przyszłości, jeżeli okaże się, że wyniki te mogą być wartościowe dla wyższych poziomów modelowania.

3.2 Równania różniczkowe

Równanie różniczkowe to równanie, w którym występuje zarówno pewna funkcja jak i jej pochodna [Arf85]. Równania te odgrywają bardzo ważną rolę w inżynierii, fizyce, ekonomii, biologi, chemii i wielu innych dyscyplinach [Str01].

Wykorzystywane są, gdy jakaś ciągła, deterministyczna i zmieniająca się wartość jest zdefiniowana poprzez definicję tempa tych zmian (wyrażonego jako pochodna funkcji opisującej tą wartość). Jeżeli rozważany jest zbiór wartości, w którym każda z nich zależy również od innych, to sytuacja taka może być opisana za pomocą układu równań różniczkowych.

Równania różniczkowe można podzielić na liniowe i nieliniowe. Liniowe równania różniczkowe zdefiniowane są następująco:

$$a_n(x)\frac{d^n y}{dx^n} + a_{n-1}(x)\frac{d^{n-1} y}{dx^{n-1}} + \ldots + a_1(x)\frac{dy}{dx} + a_0(x)y = f(x), \tag{3.1}$$

gdzie a_n,a_{n-1},...,a_0 oraz f są pewnymi funkcjami, a x oraz y są zmiennymi. Najprostszym przykładem liniowego równania różniczkowego jest:

$$\frac{dU}{dt} = U. \tag{3.2}$$

Równanie różniczkowe jest nieliniowe jeżeli którakolwiek ze zmiennych zależnych lub ich pochodnych występuje w potędze wyższej niż 1 lub równanie zawiera ich iloczyn. Najprostszym przykładem nieliniowego równania różniczkowego jest:

$$\frac{dU}{dt} = U^2. \tag{3.3}$$

Inną ważną charakterystyką równania różniczkowego jest jego stopień. Jest on równy najwyższemu stopniowi spośród występujących w nim pochodnych.

Pod względem typu zmiennych oraz pochodnych występujących w równaniu, równania różniczkowe mogą być podzielona na:

- Równania różniczkowe zwyczajne (ang. *Ordinary differential equations*, *ODE*) – nieznana funkcja jest funkcją jednej zmiennej,
- Równania różniczkowe cząstkowe (ang. *Partial differential equations*, *PDE*) – równanie, które zawiera pochodne cząstkowe funkcji wielu zmiennych,
- Równania różniczkowe z opóźnionym argumentem (ang. *Delay differential equations*, *DDE*) – pochodne nieznanej funkcji zależą od jej wartości w przeszłości,
- Równania różniczkowe stochastyczne (ang. *Stochastic differential equations*, *SDE*) – równanie, w którym występują wyrażenia będące procesami stochastycznymi.

Rozwiązanie równania różniczkowego zazwyczaj polega na znalezieniu funkcji, dla której zdefiniowana przez równanie zależność jest prawdziwa. Proces ten jest wykonalny w sposób analityczny jedynie dla stosunkowo prostych równań.

Bardziej skomplikowane równania muszą być rozwiązywane używając ich numerycznej aproksymacji. Nie mniej nawet bez dokładnego rozwiązania można analitycznie wyznaczyć wiele przydatnych charakterystyk. Najważniejszymi z nich są:

- Stany stabilne systemu. Bardzo często wartości funkcji stabilizują się po pewnym czasie na jednej określonej wartości i od tego momentu pozostają niezmienne. Wartości takie nazywa się stanami stabilnymi.
- Bifurkacje, czyli krytyczne wartości parametrów lub zmiennych. Wartości te są wartościami granicznymi dla dwóch różnych zachowań równania. Na przykład w zależności od tego czy wartość parametru jest mniejsza, czy większa, równanie może dążyć do stanów stabilnych o różnej wartości.

3.3 Macierze stochastyczne i łańcuchy Markowa

Macierzą stochastyczną (ang. *stochastic matrix*) nazywamy macierz kwadratową, której elementami są nieujemne liczby rzeczywiste i w której suma elementów w każdym wierszu (tzw. prawa macierz stochastyczna) lub w każdej kolumnie (tzw. lewa macierz stochastyczna) sumuje się do 1. Ciekawą własnością macierzy stochastycznych jest to, że ich wartość własna zawsze wynosi 1. Przykładową macierzą stochastyczną (prawą) jest:

$$\begin{bmatrix} 0{,}7 & 0{,}2 & 0{,}1 \\ 0 & 0{,}6 & 0{,}4 \\ 0{,}01 & 0{,}69 & 0{,}3 \end{bmatrix} \tag{3.4}$$

Łańcuchy Markowa zostały zdefiniowane przez rosyjskiego matematyka Andreya Markova w 1906 roku. Niech M oznacza pewien skończony i policzalny zbiór, nazywany przestrzenią stanów. Niech dana będzie następująca funkcja losowa X, której każda wartość leży w przestrzeni pewnego zdarzenia losowego, którego wynikiem jest jedno ze zdarzeń ze zbioru M:

$$\{ X^{(k)} \in M,\, k = 0, 1, 2, \ldots \} \tag{3.5}$$

Powyższa funkcja nasywana jest procesem stochastycznym X, natomiast proces X nazywany jest łańcuchem Markowa jeżeli dla każdego $i,j \in M$ istnieje określona, niezależna od czasu wartość $P_{i,j}$ reprezentująca prawdopodobieństwo, że stan procesu zmieni się na stan j jeżeli aktualnie znajduje się on w

stanie i, taka że [CNBC+06, GP01, GKP08]:

$$P_{i,j} = P(X^{k+1} = i | X^{(k)} = j, X^{(k-1)} = i_{k-1}, \ldots, X^{(0)} = i_0), \;\; k \geq 0 \qquad (3.6)$$

gdzie $i_0, i_1, \ldots, i_{k-1} \in M$ są dowolnymi stanami. Równanie 3.6 oznacza, że mając dane stany z przeszłości $X^{(0)}, X^{(1)}, \ldots, X^{(k-1)}$ oraz stan bieżący $X^{(k)}$ przyszły stan $X^{(k+1)}$ jest niezależny od stanów poprzednich i zależy wyłącznie od stanu bieżącego. Czasami własność tą nazywa się brakiem pamięci. Wartość prawdopodobieństwa musi być z definicji większa lub równa 0, a suma prawdopodobieństw w każdym wierszu macierzy P, zawierającej prawdopodobieństwa przejść pomiędzy każdą możliwą parą stanów, musi być równa 1. Macierz P nazywana jest macierzą przejść pomiędzy stanami procesu i zgodnie z powyższą definicją jest zawsze prawą macierzą stochastyczną:

$$P_{i,j} \geq 0, \quad \sum_{j \in M} P_{i,j} = 1 \qquad (3.7)$$

Ciekawą własnością tej macierzy jest fakt, iż jeżeli przemnożymy ją przez siebie nieskończoną liczbę razy (podniesiemy do nieskończonej potęgi), to wszystkie wiersze macierzy wynikowej będą identyczne (w każdej kolumnie będzie występowała tylko jedna, identyczna wartość). Wartość ta oznacza prawdopodobieństwo, że proces znajdzie się w stanie reprezentowanym przez daną kolumnę, jeżeli wykona się nieskończoną liczbę razy. W praktyce proces nie musi powtórzyć się nieskończoną liczbę razy, ale n razy, gdzie n jest wystarczająco dużo, aby różnica wartości każdego elementu macierzy P^n i P^{n+1} była stosunkowo mała (na przykład 10^{-6}).

Następujący przykład dobrze demonstruje ideę łańcuchów Markowa. W pewnym mieście są dwa kluby sportowe (A i B), tak więc każdy mieszkaniec musi być zrzeszony w jednym z nich. Załóżmy, że każdy mieszkaniec wykupuje roczne członkostwo w którymś klubie i przez cały rok nie może go zmienić. Jednak po roku możliwa jest zmiana członkostwa na konkurencyjny klub. Niech prawdopodobieństwo, że członek klubu A zmieni członkostwo na klub B wynosi 30%, a prawdopodobieństwo zmiany z B na A wynosi 40%. W takim przypadku istnieją dwa stany procesu (członkostwo w klubie A lub B), a więc $M = \{0, 1\}$ oraz:

$$P = \begin{bmatrix} 70\% & 30\% \\ 40\% & 60\% \end{bmatrix} \qquad (3.8)$$

Po kilku mnożeniach macierzy P przez siebie otrzymujemy:

$$P^8 = \begin{bmatrix} 57.146\% & 42.854\% \\ 57.139\% & 42.861\% \end{bmatrix} \tag{3.9}$$

$$P^9 = \begin{bmatrix} 57.144\% & 42.856\% \\ 57.142\% & 42.858\% \end{bmatrix} \tag{3.10}$$

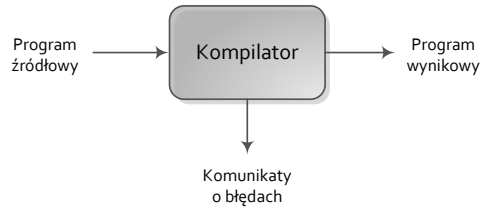
Macierze P^8 i P^9 przedstawione na równaniach 3.9 i 3.10 opisujące rozkład prawdopodobieństw po 8 i 9 latach są wystarczająco zbliżone, aby wnioskować, że rozkład prawdopodobieństwa znacząco się już nie zmieni w kolejnych latach. Tak więc ostatecznie, po upływie wystarczająco długiego czasu, około 57% mieszkańców będzie klientami klubu A, natomiast około 43% będzie klientami klubu B. Wniosek ten będzie prawdziwy oczywiście tylko jeżeli macierz P nie będzie ulegać zmianom, co jednak jest gwarantowane przez definicję łańcuchów Markowa.

Łańcuchy Markowa są powszechnie wykorzystywane w fizyce, statystyce, ekonomii, grach losowych, muzyce, a nawet przy analizie gier w baseball. Jednak prawdopodobnie najczęściej wykorzystywanym na świecie zastosowaniem jest użycie ich przez silnik wyszukiwarki Google do szeregowania znalezionych stron według istotności [PBMW98].

3.4 Języki formalne i kompilatory

Z matematycznego punktu widzenia język formalny to podzbiór zbioru wszystkich słów nad skończonym alfabetem. W praktyce oznacza to, że język formalny to ściśle zdefiniowany sposób zapisywania pewnych treści, który dopuszcza użycie tylko wybranych ciągów słów i liter i nie dopuszcza żadnych odstępstw od zdefiniowanych reguł. Języki formalne są powszechnie wykorzystywane w informatyce do definiowania języków programowania, ale znajdują również szerokie zastosowanie w matematyce i językoznawstwie. W matematyce i informatyce, które zazwyczaj nie zajmują się językami naturalnymi, przymiotnik „formalny" jest często pomijany. Do zdefiniowania języka formalnego wykorzystuje się podział na symbole terminalne (mogące pojawić się w opisie napisanym w danym języku) oraz nieterminalne (pełniące rolę pomocniczą). Wykorzystując te symbole definiuje się reguły, zwane produkcjami, które określają jakie ciągi symboli są dozwolone.

Podstawowym zadaniem kompilatora jest czytanie pliku zapisanego w pewnym formalnym języku źródłowym i konwertowanie go na inny język wynikowy.



Rysunek 3.1: Podstawowa architektura kompilatora.

W trakcie tego procesu, przedstawionego na rysunku 3.1, bardzo ważną funkcjonalnością jest informowanie użytkownika o napotkanych błędach. W praktyce proces kompilacji jest zazwyczaj bardziej złożony i w przypadku większości języków programowania składa się z następujących etapów [ASU86]:

1. **Preprocesing**, w trakcie którego część konstrukcji zastosowanych w pliku źródłowym, w celu ułatwienia pisania kodu źródłowego i zwiększenia jego czytelności, jest usuwana lub zastępowana innym kodem. W wyniku tego etapu powstaje plik z kodem źródłowym, zapisany w tym samym języku, ale zdecydowanie prostszy w interpretacji na kolejnych etapach.
2. **Kompilacja**, w czasie której następuje właściwe tłumaczenie programu wejściowego na program zapisany w innym języku. Zazwyczaj językiem wynikowym jest assembler albo język pośredni dla pewnej maszyny wirtualnej. Są to języki, które definiują rozkazy, które mogą być bezpośrednio wykonywane przez maszynę, na której następnie program będzie uruchamiany. Kompilacja zazwyczaj zawiera następujące fazy:
   a) **analiza leksykalna**, która dzieli program wejściowy na ciąg symboli terminalnych,
   b) **analiza składniowa**, w trakcie której sprawdzana jest zgodność programu z językiem formalnym, w którym jest on zapisywany,
   c) **analiza semantyczna** sprawdzająca, czy składnia języka programowania została użyta zgodnie z jej funkcją przewidzianą przez definicję języka,
   d) **generowanie kodu wynikowego**, czyli właściwe tłumaczenie programu na język wynikowy,
   e) **optymalizacja**, która opcjonalnie tak modyfikuje kod wynikowy, aby wykonywał się on szybciej na docelowej maszynie lub zajmował mniej miejsca.

3. **Konsolidacja** (asemblacja), w czasie której można połączyć kilka skompilowanych plików jeżeli program ma budowę modułową. W rezultacie powstaje program zakodowany w kodzie maszynowym, który może być już uruchamiany na maszynie docelowej.
4. **Ładowanie i uruchamianie**, w czasie którego program jest ładowany i uruchamiany w środowisku docelowym.

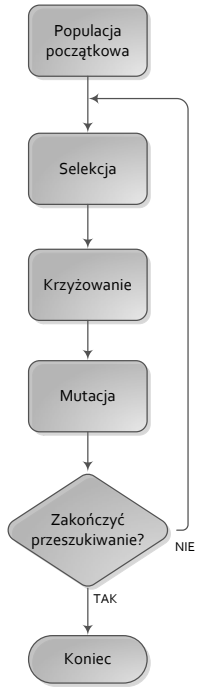
3.5 Algorytmy genetyczne

Algorytmy genetyczne, to algorytmy przeszukiwania przestrzeni rozwiązań, które zostały zainspirowane procesem ewolucji biologicznej. W odróżnieniu od innych prezentowanych w tej pracy algorytmów i metod algorytmy genetyczne nie służą do modelowania procesów dynamicznych, tylko do rozwiązywania statycznych problemów optymalizacyjnych. W tej pracy wykorzystywane są nie do modelowania całych infekcji wirusowych, ale do rozwiązania jednego z podproblemów występujących w modelu wieloagentowym w rozdziale 6.

Precyzyjny opis algorytmów genetycznych oraz ich popularyzacja rozpoczęła się od wydania w 1975 roku książki Johna Hollanda „Adaptation in Natural and Artificial Systems" [Hol75], choć pierwsze prace nad ich zdefiniowaniem powstały już w latach pięćdziesiątych dwudziestego wieku [Bar57]. Algorytmy genetyczne operują na populacji osobników nazywanych fenotypami, które reprezentowane są w pamięci komputera poprzez ciąg zwany genotypem lub chromosomem. Ciąg ten zapisany jest zazwyczaj za pomocą bitów, choć używane są również inne kodowania (na przykład za pomocą liczb rzeczywistych). Osobniki te oceniane są za pomocą funkcji przystosowania (ang. *fitness function*), która jest optymalizowana. Algorytm genetyczny działa iteracyjnie, cyklicznie konstruując kolejne populacje osobników. Algorytm zazwyczaj zaczyna od wygenerowania losowej populacji, następnie generując kolejne o coraz większych wartościach przystosowania za pomocą operacji krzyżowania, mutacji i selekcji.

Szczegółowy schemat działania algorytmu genetycznego prezentowany jest na rysunku 3.2 i składa się z następujących etapów:

1. **Inicjalizacja** - początkowa populacja złożona z kilkudziesięciu lub kilkuset osobników tworzona jest w sposób losowy. Losowe wygenerowanie wielu osobników pozwala na rozpoczęcie poszukiwań w wielu punktach przestrzeni rozwiązań jednocześnie.



Rysunek 3.2: Schemat działania algorytmu genetycznego.

2. **Selekcja** - z aktualnej populacji wybierane są osobniki, które zostaną przekazane do kolejnej generacji osobników. Najczęściej są to najlepiej przystosowane osobniki, bądź osobniki wybrane losowe, przy czym prawdopodobieństwo wyboru jest proporcjonalne do przystosowania osobnika. Losowy wybór daje możliwość wyboru z niskim prawdopodobieństwem osobników, które nie są dobrze przystosowane, ale mają pewne cechy, które umożliwią im wygenerowanie dobrych rozwiązań w przyszłości.
3. **Krzyżowanie** - dwa osobniki wybierane są z wyselekcjonowanej populacji, a następnie na podstawie ich genotypów tworzone jest dziecko, posiadające po fragmencie genotypu każdego rodzica. W ten sposób istnieje szansa, że na podstawie dwóch dobrych osobników powstanie trzeci, który będzie łączył ich najlepsze cechy.
4. **Mutacja** - z niewielkim prawdopodobieństwem fragment chromosomu jest losowo zmieniany o niewielką wartość. Celem mutacji jest uniknięcie utknięcia w optimum lokalnym, poprzez wprowadzanie nowych, losowych cech do populacji.
5. **Zakończenie algorytmu** - istnieje wiele strategii zakończenia obliczeń, najpopularniejszymi są upłynięcie limitu czasu bądź nie uzyskanie polepszenia przystosowania najlepszego osobnika przez określoną liczbę pokoleń.

Przyczyna, dla której algorytmy genetyczne są wydajne i często bardzo dobrze radzą sobie w znajdowaniu rozwiązań optymalnych, może być początkowo trudna do zrozumienia. Jednak istnieje logiczne wytłumaczenie ich poprawności za pomocą hipotezy cegiełek (ang. *building block hypothesis*) [Gol89]. Należy jednak pamiętać, że wydajność algorytmów genetycznych mocno zależy od wybranego sposobu kodowania chromosomów oraz operatorów selekcji, krzyżowania i mutacji. Dlatego ich właściwemu doborowi należy poświęcić dużo uwagi.

3.6 Modele wieloagentowe

W tradycyjnym modelu podejmowania decyzji osoba lub automat podejmujący decyzję, zwany dalej agentem, ma nieograniczone możliwości zasobowe. Brak ograniczeń zasobowych jest szczególnie cenny dla agenta w przypadku czasu. Umożliwia on spokojne rozważenie wszystkich dostępnych możliwości i wybranie optymalnej w danej chwili. Strategia taka dobrze sprawdza się w

systemach, w których faza planowania poprzedza fazę realizacji. Jednak w systemach rzeczywistych dużo częściej występuje sytuacja, w której decyzje trzeba podejmować na bieżąco, reagując dodatkowo na zmiany otoczenia oraz decyzje innych agentów [CDK10, SBZ04].

Próba rozwiązania powyższych problemów doprowadziła do zdefiniowania nowego typu systemu nazwanego systemem wieloagentowym (ang. *multiagent system*, *MAS*) lub systemem opartym na agentach (ang. *agent-based system*). W celu analizy zachowania systemu należy zasymulować jego działanie, a technikę taką określa się terminem symulacji opartej na agentach (ang. *agent-based modeling and simulation*) lub symulacji wieloagentowej (ang. *multiagent simulation*) [ONU+00, KB12]. Cechy charakterystyczne takiej symulacji to [SLB08, Syc98]:

1. Każdy agent działa w pewnym, lokalnym otoczeniu i ma wiedzę o systemie ograniczoną wyłącznie do elementów systemu znajdujących się w pobliżu. Tak więc przy podejmowaniu decyzji nie może kierować się wiedzą o całym systemie, przez co większe jest prawdopodobieństwo podjęcia decyzji nieoptymalnej.
2. Nie ma możliwości globalnej kontroli systemu, czyli nie istnieje jeden agent, który miałby możliwość bezpośredniego oddziaływania na wszystkie pozostałe. Zachowanie systemu jest wynikiem równoległego zachodzenia interakcji lokalnych.
3. Dane są zdecentralizowane i przechowywane w pobliżu poszczególnych agentów lub ich grup, a nie w centralnej bazie danych.
4. Symulacja jest asynchroniczna, aby umożliwić symulowanie zachodzenia interakcji lokalnych w dowolnej kolejności, a nie z góry zadanym porządku.

Powyższa realizacja systemu powoduje, że w systemie wieloagentowym przetwarzanie następuje w kierunku oddolnym. Najpierw symulowane są interakcje lokalne, a dopiero później, jako rezultat ich występowania, zaobserwować można zachowania bardziej globalne [Eps06, Ser97]. Taki kierunek przetwarzania może być bardzo przydatny między innymi w biologii systemowej, która w swoim założeniu modeluje skomplikowane systemy jako sumę interakcji w prostszych podsystemach (patrz sekcja 2.1).

4 Podstawowe modele analizy infekcji wirusowych

4.1 Definiowanie modeli biologicznych

Podstawową kwestią od której należy rozpocząć proces modelowania jest decyzja jak formalnie, za pomocą metod matematycznych lub informatycznych dany model zapisać. Używane w tym celu metody można podzielić na kilka grup. Poniżej znajduje się ich opis wraz z opracowanymi przez autora pracy konkretnymi definicjami dla najbardziej charakterystycznych w każdej grupie metod. Zostały one zademonstrowane na podstawie modelu infekcji wirusem HCV, który został opisany w [RDP09]. Szczegółowe omówienie tego modelu znajduje się w sekcji 4.1.1 opisującej podstawowy sposób definiowania modeli infekcji wirusowych za pomocą równań różniczkowych, a jego analiza przeprowadzona przez autora w sekcji 4.2 poświęconej oprogramowaniu służącemu do analizy zapisanych modelów.

4.1.1 Klasyczna metoda matematyczna

Do modelowania przebiegu infekcji wirusowych modele oparte na równaniach różniczkowych na większą skalę wykorzystywane są od ponad dwudziestu lat, od czasu zdefiniowania pierwszego takiego modelu dla wirusa HIV przez

Alana Perelsona w 1989 roku [Per89]. Obecnie jest to najczęściej wykorzystywana metoda, która opisuje dynamiczny model systemu biologicznego za pomocą równań opisujących zmianę pewnej wartości w czasie [DGPL11, WN02, SH08]. Zazwyczaj wartością tą jest liczba pewnych typów komórek gospodarza oraz liczba wirionów. Przykładowe równania opisujące model infekcji HCV zaprezentowany w [RDP09], pochodzące z tego artykułu, wyglądają następująco:

$$\frac{dT}{dt} = s + r_T T \left(1 - \frac{T+I}{T_{max}}\right) - d_T T - (1-\eta)\beta V T + qI \tag{4.1}$$

$$\frac{dI}{dt} = r_I I \left(1 - \frac{T+I}{T_{max}}\right) + (1-\eta)\beta V T - d_I I - qI \tag{4.2}$$

$$\frac{dV}{dt} = (1-\epsilon)pI - cV \tag{4.3}$$

W powyższych równaniach T oznacza liczbę niezainfekowanych hepatocytów (ang. *Target cells*), I oznacza liczbę zainfekowanych hepatocytów (ang. *Infected cells*), natomiast V oznacza liczbę wirionów (ang. *Virions*). Niezainfekowane hepatocyty są produkowane przez różnicowanie komórek blastycznych w tempie s oraz są infekowane w tempie β proporcjonalnym do ich liczby oraz liczby wirionów. Zarówno niezainfekowane, jak i zainfekowane komórki umierają w tempie odpowiednio d_T oraz d_I i rozmnażają się przez podział z maksymalnym tempem równym r_T oraz r_I tak długo, aż nie zostanie osiągnięta maksymalna dopuszczalne liczba hepatocytów T_{max} modelująca pojemność organizmu. Zainfekowane komórki mogą być również w niewielkim stopniu leczone w tempie q [DMZ+05]. Aktywne wiriony lub wiriony nieaktywne w skutek powstania niszczących je mutacji genetycznych są produkowane z zainfekowanych hepatocytów w tempie p i usuwane przez system immunologiczny w tempie c. Współczynniki ϵ i η modelują proces terapii z użyciem leków przeciwwirusowych (interferonu i rybawiryny) i przy braku terapii równe są 0. Czas jest mierzony w dniach, a liczba komórek w ich liczbie w mililitrze tkanki.

Równania różniczkowe dzięki matematycznej strukturze są w stanie bardzo precyzyjnie opisać modelowany system. Co więcej, jest to dobrze znane narzędzie matematyczne, dla którego istnieje bogaty zestaw metod i algorytmów służących do ich analizy. Wadą metody jest natomiast to, że jest niezmiernie trudna do zrozumienia przez biologów, którzy nie posiadają wymaganych podstaw matematycznych. Aby trochę uprościć zapis i pozbyć się konieczności zapisywania modelu przy pomocy wzorów matematycznych, zaimplementowane zostały środowiska, które umożliwiają skonstruowanie modelu za pomocą graficznych bloków i połączeń między nimi. Przykładowymi programami tego typu są Mathworks Simulink (uzupełniający środowisko Matlab) [KA11] oraz

WinFACT [Ber12]. Rysunek 4.1 przedstawia reprezentację powyższego modelu wykonaną przez autora w programie Simulink, natomiast dodatek A.3 przedstawia inny przykładowy model dla wirusa HIV opisany w [CR00].

Innym sposobem komputerowej reprezentacji równań różniczkowych jest wykorzystanie formatu, który został specjalnie przygotowany w celu zapisu takich modeli. Jednym z nich jest na przykład XMLlab [MP04] bazujący na języku XML. Niestety taki format nie dosyć że nie rozwiązuje problemu konieczności rozumienia działania równań różniczkowych, to jeszcze wprowadza dodatkową trudność związaną z przekształceniem ich postaci matematycznej na język formalny (w tym przypadku XML).

4.1.2 Dedykowane języki programowania

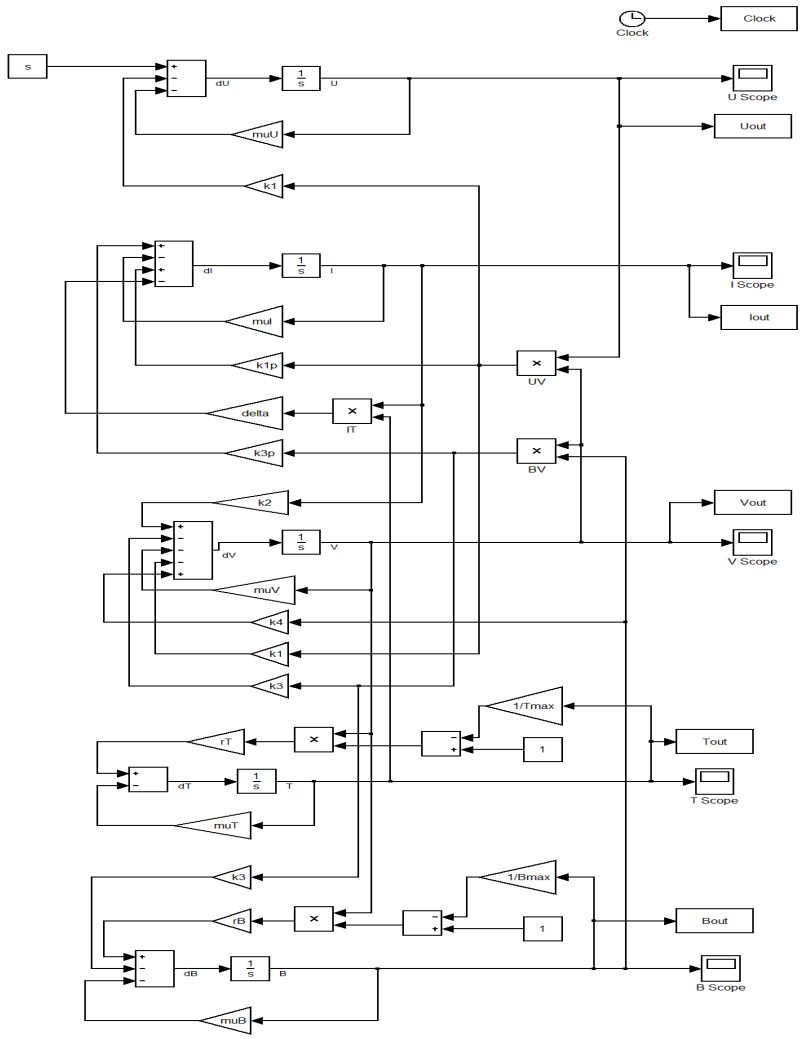
Alternatywną metodą zapisu definicji modeli infekcji wirusowych jest wykorzystanie dedykowanych do tego języków programowania. Najpopularniejszym językiem wykorzystywanym w biologii systemowej, stanowiącym nieformalny standard dla oprogramowania operującego na modelach systemów biologicznych jest język SBML (Systems Biology Markup Language) [FH03, HFS+03]. Jest to język nad którym prace rozpoczęły się w 1999 roku i który oparty został na języku XML. Od tego czasu język jest cały czas rozwijany, a jego najnowszą rewizją jest wersja 3 [HBK+10]. Głównym celem przy projektowaniu języka była standaryzacja sposobu zapisu definicji modeli używanych w biologii systemowej, która miała służyć zdefiniowaniu formatu, który umożliwi:

- pracę nad raz zapisanym modelem w wielu programach potrafiących interpretować język SBML,
- łatwą wymianę modeli pomiędzy naukowcami,
- dalszą pracę i analizowanie modelu, gdy program w którym został on zdefiniowany przestanie być już utrzymywany lub dostępny.

Język SBML odniósł duży sukces, w październiku 2012 oficjalna strona języka zawierała opis 247 programów, które umożliwiają przetwarzanie modeli w języku SBML. Model infekcji HCV opisany w [RDP09] prezentowany jest na listingu 4.1 w skróconej wersji, ze względu na długość jego kodu. Pełna wersja modelu przedstawiona jest w dodatku A.1.

Listing 4.1: Fragment opisu modelu infekcji HCV za pomocą języka SBML przygotowany przez autora pracy.

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <!-- Created by libAntimony version v2.2 on 2012-10-27 13:24 with
      libSBML version 5.6.0. -->
```



Rysunek 4.1: Model infekcji HCV zaprojektowany przez autora pracy w programie Mathworks Simulink uzupełniającym środowisko Matlab.

```
<sbml xmlns="http://www.sbml.org/sbml/level3/version1/core" level="3" version="1">
  <model id="__main" name="__main">
    <listOfCompartments>
      <compartment sboTerm="SBO:0000410" id="default_compartment" spatialDimensions="3" size="1" constant="true"/>
    </listOfCompartments>
    <listOfSpecies>
      <species id="Healthy_hepatocytes" compartment="default_compartment" hasOnlySubstanceUnits="false" boundaryCondition="false" constant="false"/>
      <species id="Infected_hepatocytes" compartment="default_compartment" hasOnlySubstanceUnits="false" boundaryCondition="false" constant="false"/>
      <!-- ... -->
    </listOfSpecies>
    <listOfParameters>
      <parameter id="rHU" constant="false"/>
      <parameter id="rHI" constant="false"/>
      <parameter id="s" constant="false"/>
      <!-- ... -->
    </listOfParameters>
    <listOfReactions>
      <reaction id="HH_Reproduction" reversible="true" fast="false">
        <listOfReactants>
          <speciesReference species="Healthy_hepatocytes" stoichiometry="1" constant="true"/>
        </listOfReactants>
        <listOfProducts>
          <speciesReference species="Healthy_hepatocytes" stoichiometry="1" constant="true"/>
        </listOfProducts>
        <kineticLaw>
          <math xmlns="http://www.w3.org/1998/Math/MathML">
            <ci> rHU </ci>
          </math>
        </kineticLaw>
      </reaction>
      <!-- ... -->
    </listOfReactions>
  </model>
</sbml>
```

Język SBML pomimo wielu zalet i swojej popularności ma dwie wady. Po pierwsze wymaga umiejętności ścisłego i strukturalnego zapisu kodu, a jego interpretacja wymaga interpretacji wielu słów kluczowych (nazw znaczników) zdefiniowanych w specyfikacji języka. Dodatkowo język jako podstawę wykorzystuje standard XML, który jest mocno nadmiarowy, a tworzone w nim opisy długie i rozwlekłe, przez co czytelność opisu modeli jest dodatkowo zmniej-

szona.

Częściowo powyższe wady języka SBML rozwiązuje język Antimony [SBCS09]. Jest to język przy którego tworzeniu głównymi celami były między innymi możliwość łatwego odczytywania i interpretacji kodu przez człowieka oraz łatwego i szybkiego opisywania nowych modeli. Przykładowy model zapisany w języku Antimony przedstawiony jest na listingu 4.2. Zaletą tego języka jest stosunkowo prosta budowa reguł opisujących interakcje w systemie biologicznym oraz jego zwięzłość. Co prawda język opisu reguł cały czas nie przypomina powszechnie wykorzystywanego języka naturalnego, to powinien on być stosunkowo łatwy do zrozumienia. Niestety jego wadami są niemożliwość zdefiniowania przedziału wartości, w którym powinny zawierać się parametry oraz problem z dodawaniem do modelu dodatkowych ograniczeń.

Listing 4.2: Model infekcji HCV zdefiniowany przez autora pracy w języku Antimony.

```
1  //Created by libAntimony v2.2
2  // Compartments and Species:
3
4  species Healthy_hepatocytes, Infected_hepatocytes, Blastic_cells,
       Dead_hepatocytes;
5  species Virions, Dead_virions;
6
7  // Reactions:
8  HH_Reproduction: Healthy_hepatocytes -> Healthy_hepatocytes; rHU;
9  IH_Reproduction: Infected_hepatocytes -> Infected_hepatocytes; rHI
       ;
10 BC_Upgrade: Blastic_cells -> Healthy_hepatocytes; s;
11 HH_Death: Healthy_hepatocytes -> Dead_hepatocytes; dHU;
12 IH_Death: Infected_hepatocytes -> Dead_hepatocytes; dHI;
13 Infection: Healthy_hepatocytes + Virions -> Infected_hepatocytes;
       beta;
14 Virus_emission: Infected_hepatocytes -> Infected_hepatocytes +
       Virions; pH;
15 Virus_death: Virions -> Dead_virions; cv;
16 Cure: Infected_hepatocytes -> Healthy_hepatocytes; cI;
17
18 // Species initializations:
19 Healthy_hepatocytes = ;
20 Infected_hepatocytes = ;
21 Blastic_cells = ;
22 Dead_hepatocytes = ;
23 Virions = ;
24 Dead_virions = ;
25
26 // Variable initializations:
27 rHU = ;
28 rHI = ;
29 s = ;
```

```
30 dHU = ;
31 dHI = ;
32 beta = ;
33 pH = ;
34 cv = ;
35 cI = ;
36
37 //Other declarations:
38 var rHU, rHI, s, dHU, dHI, beta, pH, cv, cI;
```

Ciekawym podejściem nawiązującym do opisu systemów biologicznych za pomocą opisanych powyżej dedykowanych języków formalnych jest podejście oparte na regułach (ang. *rule based*) [HFB+06]. W podejściu tym procesy zachodzące w systemie opisane są za pomocą reguł zbliżonych strukturalnie do reakcji chemicznych. Każda reguła składa się z dwóch części - definicji substratów, które w trakcie zachodzenia procesu przemieniają się w produkty. W szczególności proces może być dwukierunkowy, czyli możliwa jest również ponowna przemiana otrzymanych produktów w elementy, które były substratami.

Jednym z popularniejszych sposobów zapisu reguł w biologii systemowej jest język BioNetGen [FBH09]. Język ten definiuje formalny język programowania, za pomocą którego reguły mogą być zapisywane i przetwarzane przez narzędzia informatyczne. Sposób zapisu reguł zostanie natomiast przedstawiony za pomocą notacji ML-Rules [MRU11], która w odróżnieniu od BioNetGen po pierwsze umożliwia łatwe modelowanie zależności hierarchicznych (wielopoziomowych), a po drugie definiuje wygodną w użyciu notację matematyczną używaną do zapisu reguł. Model infekcji HCV zdefiniowany przez autora pracy z wykorzystaniem ML-Rules przedstawiony jest na poniższych równaniach:

$$\oslash \xrightarrow{s} T \tag{4.4}$$

$$T^t + 0\,I^i \xrightarrow[t+i<T_{max}]{r_T} 2\,T \tag{4.5}$$

$$0\,T^t + I^i \xrightarrow[t+i<T_{max}]{r_I} 2\,I \tag{4.6}$$

$$I \xrightarrow{q} T \tag{4.7}$$

$$V + T \xrightarrow{\beta} I \tag{4.8}$$

$$I \xrightarrow{p} I + V \tag{4.9}$$

$$T \xrightarrow{d_T} \oslash \tag{4.10}$$

$$I \xrightarrow{d_I} \oslash \tag{4.11}$$

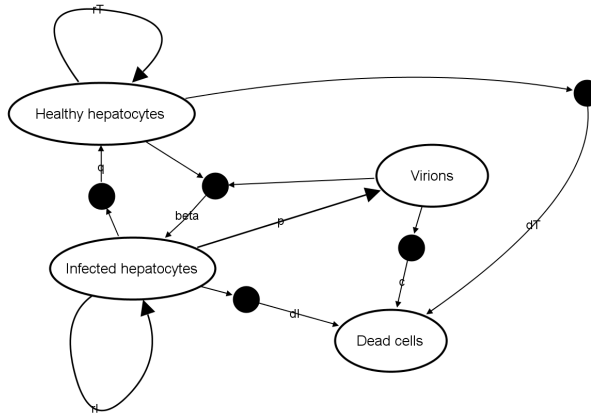
$$V \xrightarrow{c} \oslash \tag{4.12}$$

W powyższym przykładzie wielkie litery oznaczają elementy biorące udział w procesach, małe litery ich liczność, strzałki oznaczają produkcje, współczynniki nad strzałkami oznaczają tempo zachodzenia procesu, natomiast warunki pod strzałkami muszą być spełnione, aby proces zachodził. Modelowanie za pomocą reguł jest z pewnością ciekawym podejściem ze względu na swoją zwięzłość i bezpośrednie przełożenie zachodzących procesów na konkretne reguły. Z drugiej strony zrozumienie notacji wymaga precyzyjnego zrozumienia zasad definiowania reguł. Dla bardziej skomplikowanych modeli, wykorzystujących w pełni możliwości języka takie jak atrybuty i modelowanie hierarchiczne, zapis może wymagać dłuższej analizy w celu jego zrozumienia i może być trudny do przyswojenia przez osoby bez doświadczenia w interpretowaniu skomplikowanych definicji matematycznych. Trzeba również pamiętać, że aby automatycznie przeanalizować taki model na komputerze należy najpierw wprowadzić go za pomocą języka takiego jak na przykład BioNetGen, który tak samo jak wcześniej opisane języki programowania łatwy w użyciu będzie tylko dla programistów.

4.1.3 Graficzna reprezentacja modeli

W celu uproszczenia procesu rozumienia opisu modelu przez człowieka zaprojektowana została graficzna notacja SBGN (Systems Biology Graphical Notation) [LHM+09]. Prace nad nią rozpoczęły się w 2006 roku i dotychczas zdefiniowane zostały trzy zestawy oznaczeń graficznych do modelowania interakcji (Process Description), relacji pomiędzy bytami (Entity Relationship) oraz przepływu informacji (Activity Flow). Zgodnie z założeniami notacja prezentuje model w czytelny, graficzny sposób, co zdecydowanie ułatwia przekaz. Niestety, mimo że notacja ta wspiera import z języka SBML, została ona zaprojektowana głównie w celu reprezentacji sieci metabolicznych, ekspresji genów oraz przekazywania sygnałów. Przez to nie najlepiej nadaje się do reprezentacji infekcji wirusowych. Rysunek 4.2 przedstawia reprezentację modelu infekcji HCV w notacji SBGN na tyle dokładnie, na ile notacja to umożliwia. Głównymi problemami jest brak elementów na oznaczenie komórek, niemożliwość prostego zamodelowania ich śmierci oraz problem z podaniem zakresu parametrów.

Aby umożliwić łatwą edycję modeli reprezentowanych w notacji SBGN zapisywane są one nie w plikach graficznych, ale w dedykowanym formacie SBGN-ML (SBGN Markup Language) opartym na języku XML. Niestety liczba programów, które potrafią go interpretować jest zdecydowanie bardziej ograniczona niż w przypadku języka SBML. Zapis modelu przedstawionego na rysunku 4.2 w formacie SBGN-ML znajduje się w dodatku A.2.



Rysunek 4.2: Graficzna reprezentacja modelu infekcji HCV w notacji SBGN. Grafika została utworzona przez autora pracy w programie VANTED [JKS06] z użyciem wtyczki SBGN-ED [CKS10].

Oprócz notacji SBGN istnieje jeszcze wiele środowisk symulacyjnych, w których wprowadzanie modelu do symulacji następuje lub jest wspomagane za pomocą narzędzi graficznych. Jednak w ich przypadku format zapisu opisów modeli jest dedykowany dla tej konkretnej aplikacji i nie jest tak uniwersalny jak SBGN. Dlatego też prezentacja tego sposobu opisu modeli znajduje się w sekcji 4.2 wraz z opisem aplikacji symulacyjnych.

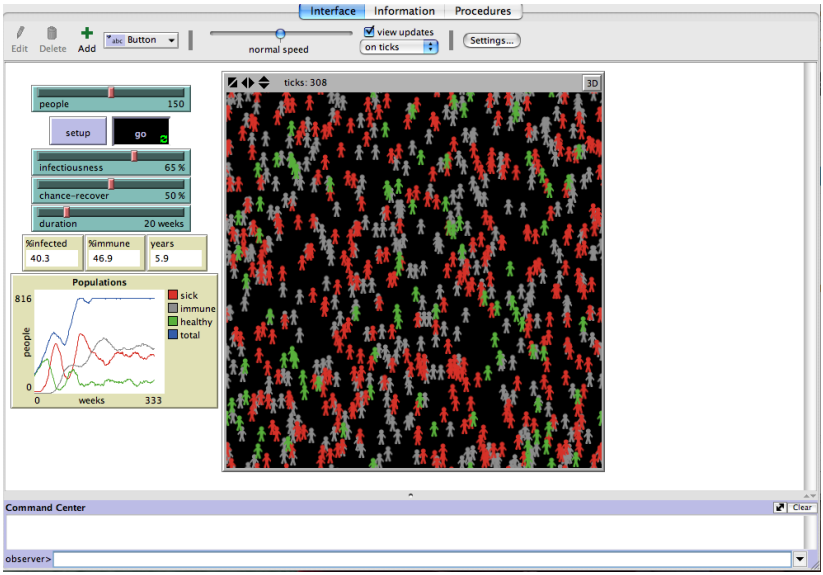
4.2 Komputerowa analiza modeli

Zdefiniowanie modelu infekcji wirusowej za pomocą jednej z metod przedstawionych w sekcji 4.1 jest tylko niezbędnym etapem wstępnym, po którym następuje właściwa analiza modelu za pomocą metod matematycznych lub informatycznych. Sekcja ta zawiera przegląd dostępnych metod analizy modeli infekcji wirusowych (patrz również obszerny przegląd dostępnych metod w [MAW12]) wraz z wynikami przeprowadzonej przez autora szczegółowej analizy klasycznego modelu infekcji HCV [RDP09].

4.2.1 Oprogramowanie do symulacji komputerowych

Istnieje wiele programów komputerowych służących do symulowania i analizowania równań różniczkowych. Przykładowo w pracy tej używany był do tego program Mathworks Simulink opisany w sekcji 4.1.1. Trochę bardziej ograniczony wybór występuje w przypadku, gdy oprogramowanie oprócz numerycznej symulacji równań powinno również wyznaczać wartości parametrów występujących w równaniach na podstawie danych eksperymentalnych. Jednak w tym przypadku również dostępnych jest kilka sprawnych programów takich jak darmowy w zastosowaniach akademickich PottersWheel [MT08], Berkeley Madonna [MOZ09] lub Easy-Fit [Sch02] obsługujący oprócz równań zwyczajnych również równania cząstkowe, a także popularny arkusz kalkulacyjny Microsoft Excel [CCHK07].

Jak zaprezentowano powyżej obszar programów wspomagających analizowanie układów równań różniczkowych jest bardzo dobrze zagospodarowany. Odmiennie wygląda obszar symulacji wieloagentowych systemów biologicznych, które są obszernie omawiane w tej pracy. Dotychczas stworzono kilka środowisk, które w znacznym stopniu mogą wspomóc projektowanie modeli opartych na agentach. Są to środowiska takie jak Brahms[SCvH03], AndroMeta [And12] i NetLogo [Wil99]. Niestety żadne z nich nie nadaje się do przeprowadzania bardziej skomplikowanych symulacji. Główną brakującą w nich funkcjonalnością



Rysunek 4.3: Przykładowy model stworzony w programie NetLogo - rozprzestrzenianie się wirusa HIV w populacji ludzkiej. [Źródło: Stanford University - Computational Modeling and Analytics in Social Sciencehttp://computationalmodelingblogs.stanford.edu/winter2012/2012/01/18/aids-who-makes-the-decision/]

jest możliwość automatycznego dostrojenia wartości parametrów występujących w modelu do danych eksperymentalnych.

Podjętych zostało również kilka prób własnej implementacji symulacji wieloagentowych w celach wirusologicznych, a także wykorzystania w tym celu któregoś z gotowych, powyższych środowisk. Jednak próby te są nieliczne i zazwyczaj ograniczone do wirusa HIV [ZKC05, MLF+08, IKI+10], a tylko pojedyncze prace zakładają, że opracowane modele mogą być stosowane dla innych wirusów, na przykład HCV [IKI+10]. Co więcej większość z istniejących prac prezentuje tylko możliwość wykorzystania symulacji wieloagentowej do symulowania przebiegu infekcji wirusowej, a nie pokazuje metody, która mogłaby służyć wyciągnięciu biologicznych wniosków. Natomiast jak zauważono w [MAW12] wartość modelu nie powinna być oceniana po tym jak dokładny, albo złożony on jest, ale po tym czego można się z niego nauczyć. Jedynie [ZKC05] prezentuje jak opracowane modele mogą być wykorzystane do zweryfikowania kilku powszechnie panujących hipotez o infekcji HIV. Żadna ze wspomnianych

prac nie odpowiada również na pytanie jak inaczej niż za pomocą prób i błędów można dobrać wartości parametrów modelu dla konkretnego przypadku medycznego.

Istnieje również grupa dedykowanych języków programowania, które stworzone zostały specjalnie w celu przeprowadzania różnorodnych, nie tylko biologicznych symulacji. Do najpopularniejszych z nich należą oparty na Javie JiST [BHvR05] oraz oparty na Pythonie SimPy [MV03]. Języki te niestety są trudne w obsłudze przez biologów, gdyż tak jak sposoby opisu modeli zaprezentowane w sekcji 4.1.2 wymagają najpierw, aby zaimplementować definicję modelu w wybranym języku formalnym. Dodatkowo są one bardzo ogólne i nie zawsze zastosowanie ich do symulacji infekcji wirusowej jest łatwe. Druga z tych wad w zdecydowanie mniejszym, choć ciągle zauważalnym stopniu dotyczy środowisk MSI [MLF+08] oraz CAFISS [TJ05], które zostały stworzone z myślą o symulacji działania układu immunologicznego. Wykorzystanie ich wiąże się jednak również z nauką dedykowanego języka opisu modeli.

4.2.2 Analiza równań różniczkowych

Model zaprezentowany w sekcji 4.1.1 może być analizowany analitycznie lub za pomocą jednego z wielu programów komputerowych. Szczegółowa analiza analityczna tego modelu została przeprowadzona w [RDP09]. Wynika z niej, że aby pacjent wyzdrowiał spełniony musi być warunek:

$$\frac{p\beta T_{max}(r_T - d_T)}{c(d_I r_T - d_T r_I)} > 1 \tag{4.13}$$

oraz istnieją dwa stany stabilne - zainfekowany o niezerowych liczbach komórek $(T'; I'; V')$ oraz wyleczony:

$$\left(\frac{r_U - d_U}{r_U} \cdot U_{max}; 0; 0\right). \tag{4.14}$$

Na rysunku 4.4 przedstawiono natomiast wyniki symulacji tego modelu przeprowadzone przez autora pracy w programie Simulink za pomocą modelu opisanego w sekcji 4.1.1. Wyniki przeprowadzonej symulacji są zgodne z wynikami prezentowanymi przez twórców modelu [RDP09].

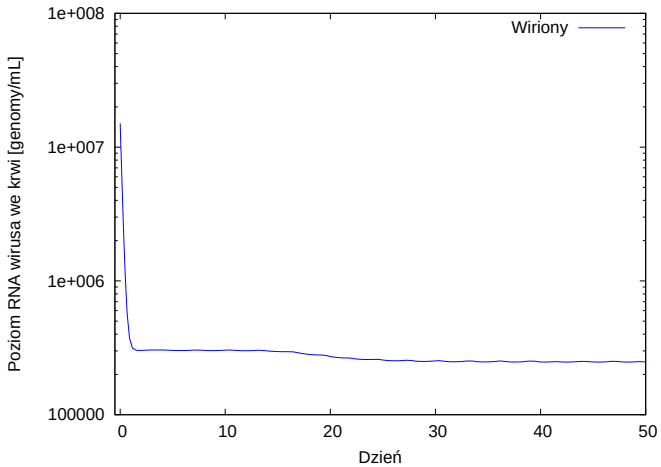


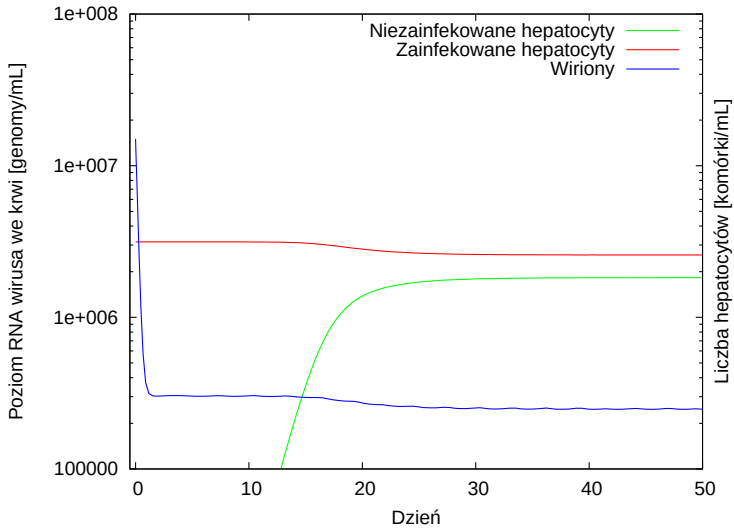
Figure 4.4: Wykonana przez autora symulacja w środowisku Simulink modelu opisanego w [RDP09], której wyniki pokrywają się z wynikami otrzymanymi przez autorów pracy źródłowej.

Dane pozwalające zdiagnozować stan chorego zakażonego wirusem HCV są zazwyczaj ograniczone do poziomu RNA wirusa we krwi (patrz sekcja 2.3.2). Z tego powodu autorzy modeli infekcji HCV zazwyczaj koncentrują się tylko na równaniu dotyczącym liczby wirionów i tylko na jego podstawie starają się dopasować model do danych. Może to prowadzić do wielu błędnych wniosków, ponieważ niemożliwe jest udowodnienie, że model jest poprawny, jeżeli równania modelujące liczbę hepatocytów nie prezentują choćby prawdopodobnych wyników. Mimo to w większości modeli zakłada się, że w czasie pierwszych dni terapii liczba hepatocytów jest stała, a fakt ten nie jest w żaden sposób weryfikowany. W ten sposób pomijany jest fakt, że nawet jeżeli równanie prezentujące poziom RNA pasuje idealnie do danych klinicznych, to dopóki równania opisujące poziom hepatocytów również nie generują prawidłowych wyników, model nie może być uznany za model poprawnie wyjaśniający cały proces infekcji wirusem.

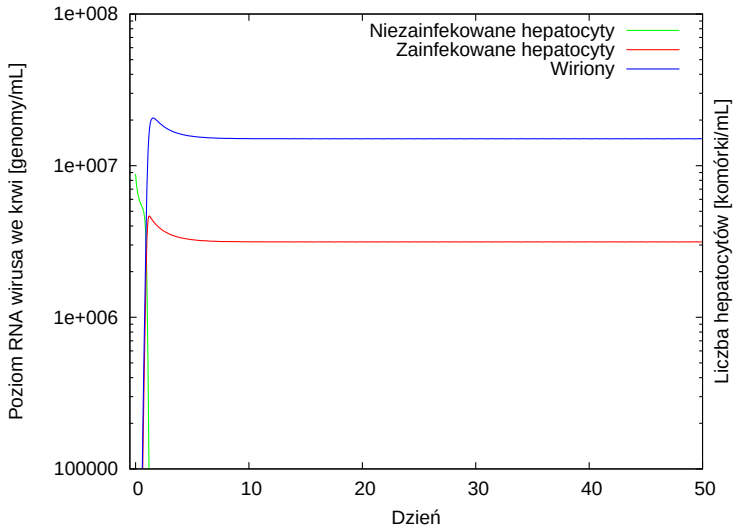
Jako przykład przedstawiona zostanie analiza klasycznego modelu, opisanego wcześnie, a zdefiniowanego w [RDP09]. Wyniki tego modelu dla równania opisującego poziom RNA wirusa, które prezentowane są w artykule źródłowym, przedstawione zostały wcześniej na rysunku 4.4. Rysunek 4.5a przedstawia te same wyniki, ale z uwzględnieniem równań prezentujących liczbę zainfekowanych i niezainfekowanych hepatocytów. Na wykresie można zauważyć, że całkowita liczba hepatocytów po 50 dniach od rozpoczęcia terapii spada do $4{,}4 \cdot 10^6$

komórek na mililitr. Jest to 12% mniej niż liczba komórek w zdrowej wątrobie, co modelowane jest przez ich liczbę w stanie stabilnym dla zdrowego pacjenta i równe jest $4{,}98 \cdot 10^6$ komórek na mililitr. Tak szybki i znaczący spadek liczby komórek jest wysoko nieprawdopodobny.

Jeszcze większy problem pojawia się podczas analizowania przebiegu infekcji przed rozpoczęciem terapii. Model zakłada, że parametry ϵ oraz η poprawnie modelują proces terapii, a pozostałe parametry opisują resztę procesów występujących w ciele pacjenta i nie mają żadnego wpływu na modelowanie terapii. Zgodnie z tym założeniem ustawienie wartości parametrów ϵ oraz η na 0 powinno modelować przebieg infekcji u pacjenta przed rozpoczęciem terapii. Przypadek taki prezentowany jest na rysunku 4.5b. Z wykresu można odczytać, że całkowita liczba hepatocytów spada do $3{,}14 \cdot 10^6$ komórek na mililitr po zaledwie 10 dniach infekcji. Oznacza to, że objętość wątroby maleje o 40% w 10 dni od momentu zakażenia, co w rzeczywistości jest niemożliwe. Co więcej już po 3 dniach liczba niezainfekowanych komórek maleje do $3 \cdot 10^3$ komórek na mililitr. Oznacza to, że momentalnie większość komórek zostaje zainfekowana, co jak wykazano w [LSX$^+$09] nie jest możliwe.



(a) Po rozpoczęciu terapii.



(b) Przed rozpoczęciem terapii.

Rysunek 4.5: Symulacja dynamiki infekcji wirusowej oraz liczby komórek zainfekowanych i niezainfekowanych po rozpoczęciu terapii (a) oraz przed rozpoczęciem terapii (η,$\epsilon = 0$) (b).

Analiza zaprezentowana w tej sekcji pokazuje, że nawet wykorzystanie dobrze znanych i powszechnie wykorzystywanych metod oraz narzędzi może doprowadzić do pominięcia kluczowych etapów analizy modelu. W dalszej części pracy, w rozdziale 6 prezentowana jest bardziej wszechstronna metoda symulowania modeli infekcji wirusowych oraz jej wykorzystanie do głębszego zanalizowania problemów występujących w powyższym modelu.

5
Formalny opis modeli dynamicznych

Jak zauważono we wstępie do tej pracy modelowanie matematyczne pełni ważną rolę w biologii. Pozwala analizować skomplikowane procesy biologiczne i wyciągać wartościowe wnioski. Niestety narzędzia służące do modelowania są zazwyczaj niezrozumiałe dla biologów, a możliwość pomocy oferowana im przez informatyków i matematyków jest mocno utrudniana poprzez brak u nich wiedzy dziedzinowej. Problem ten jest w środowisku biologicznym znany już od wielu lat. Na przykład w roku 2002 Yuri Lazabnik w tekście „Czy biolog może naprawić radio?" (ang. *Can a biologist fix a radio?*) [Laz02] zauważa, że w tamtych czasach biologom brakowało umiejętności modelowania systemów biologicznych, ale:

> *„Rozwój biologii systemowej może zmusić biologów do szybkich zmian swojego podejścia."*

Niestety w roku 2011, polemizując z powyższym tekstem, James Faeder zauważa [Fae11], że:

> *„Prawie dekadę później ciągle można bezpiecznie założyć, że większość biologów eksperymentalnych prawdopodobnie nie posiada żadnych umiejętności modelowania oraz wykorzystania oprogramowania modelującego."*

Wyniki przedstawione w tym rozdziale mogą pomóc zmienić powyższy, niekorzystny stan rzeczy. Jak przedstawiono w rozdziale 4.1 istnieje wiele spo-

sobów formalnego zapisu modeli biologicznych, jednak każdy z nich posiada pewne ograniczenia. Celem prac opisanych w tym rozdziale było zdefiniowanie nowego sposobu opisu modeli dynamicznych stosowanych w biologii, między innym do modelowania infekcji wirusowych. W rezultacie powstał język ModeLang, wraz z przykładową implementacją jego parsera. Język ten rozwiązuje problemy istniejące w aktualnie stosowanych podejściach, zachowując przy tym ich funkcjonalność. Język ModeLang został przetestowany w oparciu o modele infekcji HCV i HIV, które zostały w nim zapisane. Aby dotrzeć do szerokiego grona odbiorców podjęto decyzję, aby tworzony język oprzeć o język angielski. Dlatego też wszystkie przykłady reguł i definicje znajdujące się w tym rozdziale zapisane są w tym języku.

Prace opisane w tym rozdziale prowadzone były w większości wspólnie z Tomaszem Prejzendancem, pod kierownictwem promotora i w całości są wynikiem prowadzonych przez autorów badań zgłoszonych do publikacji w [WPB12] i prezentowanych na konferencji międzynarodowej ICOLE'12 w Austrii. Większość przedstawionych w tym rozdziale idei i pomysłów pochodzi od autora tej rozprawy i została sformułowana na podstawie jego doświadczeń z modelowaniem infekcji wirusowych, jednak niektóre idee i większość implementacji parsera języka ModeLang wsparte zostały przez współautora.

5.1 Założenia wstępne i cel prac

Projektując język koncentrowano się głównie na jego przydatności do modelowania biologicznych systemów dynamicznych, a w szczególności infekcji wirusowych, jednak jak wykazano dalej jego przydatność jest dużo szersza i nie ogranicza się tylko do biologii. Głównym celem projektowym było, aby język był łatwy w użytkowaniu dla biologów, którzy nie posiadają obszernej wiedzy matematycznej i informatycznej. Realizacji powyższego celu posłużyło zdefiniowanie następujących celów szczegółowych:

1. Projektowany język powinien być z punktu widzenia użytkownika jak najmniej formalny, czyli umożliwiać stosowanie jak najbardziej elastycznej składni, zbliżonej do składni języka angielskiego. Powinna istnieć możliwość opisywania występujących w nim deklaracji za pomocą różnych synonimów oraz konstrukcji gramatycznych, tak aby jak najlepiej imitować język angielski. W szczególności należy umożliwić zapis reguł zarówno w stronie czynnej, jak i biernej, gdyż obie konstrukcje są powszechnie wykorzystywane przy opisywaniu systemów biologicznych.

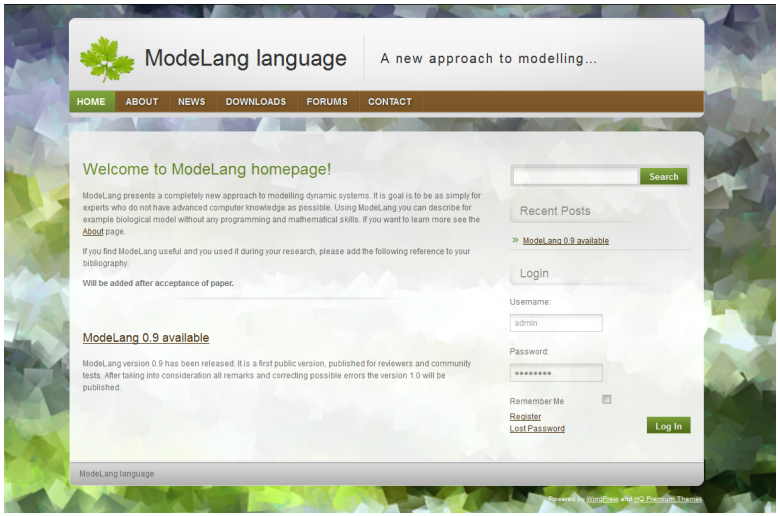
2. Słowa, których nie uda się zinterpretować jako słów kluczowych należy pominąć, gdyż prawdopodobnie są to tylko wyrazy pomocnicze, nie wnoszące żadnej wartościowej informacji. Twórca modelu powinien być oczywiście o tym fakcie powiadomiony, aby móc skorygować ewentualne błędy.
3. Nazwy własne obiektów występujących w modelu powinny być automatycznie wykrywane, aby nie wymuszać na twórcy opisu konieczności definiowania ich formalnie z użyciem specjalnej składni.
4. Jedyna matematyczna część, której nie da się uniknąć to definicja wartości parametrów występujących w modelu, takich jak na przykład prędkość namnażania się komórek. Definicja taka może określać konkretną wartość parametru lub przedział, w którym musi się on znajdować. Aby definicja parametrów była jak najprostsza, należy umożliwić nadawanie parametrom złożonych nazw oraz zapis ich wartości w najbardziej intuicyjny sposób – za pomocą znaków równości, mniejszości i większości.

5.2 ModeLang - nowy język opisu modeli biologicznych

W celu realizacji celów postawionych przed nowym sposobem definiowania modeli systemów biologicznych zdefiniowany został nowy język nazwany ModeLang. W celu dostarczenia wsparcia jego użytkownikom oraz zbierania opinii społeczności na temat języka, utworzona została dla niego dedykowana strona www (http://modelang.cs.put.poznan.pl/). Poniżej znajduje się szczegółowy opis języka.

5.2.1 Sposób wykorzystania języka

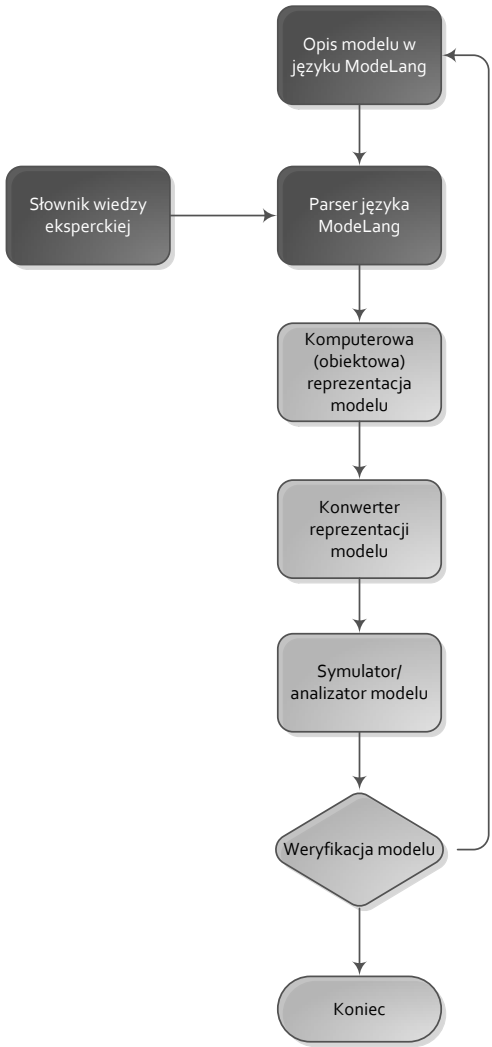
Z formalnego punktu widzenia wykorzystanie języka ModeLang nie odbiega znacząco od sposobu wykorzystania innych języków formalnych wykorzystywanych w informatyce, w tym języków programowania. Jedyną różnicą jest składnia, która jest dużo bardziej elastyczna niż w większości istniejących języków. Aby w przejrzysty sposób ukazać, w jaki sposób w procesie modelowania infekcji wirusowej wykorzystywany jest język ModeLang na rysunku 5.2 przedstawiono schemat takiego procesu. Etapy na których wykorzystywany jest język



Rysunek 5.1: Zrzut strony domowej języka ModeLang.

ModeLang zaznaczone zostały ciemniejszym kolorem. W dalszej części tej sekcji znajduje się ich szczegółowy opis wraz z odniesieniem do ich odpowiedników w procesie kompilacji większości języków programowania (patrz sekcja 3.4).

1. Pierwszym etapem jest opisanie infekcji wirusowej w języku ModeLang. Na tej podstawie powstaje definicja modelu opisującego tą infekcję, którą formalnie można traktować jak plik z kodem źródłowy.
2. Ponieważ język ModeLang jest językiem zbliżonym do języka naturalnego, jego parser nie jest w stanie samodzielnie zinterpretować opisu modelu. W tym celu potrzebna jest dodatkowa baza wiedzy, która zawiera wpisy definiujące jak konkretne wyrazy z języka naturalnego wykorzystywane są do opisywania infekcji. Baza ta nazywana słownikiem wiedzy eksperckiej wykorzystywana jest na etapie parsowania opisu infekcji zdefiniowanego w punkcie pierwszym. Taki słownik to odpowiednik standardowych bibliotek wykorzystywanych przez wiele języków programowania (na przykład biblioteka *System* w języku Pascal lub *stdlib* w języku C).
3. Wynikiem procesu parsowania jest komputerowa reprezentacja modelu. Reprezentacja ta zapisana jest w pamięci w postaci obiektowej i może być łatwo wykorzystana w dalszych etapach przez oprogramowanie, dla którego opis modelu jest wejściem. W rezultacie wyjście parsera języka



Rysunek 5.2: Kolejne etapy modelowania z wykorzystaniem języka ModeLang. Kolor czarny oznacza etapy, na których bezpośrednio wykorzystywany jest język ModeLang.

ModeLang odpowiada wyjściu kompilatorów języków programowania.

4. Aby struktura danych wygenerowana przez parser była w praktyce użyteczna należy przekonwertować ją na format danych wejściowych oprogramowania, które jako wejście pobiera opis modelu infekcji wirusowej, a na wyjściu zwraca obliczone na jego podstawie dane (np. prognozę sposobu przebiegu infekcji wirusowej). Ten etap odpowiada procesowi konsolidacji. Jedyna różnic polega na tym, że w wyniku nie powstaje kod wykonywany przez procesor lub maszynę wirtualną, ale stanowiący wejście do innego programu symulacyjnego.
5. Utworzone dane są wykorzystywane w oprogramowaniu symulującym bądź analizującym model, co odpowiada procesowi uruchomienia programu.
6. Jeżeli wyniki nie są satysfakcjonujące, to opis modelu w języku ModeLang może być poprawiony, a cały proces powtórzony.

5.2.2 Terminologia

Przy definiowaniu i opisywaniu języka ModeLang stosowane są następujące terminy:

- **Agent** to pojedynczy obiekt (byt), którego zachowanie jest modelowane. W przypadku infekcji wirusowej może to być na przykład zdrowa lub zainfekowana komórka, albo wirion.
- **Interakcja** definiuje sposób w jaki agenty mogą na siebie oddziaływać. Przede wszystkim opisuje ile różnych typów agentów uczestniczy w pewnym procesie, które z nich mają wpływ na które oraz jakie jest tempo zachodzenia interakcji.
- **Reguła** określa konkretne zachowanie agentów w systemie biologicznym. Reguły tworzone są na podstawie interakcji. Na podstawie jednej interakcji może być utworzonych wiele reguł, ponieważ interakcja pomiędzy tymi samymi typami agentów może mieć różne przyczyny biologiczne. W przypadku modelu infekcji wirusowej może to być na przykład infekcja, w trakcie której wirion i zdrowa komórka łączą się, tworząc komórkę zainfekowaną.
- **Parametr** definiuje pewną stałą wartość, którą może być na przykład częstotliwość zachodzenia pewnego procesu opisanego za pomocą reguły.

5.2.3 Składnia

Definicja składni języka w notacji bazującej na notacji EBNF i wyrażeniach regularnych przedstawiona została na listingu 5.1.

Listing 5.1: Składnia języka ModeLang

```
value ::= [0-9]+ ('.' [0-9]+)? ([eE] [0-9]+)?
parameter_name ::= [a-zA-Z][a-zA-Z0-9]*
agent ::= [a-zA-Z\ ]+
parameter ::= parameter_name ('=' | '<' | '>') value
            | value '<' parameter_name '<' value
            | value '>' parameter_name '>' value
interaction1 ::= agent (VERB | ADJECTIVE | VERB ADJECTIVE)
    PREPOSITION? agent CONSTRAINT parameter_name
interaction2 ::= agent (VERB | ADJECTIVE | VERB ADJECTIVE)
    CONSTRAINT parameter_name
interaction3 ::= agent 'and' agent VERB PREPOSITION? agent
    CONSTRAINT parameter_name
interaction4 ::= NOUN agent ('and' agent)* VERB CONSTRAINT
    parameter_name
```

Za pomocą wielkich liter zapisano symbole terminalne, które muszą być zdefiniowane w słowniku wiedzy eksperckiej, ponieważ reprezentują informację o wiedzy dziedzinowej. Szczegóły przedstawione są w rozdziale 5.2.6 dotyczącym słowników wiedzy eksperckiej, natomiast przykładowe wartości tych symboli podane są w rozdziale 5.2.4 zawierającym omówienie konkretnych reguł. Znaczenie poszczególnych symboli nieterminalnych jest następujące:

- **value** to dowolna, nieujemna liczba rzeczywista, którą można zaprezentować w typie prostym long double,
- **parameter_name** to nazwa parametru, która możne być dowolnym ciągiem liter i cyfr rozpoczynającym się literą,
- **agent** to nazwa typu agentów, która w odróżnieniu od nazw parametrów nie może zawierać cyfr, ale może zawierać spacje,
- **parameter** to definicja wartości parametru, która może określać jego konkretną wartość lub przedział wartości, do którego musi należeć,
- **interaction1** to interakcja w ramach której jeden agent oddziałuje na innego,
- **interaction2** to akcja, która angażuje tylko jednego agenta,
- **interaction3** to interakcja dwóch agentów z trzecim,
- **interaction4** to ograniczenie dotyczące pewnego zbioru typów agentów.

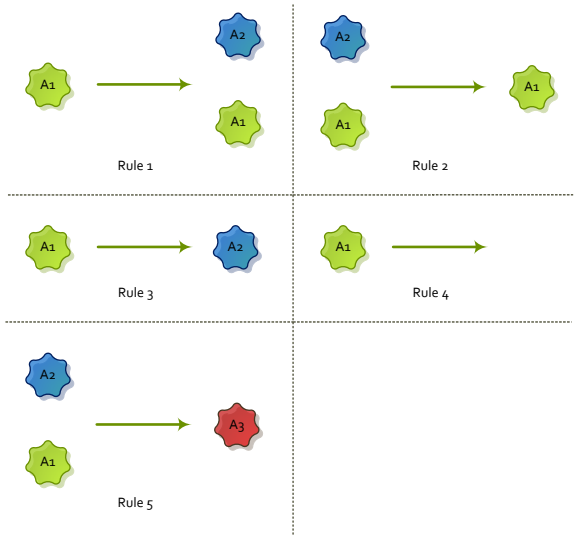
Należy w tym miejscu zaznaczyć, że ponieważ jednym z celów zdefiniowanych w rozdziale 5.1 jest łatwość użycia języka, powyższa składnia traktowana jest przez parser w sposób mało restrykcyjny. Oznacza to, że parser dopuszcza pojawienie się dodatkowych słów i znaków, które nie są w powyższej składni uwzględnione, tak długo dopóki po ich usunięciu konkretne zdanie da się dopasować do dokładnie jednego typu interakcji. Jeżeli poprzez wprowadzenie dodatkowych słów zdanie staje się niejednoznaczne lub niezrozumiałe i niemożliwe do dopasowania do jakiejkolwiek reguły, zostanie zasygnalizowany błąd.

Elastyczność języka osiągnięto również za pomocą dwóch innych zabiegów. Po pierwsze słowem kluczowym pochodzącym ze słownika wiedzy eksperckiej w przypadku interakcji 1 i 2 może być czasownik lub przymiotnik. Umożliwia to zapisanie reguły zarówno jako przykładowo *„Agent **dies**”*, jak i *„Agent become **inactive**”*, o ile oba słowa *„dies”* oraz *„inactive”* znajdą się w słowniku wiedzy eksperckiej. Po drugie powyższa definicja umożliwia zarówno użycie strony czynnej, jak i biernej. W przypadku strony biernej używana jest konstrukcja VERB ADJECTIVE, w której VERB oznacza odpowiednią odmianę czasownika *„to be”*, natomiast ADJECTIVE pełni rolę trzeciej formy czasownika. Przy definiowaniu języka nie wprowadzono rozróżnienia pomiędzy trzecią formę czasownika oraz przymiotnikiem, gdyż często można stosować je zamiennie. Dlatego też łatwiej definiować je w słowniku wiedzy eksperckiej razem, w jednej grupie oznaczanej jako ADJECTIVE.

5.2.4 Reguły

Reguły używane do opisywania infekcji wirusowych zdefiniowane zostały na podstawie interakcji występujących w składni języka opisanej w sekcji 5.2.3. Definiowanie odrębnych reguł było konieczne, ponieważ na podstawie niektórych typów interakcji można zdefiniować kilka reguł o odmiennym znaczeniu biologicznym (posiadających odmienną semantykę). Na przykład obie reguły *„Agent 1 creates Agent 2”* oraz *„Agent 1 destroys Agent 2”* wykorzystują interakcję pierwszą, ale skutek ich wykonania jest całkowicie przeciwny.

Na rysunku 5.3 zaprezentowano wizualnie zasadę działania pierwszych pięciu reguł. Agenty przedstawione z lewej strony strzałki, to obiekty inicjujące daną interakcję, dalej zwane będą agentami **źródłowymi**. Przedstawione z prawej strony strzałek, powstające w w wyniku zajścia interakcji agenty zwane będą dalej **docelowymi**. Na rysunku brakuje prezentacji reguły szóstej, która nie modeluje żadnej akcji tylko modeluje ograniczenia i w związku z tym nie może być przedstawiona graficznie. Poniżej znajduje się dokładniejszy opis reguł. Dla reguł podano również odpowiadający im składnik w przypadku opisywania systemu biologicznego za pomocą równań różniczkowych. W poniższych



Rysunek 5.3: Wizualizacja reguł opisujących infekcję wirusową.

wzorach p oznacza parametr definiujący częstotliwość zachodzenia danej interakcji.

Reguła 1 - Tworzenie

Reguła ta opisuje utworzenie nowego agenta docelowego B przez pewnego agenta źródłowego A, który pozostaje w skutek zajścia interakcji niezmieniony. Reguła ta wykorzystuje interakcję 1 opisaną składnią języka ModeLang. Na przykład nowa komórka jest tworzona przez podział. W równaniach różniczkowych zapisywana jest jako składnik:

$$\frac{dB}{dt} = p \cdot A \tag{5.1}$$

Reguła 2 - Niszczenie

Reguła ta opisuje niszczenie pewnego agenta docelowego B przez agenta źródłowego A. Na przykład komórka układu immunologicznego niszczy pato-

gen. Reguła ta wykorzystuje interakcję 1 opisaną składnią języka ModeLang. W równaniach różniczkowych zapisywana jest jako składnik:

$$\frac{dB}{dt} = -p \cdot A \tag{5.2}$$

Reguła 3 - Tranzycja

Reguła ta opisuje sytuację, gdy agent źródłowy A przekształca się w pewnego agenta B. Na przykład nowe komórki powstają z komórek macierzystych. Reguła ta wykorzystuje interakcję 1 opisaną składnią języka ModeLang. W równaniach różniczkowych zapisywana jest jako składniki:

$$\frac{dA}{dt} = -p \cdot A \tag{5.3}$$

$$\frac{dB}{dt} = p \cdot A \tag{5.4}$$

Reguła 4 - Śmierć

Agent A z pewnego powodu umiera. Reguła ta wykorzystuje interakcję 2 opisaną składnią języka ModeLang. W równaniach różniczkowych zapisywana jest jako składnik:

$$\frac{dA}{dt} = -p \cdot A \tag{5.5}$$

Reguła 5 - Łączenie

Reguła ta opisuje sytuację, gdy dwa agenty źródłowe A oraz B łączą się, aby stworzyć trzeciego agenta C. Na przykład wirion łączy się z komórką niezainfekowaną, aby stworzyć komórkę zainfekowaną. Reguła ta wykorzystuje interakcję 3 opisaną składnią języka ModeLang. W równaniach różniczkowych zapisywana jest jako składniki:

$$\frac{dA}{dt} = -p \cdot A \cdot B \tag{5.6}$$

$$\frac{dB}{dt} = -p \cdot A \cdot B \tag{5.7}$$

$$\frac{dC}{dt} = p \cdot A \cdot B \tag{5.8}$$

Reguła 6 - Ograniczenie

Reguła ta umożliwia wprowadzenie ograniczenia na liczbę agentów konkretnego typu lub sumę liczby agentów kilku typów. Na przykład sumaryczna liczba komórek zdrowych i zainfekowanych nie może być większa niż pojemność organizmu. Reguła ta wykorzystuje interakcję 4 opisaną składnią języka ModeLang. Jeżeli przykładowo liczba Agentów A oraz B nie może przekroczyć wartość AB_{max}, w równaniach różniczkowych można uzyskać ten efekt mnożąc składnik odpowiedzialny za powstawanie agentów (Reguła 1) przez wyrażenie:

$$\frac{dA}{dt} = p_A \cdot A \cdot \left(1 - \frac{A+B}{AB_{max}}\right) \tag{5.9}$$

$$\frac{dB}{dt} = p_B \cdot B \cdot \left(1 - \frac{A+B}{AB_{max}}\right) \tag{5.10}$$

5.2.5 Parametry

Parametry w języku ModeLang używane są do reprezentacji wartości lub ich przedziałów. Konstruując model czasami z góry znana jest konkretna wartość charakteryzująca jakiś proces biologiczny (na przykład częstotliwość jego występowania), a czasami znane są tylko bardziej lub mniej dokładne oszacowania na jej dolną i górną granicę. Be względu na to, który z powyższych przypadków ma miejsce, zazwyczaj wymagana jest znajomość tych wartości, aby umożliwić symulację i analizowanie modelowanego systemu. Aby je opisać w języku ModeLang zostały wprowadzone parametry, które umożliwiają zapisywanie powyższych stałych.

5.2.6 Słowniki wiedzy eksperckiej

Jak opisano wcześniej w rozdziale 5.2.1 dotyczącym przetwarzania opisów modeli w języku ModeLang słowniki wiedzy eksperckiej służą do reprezentacji wiedzy dziedzinowej, która umożliwia przetworzenie opisu wprowadzonego przez eksperta na opis formalny. Przez parser języka ModeLang wykorzystywane są dwa typy słowników:

1. Słownik z terminami dziedzinowymi (*keywords.xml*) definiującymi symbole terminalne występujące w gramatyce języka jako VERB, ADJECTIVE, PREPOSITION oraz NOUN. Jest to słownik, który musi zostać zdefiniowany raz dla każdej dyscypliny, w której jest używany, ale po jego spisaniu może być wielokrotnie wykorzystywany do opisywania wielu

różnych modeli z tej dyscypliny. Co więcej o ile samo napisanie słownika może wymagać drobnej pomocy ze strony informatyka, to później może on być bez problemu wykorzystywany przez biologów nie znających sposobów jego definicji. Dodatek A.6 przedstawia przykładowy słownik, który może być wykorzystywany do modelowania infekcji wirusowych. Natomiast strona internetowa języka ModeLang umożliwia współdzielenie słowników zdefiniowanych dla różnych dziedzin pomiędzy użytkownikami języka.

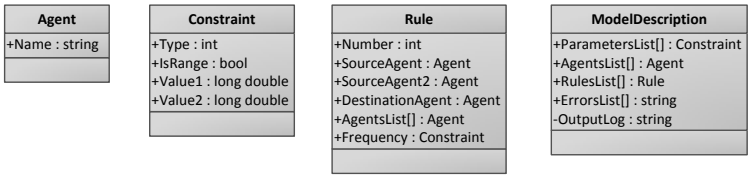
2. Słownik z nazwami ograniczeń (*constraints.xml*) używany do definiowania słów opisujących ograniczenia występujące w interakcjach. Informacje te wydzielono z opisywanego wyżej słownika z terminami dziedzinowymi, gdyż w odróżnieniu od nich słowa kluczowe opisujące ograniczenia zazwyczaj są takie same, niezależnie od dziedziny. Zdefiniować można słowa kluczowe opisujące jeden z pięciu typów ograniczeń (konkretny przykład umieszczony jest w dodatku A.7):

   a) określające częstotliwość zachodzenia danego zjawiska w hercach (liczba wystąpień na sekundę),
   b) określające prawdopodobieństwo zajścia danego zjawiska w ciągu sekundy,
   c) określające średni czas, po którym zjawisko się powtarza,
   d) określające, że wartość jest mniejsza od pewnej stałej,
   e) określające, że wartość jest większa od pewnej stałej.

5.2.7 Implementacja

Parser języka ModeLang został zaimplementowany w języku C++ z wykorzystaniem bibliotek STL oraz plugixml. Parser dostępny jest za darmo, wraz z kodem źródłowym, na podstawie licencji LGPL (Lesser General Public License). Dzięki temu każda osoba, która chciałaby skorzystać z języka ModeLang, ma znacząco ułatwione zadanie. Nawet jeżeli docelowa technologia programistyczna będzie wykorzystywać inny język niż C++, nie powinno być problemu, aby zintegrować kod za pomocą którejś z powszechnie wykorzystywanych metod interoperacyjności (na przykład bibliotek dynamicznych).

Parser języka ModeLang wczytuje opis modelu z pliku tekstowego lub ze standardowego wejścia, natomiast zawartość słowników wiedzy eksperckiej z dedykowanych plików XML. Definicja XML Schema umożliwiająca sprawdzenie poprawności tych plików znajduje się w dodatku A.5. Po wczytaniu i sparsowaniu opisu modelu przechowywany jest on w postaci obiektowej w pamięci komputera. W trakcie procesu parsowania powstaje szczegółowy dziennik opi-



Rysunek 5.4: Reprezentacja opisu modelu w pamięci komputera.

sujący ten proces oraz wykaz błędów w plikach źródłowych. Umożliwiają one dokładną weryfikację przez eksperta definiującego model, czy rzeczywiście komputerowa interpretacja zdefiniowanego przez niego opisu modelu jest zgodna z jego założeniami. Jest to bardzo ważny etap, gdyż przy dużej elastyczności języka ModeLang łatwo może zdarzyć się automatyczna interpretacja sprzeczna z założeniami eksperta.

Ogólny podział na obiekty oraz przechowywane w nich dane przedstawiony został na rysunku 5.4. Oczywiście rzeczywista definicja klas jest bardziej skomplikowana, jednak prezentacja szczegółów technicznych w tym miejscu nie wniosłaby żadnych interesujących treści.

Reprezentacja komputerowa modelu składa się z następujących klas:

- **Agent** - reprezentuje pojedynczy typ agenta, przechowuje jedynie jego nazwę.
- **Constraint** - reprezentuje pojedynczy parametr lub całe ograniczenie przypisane do reguły. Przechowuje informacje o typie ograniczenia i jego wartości/wartościach.
- **Rule** - reprezentuje pojedynczą regułę. Przechowuje jej numer, wykorzystywane w niej typy agentów (źródłowe i docelowe lub ich listę występującą w regule 6) oraz częstotliwość jej zachodzenia.
- **ModelDescription** - centralna klasa reprezentująca cały model. Reprezentuje i przechowuje informacje o parametrach, agentach i regułach oraz informacje z procesu parsowania - utworzony w czasie tego procesu dziennik oraz informacje o błędach, gdy takie wystąpią.

5.3 Analiza przykładowych modeli

Język ModeLang i jego parser zostały zweryfikowane w oparciu o dwa powszechnie wykorzystywane modele - jeden dla infekcji HCV i jeden dla HIV. Wybrane zostały modele, które są najczęściej spotykane (czasami z drobnymi modyfikacjami), aby wykazać przydatność języka przy analizie wielu popularnych problemów. Dla infekcji HCV zdefiniowano parametry, dla których znane są tylko przedziały ich wartości, natomiast w przypadku infekcji HIV podano konkretne wartości dla konkretnego pacjenta. Dzięki temu demonstrowane są dwa odmienne przypadki użycia. Parser języka został również zintegrowany z opisywanym w rozdziale 6 oprogramowaniem do przeprowadzania symulacji wieloagentowej, aby wykazać elastyczność języka i możliwość jego integracji z zewnętrznym oprogramowaniem.

5.3.1 Infekcja HCV

Dla infekcji HCV wybrano model opisany w [RDP09], przedstawiony również w sekcji 4.2.2 i przygotowano jego opis w języku ModeLang. Matematyczna reprezentacja tego modelu może być przedstawiona za pomocą następującego układu równań:

$$\frac{dT}{dt} = s + r_T T \left(1 - \frac{T+I}{T_{max}}\right) - d_T T - (1-\eta)\beta V T + qI \tag{5.11}$$

$$\frac{dI}{dt} = r_I I \left(1 - \frac{T+I}{T_{max}}\right) + (1-\eta)\beta V T - d_I I - qI \tag{5.12}$$

$$\frac{dV}{dt} = (1-\epsilon)pI - cV \tag{5.13}$$

Powyższe równania obrazują najczęściej stosowany sposób opisu modeli infekcji wirusowych za pomocą wzorów matematycznych. Jednak dzięki językowi ModeLang ich znajomość oraz rozumienie przestaje mieć znaczenie dla biologów. Dzięki temu językowi model może być przedstawiony w prosty w interpretacji sposób pokazany na listingu 5.2.

Listing 5.2: Model infekcji HCV opisany w języku ModeLang.

```
1 0 < s < 4
2 0.002 < rHU < 3.4
3 0 < rHI < 1.4e-6
4 1e-3 < dHU < 0.014
```

```
1e-3 < dHI < 0.5
4e6 < Umax < 1.3e7
1e-8 < beta < 1e-6
0.1 < pH < 44
0.8 < cv < 22
0 < cI < 1

Healthy hepatocytes are created by Healthy hepatocytes at speed rHU
Infected hepatocytes are created by Infected hepatocytes at speed rHI
The number of Healthy hepatocytes and Infected hepatocytes is less than Umax
Blastic cells change into Healthy hepatocytes at speed s
Healthy hepatocytes die with mean time dHU
Infected hepatocytes die with mean time dHI
Healthy hepatocytes and Virions generate Infected hepatocytes with probability beta
Virions are emitted by Infected hepatocytes at rate pH
Virions die with probability cv
Infected hepatocytes change into Healthy hepatocytes with probability cI
```

Zaproponowana reprezentacja została pomyślnie sprasowana przez parser język ModeLang. Fragment pliku dziennika, który został w tym procesie wygenerowany, przedstawiony jest na listingu 5.3.

Listing 5.3: Dziennik z komunikatami wygenerowanymi w czasie parsowania skyptu przedstawionego na listingu 5.2.

```
...

Rule:
0.8 < cv < 22

Rule:
0 < cI < 1

Rule:
Healthy hepatocytes are created by Healthy hepatocytes at speed rHU
Matched rule: 1
Agent: Healthy hepatocytes
Agent: Healthy hepatocytes

Rule:
Infected hepatocytes are created by Infected hepatocytes at speed rHI
Matched rule: 1
Agent: Infected hepatocytes
Agent: Infected hepatocytes
```

```
20
21 ...
```

5.3.2 Infekcja HIV

Dla infekcji HIV wybrano model opisany w [CR00] i przygotowano jego opis w języku ModeLang. Matematyczna reprezentacja tego modelu może być przedstawiona za pomocą następującego układu równań:

$$\frac{dT}{dt} = s - \mu_T T + rT\left(1 - \frac{T+I}{T_{max}}\right) - k_1 VT \tag{5.14}$$

$$\frac{dI}{dt} = k_1' VT - \mu_I I \tag{5.15}$$

$$\frac{dV}{dt} = N\mu_B I - k_1 VT - \mu_V V \tag{5.16}$$

Powyższy model, przedstawiony w języku ModeLang, prezentowany jest na listingu 5.4.

Listing 5.4: Model infekcji HIV opisany w języku ModeLang.

```
1  s = 10
2  dT = 0.02
3  dI = 0.26
4  dV = 2.4
5  r = 0.03
6  p = 120
7  k1 = 2e-5
8  k1p = 0.4e-5
9  Tmax = 1500
10
11 Healthy T-cells are created from precursors at speed s
12 Healthy T-cells dies with mean time dT
13 Healthy T-cells are created by Healthy hepatocytes at speed r
14 Number of Healthy T-cells and Infected T-cells is less than Tmax
15 Healthy T-cells and Virions merge to Infected T-cells with
       probability k1
16 Healthy T-cells and Virions merge to Noninfectious T-cells with
       probability k1p
17 Infected T-cells dies with mean time dI
18 Virions are emitted by Infected hepatocytes at rate p
19 Virions die with probability dV
```

Tak samo jak w przypadku opisu infekcji HCV, opis infekcji HIV został poprawnie wczytany i zinterpretowany przez parser języka ModeLang.

5.4 Zastosowanie w innych obszarach nauki

Język ModeLang zaprojektowany został w celu rozwiązania problemów, które pojawiały się w czasie konkretnej współpracy z biologami w zakresie modelowania infekcji wirusowych. Dlatego też jego projekt i implementacja koncentrują się głównie na tym zastosowaniu. Jednak dzięki intuicyjności dla ekspertów, język ModeLang stosowany do opisu modeli dynamicznych może z pewnością znaleźć zastosowanie również na innych polach biologii oraz w zupełnie innych dyscyplinach, takich jak na przykład fizyka i chemia. W najprostszych sytuacjach wystarczyłoby tylko jednokrotnie skonstruować słowniki wiedzy eksperckiej dla danej dyscypliny. W praktyce jednak może okazać się, że wymagane jest w tym celu dodanie nowej reguły w oparciu o którąś interakcję, albo nawet zdefiniowanie nowej interakcji. Nie mniej nawet w takim przypadku ModeLang może być interesującym i przydatnym narzędziem, jedynie wymagana praca wejścia będzie trochę większa. Na pewno zagadnienie możliwości zastosowania języka w innych dziedzinach jest bardzo interesujące i warte dalszego, szczegółowego zbadania, jednak jego dokładna analiza znacząco przekracza zakres tej pracy, tak więc nie będzie w niej przedstawiana.

6
Wieloagentowy model infekcji HCV

Jako alternatywa dla modelu opartego o równania różniczkowe przedstawionego w sekcji 4.1.1 zaproponowany przez autora został model oparty o symulację wieloagentową. Model ten reprezentuje komórki w zainfekowanym organizmie w postaci niezależnych agentów zgodnie z koncepcją opisaną w sekcji 3.6. Takie podejście umożliwia dokonywanie bardziej złożonych i zaawansowanych analiz niż w przypadku modeli opartych o równania różniczkowe. W rozdziale tym przedstawiono w sekcjach 6.1 i 6.2 projekt i implementację takiego modelu, w sekcji 6.3 analizę jego złożoności, następnie w sekcji 6.4 wyniki eksperymentu obliczeniowego wykorzystującego zaimplementowany model, a na końcu w sekcji 6.5 podsumowanie i opis zalet zaproponowanego podejścia.

Rezultaty przedstawione w tym rozdziale są w całości wynikiem prac badawczych autora, prowadzonych pod kierownictwem promotora. Wyniki te zostały opublikowane w pracy [WJFB11], zgłoszone do publikacji w [WJFB12] oraz zaprezentowane na licznych konferencjach krajowych i międzynarodowych. Dwukrotnie prezentowane prace uzyskały specjalne wyróżnienie na warsztatach organizowanych przez Polskie Towarzystwo Bioinformatyczne. W 2010 roku w Ustroniu autorowi przyznano nagrodę za najlepszy plakat, natomiast w 2012 roku autorowi przyznano nagrodę za najlepszą prezentację. Konsultantami biologicznej części prac byli profesor Marek Figlerowicz oraz doktor Paulina Jackowiak z Instytutu Chemii Bioorganicznej PAN w Poznaniu.

6.1 Projekt i implementacja modelu

6.1.1 Projekt

Koncepcja modelu wieloagentowego zakłada, że występują w nim niezależne, oddziaływujące na siebie agenty umieszczone w pewnym, zazwyczaj zmiennym środowisku (patrz sekcja 3.6). W przypadku infekcji HCV zidentyfikowano trzy typy agentów: zainfekowane hepatocyty, niezainfekowane hepatocyty oraz wolne wiriony. Ze względu na złożoność organizmu i konieczność zamodelowania wielu składających się na niego komórek, liczba agentów każdego typu biorąca udział w symulacji powinna wynosić tysiące, a być może nawet setki tysięcy komórek. Agenty te współistnieją w środowisku, którym jest organizm zainfekowanego człowieka. Interakcje, które występują pomiędzy agentami oraz pomiędzy agentami i środowiskiem zdefiniowane zostały na bazie opisanego wcześniej modelu klasycznego [RDP09]. Interakcje te prezentowane są na rysunku 6.1, a ich dokładna implementacja komputerowa została opisana w sekcji 6.1.2.

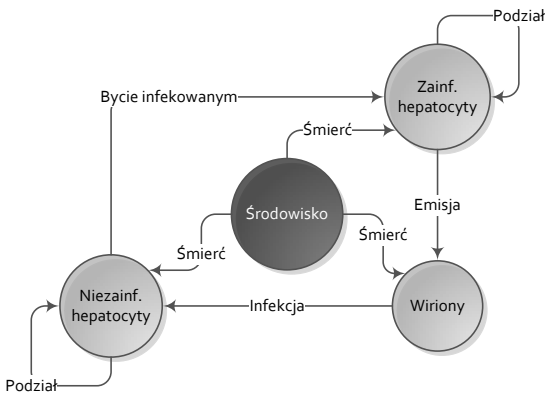


Figure 6.1: Interakcje występujące w modelu wieloagentowym. Hepatocyty mogą rozmnażać się przez podział, a po pewnym czasie umierają. Niezainfekowane komórki mogą być infekowane przez wiriony i następnie produkować nowe kopie wirionów, które mogą być usuwane przez układ immunologiczny.

Każda interakcja opisana jest przez jeden lub więcej parametrów. Są one odpowiednikami parametrów występujących w równaniach różniczkowych opisanych w sekcji 4.1.1. Pominięty został parametr q opisujący zdrowienie komórek zainfekowanych, gdyż w praktyce proces ten zachodzi tak wolno, że jest pomijalny [RDP09]. Szczegółowo wykorzystane parametry zostały opisane w tablicy 6.1.

Table 6.1: Parametry wykorzystywane w modelu wieloagentowym.

Parametr	Opis
β	tempo infekcji
d_T	tempo umierania niezainfekowanych hepatocytów
d_I	tempo umierania zainfekowanych hepatocytów
r_T	tempo podziału niezainfekowanych hepatocytów
r_I	tempo podziału zainfekowanych hepatocytów
T_{max}	pojemność wątroby (maksymalna liczba hepatocytów)
p	tempo produkcji wirionów
c	tempo usuwania wirionów przez układ immunologiczny
ϵ	procent zmniejszenia współczynnika p podczas terapii
η	procent zmniejszenia współczynnika β podczas terapii

6.1.2 Implementacja

Środowisko, w którym symulowane są agenty zamodelowano za pomocą prostokątnej siatki. Co prawda istnieją pewne przesłanki do użycia siatki sześciennej [BOB07], to nie została ona wybrana ze względu na gorszą wydajność przetwarzania oraz trudniejszą implementację, szczególnie że przeprowadzone eksperymenty wykazały, że siatka prostokątna w implementowanym zastosowaniu jest wystarczająca. Każdy agent posiada w powyższym środowisku swoją pozycję, która inicjalizowana jest losowo z rozkładem jednostajnym. Po fazie inicjalizacji następuje właściwa symulacja, której każdy krok modeluje upływ krótkiego momentu czasu i składa się z następujących etapów:

1. **Przemieszczanie się agentów.** W tej fazie każdy ruchomy agent jest przemieszczany. W przypadku opisywanego modelu dotyczy to jedynie wirionów, które mogą przemieszczać się płynąc z krwią przepływającą przez wątrobę.
2. **Interakcje pomiędzy agentami i środowiskiem.** W tej fazie symulowane są interakcje pomiędzy agentami, a środowiskiem. Umierające hepatocyty i usuwane z organizmu wiriony są usuwane z komputerowej reprezentacji modelu, nowe hepatocyty powstają przez podział oraz emito-

wane są nowe wiriony, zgodnie z interakcjami przedstawionymi w sekcji 6.1.1. Każda nowa komórka otrzymuje losowe współrzędne znajdujące się w bezpośrednim sąsiedztwie komórki, z której została utworzona.

3. **Aktualizacja środowiska.** Po aktualizacji liczby oraz pozycji komórek w poprzednich fazach wymagana jest aktualizacja komputerowej reprezentacji modelu. Faza ta nie modeluje żadnego procesu biologicznego, a wymagana jest ze względów wydajnościowych, aby kolejny etap można wykonać na komputerze w krótszym czasie.
4. **Wzajemne interakcje pomiędzy agentami.** Na tym etapie zachodzą wzajemne interakcje pomiędzy agentami. W opisywanym modelu jest to jedynie proces infekcji zdrowych hepatocytów przez wiriony.
5. **Aktualizacja statystyk.** Po wykonaniu kolejnej iteracji symulacji pewne statystyki, takie jak liczba komórek każdego rodzaju, są zapisywane na dysku w celu umożliwienia ich późniejszej analizy.

Ponieważ modelowane interakcje zachodzą w organizmie stosunkowo wolno stwierdzono, że jeden dzień wystarczy symulować za pomocą 50 opisanych powyżej iteracji (czyli jedna iteracja modeluje niecałe pół godziny czasu). Założenie to zostało zweryfikowane za pomocą porównywania wyników symulacji z wynikami modelu opartego o równania różniczkowe.

Symulacja wykonuje się tak długo, aż wykonana zostanie wyznaczona liczba kroków, albo liczba komórek się ustabilizuje.

6.2 Algorytm dostrajania parametrów modelu

Największym problemem w trakcie analizy modelu było znalezienie wartości parametrów prezentowanych w tablicy 6.1 dla konkretnego pacjenta, dla którego dostępne są dane o historycznych poziomach RNA wirusa we krwi. W przypadku równań różniczkowych wykorzystać można powszechnie używane metody służące do dopasowywania modelu do danych. Metod tych nie da się zastosować w przypadku symulacji wieloagentowej, która wykorzystuje bardziej skomplikowane algorytmy i nie może być zredukowana do równań matematycznych, które można by dopasować do danych. Z tego powodu nie da się wyznaczyć wartości parametrów przed rozpoczęciem symulacji. Aby znaleźć ich wartości zastosowana została metoda odwróconej symulacji [Ter07]. Ogólna idea tego typu symulacji prezentowana jest na rysunku 6.2.

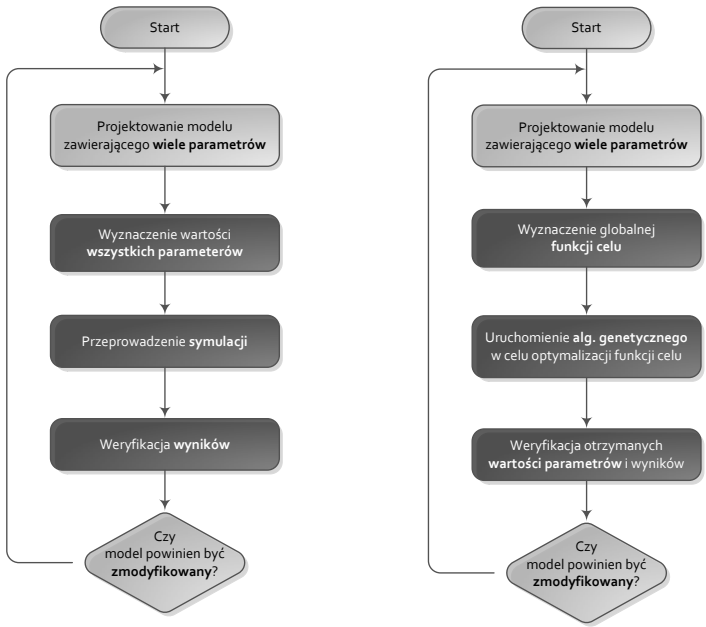


Figure 6.2: Metody symulowania modeli. Lewy schemat przedstawia klasyczny algorytm symulacji i weryfikacji modelu. Prawy schemat przedstawia metodę odwróconej symulacji.

Gdy symulowany jest skomplikowany model, opisany za pomocą wielu parametrów z wykorzystaniem klasycznego podejścia do symulacji, parametry występujące w modelu są znane od początku lub ich wyznaczenie nie jest skomplikowane (można zastosować na przykład dopasowywanie do danych). Gdy symulacja zostanie zakończona jej wyniki są analizowane i weryfikowane. Jeżeli są one niezadowalające model może być poprawiony, a cały proces powtórzony.

W metodzie odwróconej symulacji na starcie zdefiniowana jest funkcja celu (na przykład najlepsze dopasowanie modelu do danych), a następnie pewien algorytm przeszukiwania jest używany, aby tą funkcję zoptymalizować. Najczęściej algorytmem tym jest implementacja pewnej metaheurystyki, na przykład algorytmu genetycznego. W rezultacie otrzymywany jest zbiór wartości parametrów, które optymalizują funkcję celu, jest on weryfikowany w oparciu o analizę wartości tych parametrów, wartości funkcji celu oraz analizę przebiegu symulacji. Jeżeli stwierdzone zostanie, że istnieje taka konieczność, model lub funkcja celu mogą być zmodyfikowane, a cały proces powtórzony.

W celu dokonania analizy wieloagentowego modelu infekcji HCV opisywanego w tym rozdziale zdecydowano się na użycie algorytmu genetycznego [PS10], jako algorytmu optymalizującego funkcję celu. Szczegółowy opis optymalizowanej funkcji znajduje się w dalszej części tej sekcji. Natomiast w trakcie implementacji algorytmu genetycznego wzorowano się na pracy [Wri91] zawierającej szczegółowe porównanie i testy różnych implementacji algorytmów genetycznych dla problemów optymalizujących funkcje wielu parametrów będących liczbami rzeczywistymi. Zgodnie z powyższą publikacją użyto kodowania parametrów za pomocą liczb rzeczywistych znormalizowanych do przedziału $\langle 0{,}0; 1{,}0\rangle$. Tak więc chromosom reprezentowany był jako lista liczb rzeczywistych. Na etapie selekcji używana była metoda ruletki. Jako algorytm krzyżowania użyto podziału jednopunktowego, który dzieli oba chromosomy w tym samym miejscu, wypadającym pomiędzy dwoma liczbami reprezentującymi parametry i wymienia pomiędzy sobą obie części. Jako siłę mutacji zastosowano wartość 0,1, co oznacza, że wartość rzeczywistego parametru wynosząca v była zmieniana na liczbę losową z przedziału $(max(0; v - 0{,}1); v)$ lub $(v; min(1; v + 0{,}1))$ w zależności od losowo wybranego kierunku mutacji. Prawdopodobieństwo mutacji wynosiło 0,05, co zgodnie z analizą we [Wri91] przynosi najlepsze rezultaty.

Jednym z ważniejszych etapów projektowania i implementacji algorytmu genetycznego było zdefiniowanie i przetestowanie optymalizowanej funkcji celu. Niech $\mathcal{V}$ oznacza zbiór punktów przechowujących informację o poziomie RNA wirusa we krwi zmierzoną w czasie badań klinicznych. Pierwsza współrzędna takiego punktu określa dzień od początku symulacji, w którym pomiar został dokonany, a druga wartość poziomu RNA. Tak więc jeżeli $(d; v) \in \mathcal{V}$ oznacza to, że dnia d u pacjent został zmierzony poziom RNA wirusa we krwi i wynosił on v $\frac{\log \text{IU}}{\text{ml}}$. Niech $\mathcal{I}$, $\mathcal{T}$ oraz $\mathcal{H}$ oznaczają analogiczne zbiory dla hepatocytów zainfekowanych, niezainfekowanych oraz ich sumy (całkowitej liczby hepatocytów). Ponieważ jak zauważono w sekcji 2.3.2 zliczenie zdrowych i zainfekowanych hepatocytów, które wymaga biopsji wątroby, jest bardzo problematyczne to zazwyczaj w danych wejściowych zbiory $\mathcal{I}$, $\mathcal{T}$ oraz $\mathcal{H}$ są puste. Niech $M_{V,d}$ oznacza poziom RNA wirusa we krwi obliczony przez symulację wieloagentową w dniu d, a $M_{I,d}$, $M_{T,d}$ oraz $M_{H,d}$ określają analogiczną wartość dla hepatocytów zainfekowanych, niezainfekowanych i ich sumy. W celu optymalizacji parametrów modelu wieloagentowego przetestowano dwa podejścia. Pierwszym z nich jest optymalizacja funkcji, która minimalizuje sumę różnic pomiędzy danymi eksperymentalnymi, a symulacyjnymi:

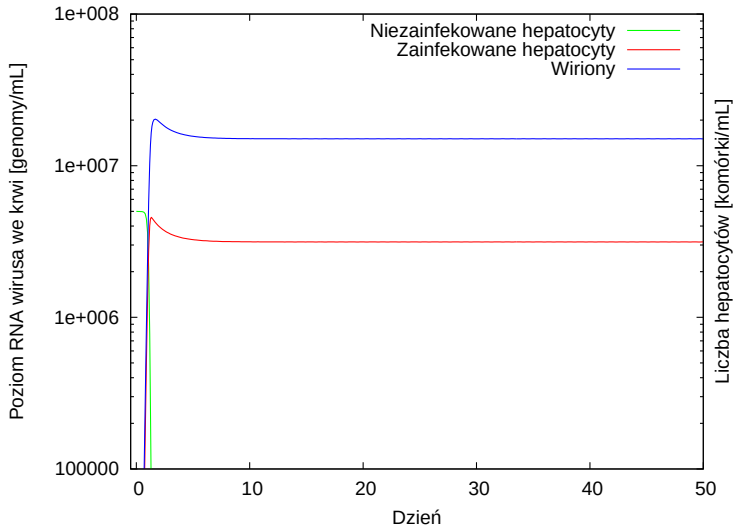
$$min \sum_{(d;v)\in\mathcal{V}} |v - M_{V,d}| + \sum_{(d;t)\in\mathcal{T}} |t - M_{T,d}| + \sum_{(d;i)\in\mathcal{I}} |i - M_{I,d}| + \\ + \sum_{(d;h)\in\mathcal{H}} |h - M_{H,d}| \,. \tag{6.1}$$

Drugim podejściem była dodatkowa normalizacja każdej różnicy poprzez podzielenie jej przez wartość oczekiwaną:

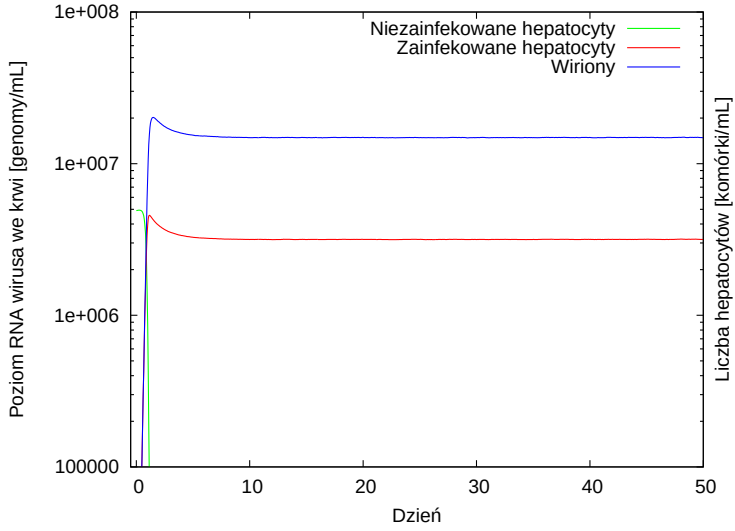
$$min \sum_{(d;v)\in\mathcal{V}} \frac{|v - M_{V,d}|}{v} + \sum_{(d;t)\in\mathcal{T}} \frac{|t - M_{T,d}|}{t} + \sum_{(d;i)\in\mathcal{I}} \frac{|i - M_{I,d}|}{i} + \\ + \sum_{(d;h)\in\mathcal{H}} \frac{|h - M_{H,d}|}{h} \,. \tag{6.2}$$

Eksperymenty obliczeniowe pokazały, że pierwsza postać funkcji przynosi lepsze rezultaty. Ponieważ zdarzało się, że dla pewnych momentów czasowych liczba wirionów w danych klinicznych była stosunkowo mała, szybko okazywało się, że algorytm genetyczny w drugim przypadku stara się zminimalizować różnicę w tym punkcie, gdyż wynosiła ona kilkukrotność wartości oczekiwanej. Natomiast z punktu widzenia biologicznego ta kilkukrotna różnica wartości przekładała się na stosunkowo niewielką liczbę komórek, która nie była aż tak istotna.

Poprawność implementacji została zweryfikowana poprzez porównanie jej z wynikami symulacji modelu opartego o równania różniczkowe. Weryfikacja pokazała, że korzystając z algorytmu genetycznego można wyznaczyć te same, poprawne wartości parametrów co poprzez dopasowywanie równań różniczkowych do danych. Wyniki przykładowego porównania przedstawiono na rysunku 6.3. Na wykresach można zauważyć, że wyniki obu modeli są bardzo zbliżone, co potwierdza również analiza numeryczna danych.



(a) Symulacja równań różniczkowych w programie Simulink.



(b) Symulacja wieloagentowa.

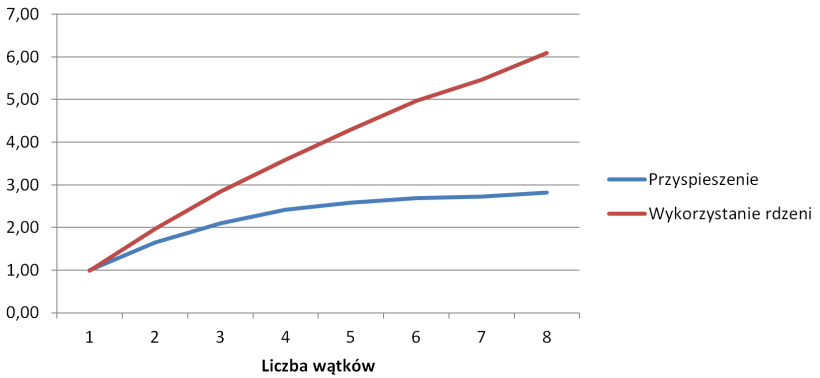
Figure 6.3: Wyniki weryfikacji, czy model wieloagentowy może wygenerować takie same, poprawne wyniki jak model klasyczny.

6.3 Złożoność obliczeniowa

Złożoność obliczeniowa pojedynczej symulacji opisanej w sekcji 6.1.2 wynosi $O(d \cdot (n+m))$ gdzie d oznacza liczbę kroków symulacji, które należy wykonać, n oznacza maksymalną liczbę agentów dowolnego typu występujących w pewnym kroku symulacji, a m oznacza całkowitą liczbę komórek siatki modelującej środowisko. Uwzględniając, że zarówno n, jak i m mogą wynosić kilkaset tysięcy, a d kilka tysięcy, pojedyncza symulacja może wykonywać się nawet około 5 minut, co zostało potwierdzone eksperymentalnie. Nie byłby to czas długi, gdyby nie fakt, że opisane w sekcji 6.2 wykorzystanie symulacji w algorytmie genetycznym wymaga wykonania kilkuset kroków, z których każdy polega na zasymulowaniu modelu 20 do 30 razy. Sumarycznie może to spowodować, że pojedynczy eksperyment przeprowadzony zgodnie z opisem w sekcji 6.2 będzie trwał wiele dni. Aby skrócić ten czas zastosowano następujące optymalizacje:

1. Symulacja została zaimplementowana ze szczególną dbałością o jej wydajność w języku C++, który generuje dużo wydajniejsze programy niż na przykład C# lub Java. Dodatkowo każdy często wykonywany fragment kodu został sprawdzony z użyciem profilera i powtórnie zoptymalizowany.
2. Biblioteka OpenMP została wykorzystana, aby zrównoleglić obliczenia i wykorzystać wszystkie osiem rdzeni procesora Intel Core i7, na którym przeprowadzane były eksperymenty. Wykorzystana została biblioteka OpenMP 3.0 stanowiąca integralną część kompilatora g++ 4.5. Zrównoleglenie realizowane jest poprzez ocenę każdego z kilkudziesięciu analizowanych w każdym pokoleniu fenotypów na kolejnym wolnym procesorze.

   Prezentacja przyspieszenia, które zostało osiągnięte dzięki zrównolegleniu kodu znajduje się na rysunku 6.4. Na wykresie wyraźnie widać, że dodanie czwartego rdzenia powoduje cały czas znaczące zwiększenie przyspieszenia, które dla większej liczby rdzeni przestaje już być tak duże. Spowodowane jest to architekturą procesora w maszynie testowej. W procesorze tym istnieją w rzeczywistości tylko cztery rdzenie, a cztery kolejne są wirtualne, stworzone dzięki technologi hyper-threading, dlatego też nie mają znaczącego wpływu na przyspieszenie. Co ciekawe można zaobserwować, że dla ośmiu rdzeni średnio wykorzystywany jest tylko czas obliczeń możliwy do wykonania na sześciu rdzeniach. Spowodowane jest to nieefektywną końcówką oceny pokolenia fenotypów, gdy wszystkie osobniki mogą być już ocenione, oprócz jednego, który będzie blokował ocenę kolejnego pokolenia, aż jego ocena zostanie zakończona (wcześniej nie



Rysunek 6.4: Przyspieszenie symulacji w zależności od liczby użytych rdzeni oraz całkowity sumaryczny czas rdzeni procesora zużyty na obliczenia.

można przeprowadzić etapu selekcji).

Powyższe optymalizacje pozwoliły skrócić długość pojedynczego eksperymentu do poniżej 6 godzin, co jest wynikiem akceptowalnym z praktycznego punktu widzenia.

6.4 Eksperyment obliczeniowy

Aby zweryfikować, czy klasyczny model skrytykowany w sekcji 4.2.2 faktycznie jest niedokładny, czy być może wystarczy wyznaczyć inne wartości parametrów niż te podane w opisującym go artykule, przeprowadzono symulację za pomocą opisywanej w tym rozdziale metody opartej o agenty. W tym celu zweryfikowane zostały dwie nieścisłości, które zostały wskazane w sekcji 4.2.2:

1. Zbyt niska całkowita liczba hepatocytów, występująca krótko po zainfekowaniu chorego. Ten problem jest weryfikowany w sekcji 6.4.1.
2. Zbyt niski poziom niezainfekowanych hepatocytów w stosunku do wszystkich hepatocytów w wątrobie. Ten problem jest weryfikowany w sekcji 6.4.2.

Dla każdego z wymienionych powyżej problemów zweryfikowane zostały dwa przypadki:

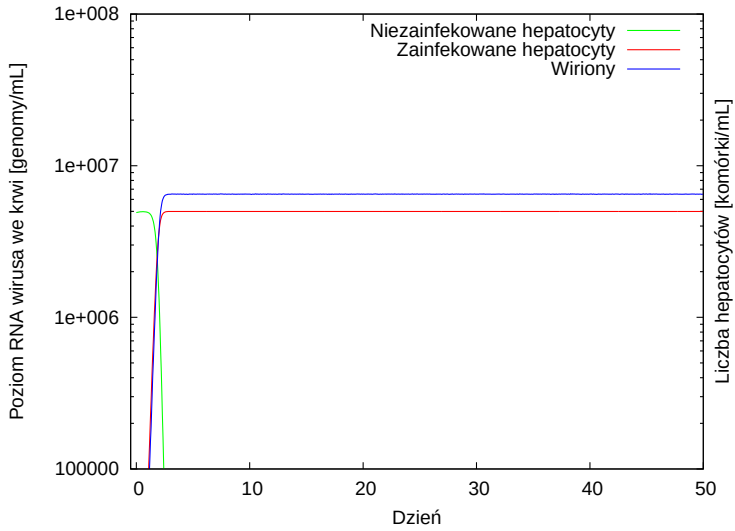
1. Podstawowa weryfikacja, w której symulowana liczba wirionów nie musi być zgodna z danymi klinicznymi o poziomie RNA wirusa we krwi. Z punktu widzenia znaczenia biologicznego otrzymanych wniosków taka weryfikacja nie ma sensu, jednak z punktu widzenia obliczeniowego jest to dobra wstępna selekcja. Jeżeli modelowi nie uda się przejść tego etapu, to tym bardziej nie będzie on w stanie poprawnie modelować przypadku, w którym weryfikowany będzie poziom RNA.
2. Bardziej realistyczny, a zarazem bardziej skomplikowany przypadek, w którym oprócz założenia sprawdzanego w punkcie pierwszym (opisanego szczegółowo w dalszych podsekcjach tej sekcji) weryfikowana jest również liczba wirionów, która musi być zgodna z danymi klinicznymi.

Dzięki zastosowaniu opisanego i zaimplementowanego wcześniej algorytmu genetycznego oba problemy w obu wersjach były łatwe do zweryfikowania.

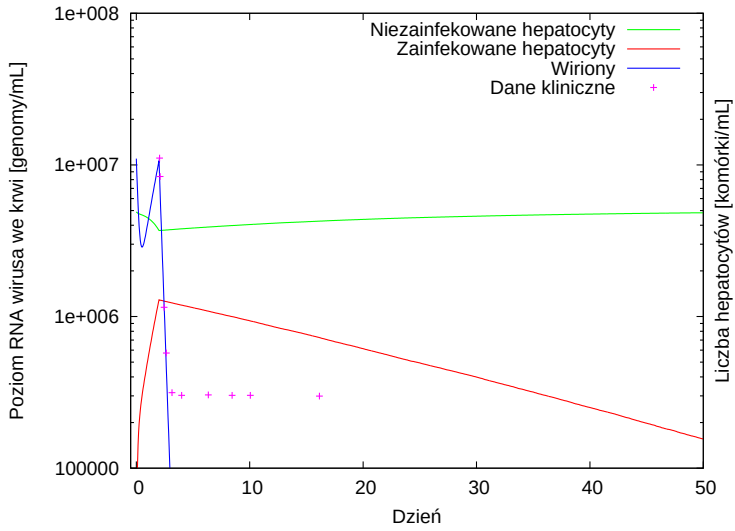
6.4.1 Maksymalizacja całkowitej liczby hepatocytów

Opisana w sekcji 6.2 funkcja celu okazała się skuteczna, jeżeli danymi wejściowymi jest poziom RNA wirusa. Aby dodatkowo sprawdzić, czy możliwe jest takie dobranie parametrów, aby również całkowita liczba hepatocytów miała realistyczną wartość, wykorzystano występujący w funkcji celu zbiór $\mathcal{H}$ (zdefiniowany w sekcji 6.2). Do zbioru tego dodano kilka punktów o wartościach $h = 4{,}98$, czyli takich ile wynosi oczekiwana liczba hepatocytów (patrz sekcja 4.2.2). Została dodana taka sama liczba punktów co liczność zbioru $\mathcal{V}$, aby optymalizacja wartości z obu zbiorów miała równą wagę w funkcji celu. Pierwszy punkt otrzymał współrzędną czasową d równą dniu, w którym po rozpoczęciu terapii ustabilizował się poziom wirionów, natomiast kolejne rozłożone były co 3 kolejne dni.

Wyniki przedstawione są na rysunku 6.6. Pokazują one, że istnieje możliwość zamodelowania wysokiego poziomu hepatocytów, ale wyłącznie wtedy, gdy liczba wirionów nie musi być zgodna z danymi klinicznymi. W przeciwnym wypadku niemożliwe jest prawidłowe zamodelowanie systemu.



(a) Poziom wirusa nie musi odzwierciedlać danych klinicznych. Liczba hepatocytów może być wysoka.



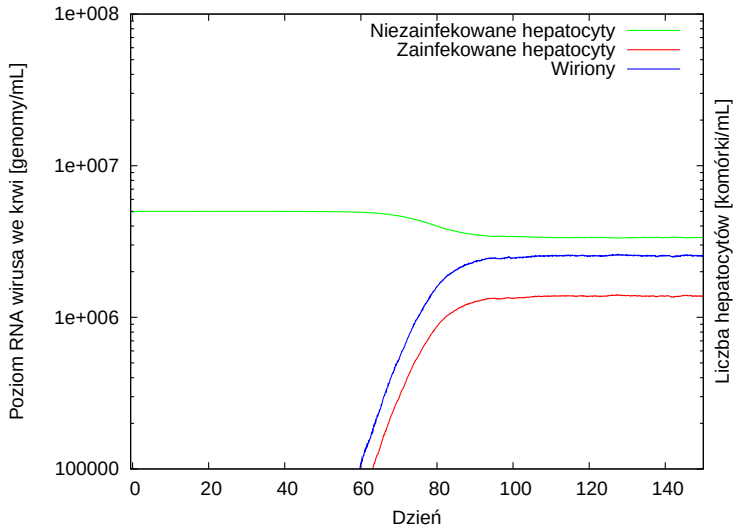
(b) Poziom wirusa powinien być zgodny z danymi klinicznymi. Niestety nie da się tak dobrać parametrów, żeby warunek ten był spełniony.

Rysunek 6.5: Wyniki symulacji, która maksymalizuje liczbę hepatocytów. Realistyczną liczbę hepatocytów można uzyskać jedynie, jeżeli poziom wirusa nie musi odzwierciedlać danych klinicznych.

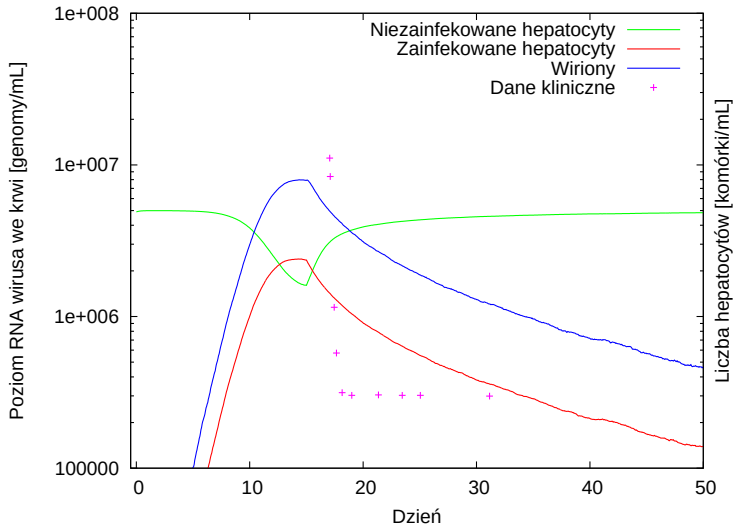
6.4.2 Maksymalizacja liczby niezainfekowanych hepatocytów

W trakcie analizy modelu klasycznego w sekcji 4.2.2 zauważono, że liczba zdrowych hepatocytów spada prawie do zera, co jest niezgodne z wynikami prowadzonych badań [LSX+09]. Dlatego też zdecydowano się zweryfikować ten fakt korzystając z tej samej metody co opisana w sekcji 6.4.1. Jedyną różnicą była konieczność drobnego zmodyfikowania danych wejściowych. W tym przypadku dane dodano do zbioru $\mathcal{T}$, a nie $\mathcal{H}$. Współrzędna czasowa punktów została wyznaczona tak samo jak w sekcji 6.4.1, natomiast dla współrzędnej t zweryfikowano kilka możliwości. Ponieważ zgodnie z obserwacjami w [LSX+09] liczba niezainfekowanych komórek cechuje się dość dużą zmiennością w czasie, wynoszącą nawet kilkanaście procent, zweryfikowano kolejne poziomy liczby niezainfekowanych komórek wynoszące 10%, 20% i dalej co 10%, aż do 90% całkowitej liczby hepatocytów. Pozwoliło to uniknąć konieczności implementacji modyfikacji algorytmu genetycznego maksymalizującej pewne kryterium (liczba hepatocytów) przy jednoczesnej optymalizacji funkcji celu. Ponieważ gęstość sprawdzanych wartości jest większa niż zmienność liczby niezainfekowanych hepatocytów, jeżeli istniałby zestaw parametrów spełniający założenia co do poziomu wirusa i modelujący wysoki poziom hepatocytów niezainfekowanych, musiałby on zostać znaleziony. Jeżeli natomiast w przyszłości, w innym zastosowaniu rzeczywiście potrzebna byłaby funkcjonalność maksymalizacji kryterium, można stosunkowo łatwo ją dodać, na przykład korzystając z jednej z metod opisanych w [KCS05].

Tak samo jak w sekcji 6.4.1 również w tym przypadku wyniki pokazały, że nie można wyznaczyć zestawu parametrów spełniającego wszystkie założenia. Przykładowe wyniki symulacji, dla liczby hepatocytów niezainfekowanych stanowiących 90% wszystkich hepatocytów przedstawiono na rysunku 6.6.



(a) Poziom wirusa nie musi odzwierciedlać danych klinicznych. Liczba hepatocytów niezainfekowanych może być stosunkowo wysoka.



(b) Poziom wirusa powinien być zgodny z danymi klinicznymi. Niestety nie da się tak dobrać parametrów, żeby wszystkie założenia były spełnione.

Rysunek 6.6: Wyniki symulacji, która stara się osiągnąć liczbę hepatocytów zdrowych równą 90% wszystkich hepatocytów. Stosunek zbliżony do 90% można uzyskać co najwyżej, jeżeli poziom wirusa nie musi odzwierciedlać danych klinicznych. Takie same wyniki osiągnięto dla innych poziomów hepatocytów niezainfekowanych.

6.5 Podsumowanie

W rozdziale tym zaprezentowany został model infekcji HCV oparty o symulację wieloagentową. Model został zaprojektowany w oparciu o klasyczny model oparty na równaniach różniczkowych, ale wykorzystana została idea systemu wieloagentowego, która ma dużo większe możliwości. Jedną z nich zademonstrowano w sekcji 6.4 opisującej eksperyment obliczeniowy poprzez zdefiniowanie własnej, niestandardowej funkcji celu, maksymalizującej pewną wartość. Inne zalety wykorzystania modelu wieloagentowego (patrz również [AMDMV09]) to:

1. Interakcje występujące w modelu są opisane prostym do zrozumienia językiem. Do ich zapisu można na przykład wykorzystać prezentowany w rozdziale 5 język ModeLang. Dzięki temu opis jest bardziej zrozumiały dla biologów. Umożliwia to również łatwe dodawanie kolejnych reguł do modelu, które mogą dokładniej modelować złożoną naturę modelowanego systemu.
2. Można łatwiej modyfikować reguły. W przypadku równań różniczkowych wprowadzenie drobnej zmiany może wiązać się z koniecznością przeprowadzenia od nowa skomplikowanych obliczeń analitycznych. W przypadku symulacji wieloagentowej wymagana jest tylko drobna zmiana w opisie modelu. Co więcej raz zaimplementwane środowisko symulacyjne może być wykorzystywane ponownie przy modelowaniu wielu, różnorodnych systemów.
3. Funkcja celu oraz definicje ograniczeń mogą być bardziej złożone. Nie muszą ograniczać się do dopasowywania do danych, ale mogą definiować na przykład maksima i minima pewnych procesów lub zmiany zachodzące w nich w konkretnych punktach czasowych.
4. Można łatwo zdefiniować szczegółowe zależności przestrzenne i rozróżnić każdą komórkę danego typu na przykład poprzez przypisanie jej atrybutów.
5. Istnieje więcej możliwości analizowania wyników symulacji, ponieważ każda komórka symulowana jest oddzielnie. Dzięki temu można na przykład analizować jak pojedyncza komórka lub grupa komórek wpływa na wyniki eksperymentu.
6. Łatwo można zamodelować losowość poprzez wprowadzenie do symulacji zmiennych losowych.

Oprócz sposobu implementacji i symulowania wieloagentowego modelu infekcji HCV bieżący rozdział przedstawia również opis metody odwróconej symulacji. Metoda ta wykorzystywana jest do znajdowania wartości parametrów występujących w modelu, a jej skuteczność została potwierdzona w eksperymencie obliczeniowym. Korzystając z tej metody oraz zaproponowanego sposobu modelowania można przeprowadzać nowe, interesujące eksperymenty obliczeniowe analizujące proces infekcji i terapii HCV w organizmie ludzkim. Istotne jest również to, że zaproponowane sposoby analizy infekcji wirusowej nie zostały jeszcze nigdzie w literaturze opisane.

7
Populacyjny model infekcji HCV

Jak szczegółowo opisano w rozdziale 2.3.3 poświęconym terapii HCV, leczenie osób zainfekowanych wirusem jest aktualnie bardzo nieefektywne. Najnowsze schematy terapii opracowane w ostatnich latach umożliwiają w przypadku niektórych genotypów wyleczenie tylko trochę ponad połowy pacjentów. Co więcej, leczeniu towarzyszy wiele niepożądanych efektów ubocznych, takich jak objawy zbliżone do grypy oraz problemy żołądkowo-jelitowe i psychiatryczne [Lia00, MH00]. Celem modelu opisywanego w tym rozdziale była analiza zależności pomiędzy różnymi wskaźnikami opisującymi zaawansowanie infekcji, a efektywnością terapii. Aby uzyskać wiarygodne wyniki, konieczne było zanalizowanie problemu na poziomie populacyjnym. W wyniku przeprowadzonych badań sprawdzono korelację pomiędzy odpowiedzią na aplikację terapii, a poziomem enzymu ALT oraz współczynnikami opisującymi różnorodność genetyczną wirusa. Następnie zdefiniowano współczynnik opisujący efektywność terapii oraz algorytm służący sprawdzaniu jak wpływają na niego różne warunki kwalifikacji do terapii. W rezultacie powinno to umożliwić lepszy dobór terapii do potrzeb konkretnego pacjenta oraz ułatwić wybór odpowiedniego momentu jej rozpoczęcia, tak aby zmaksymalizować szansę na wyleczenie i zminimalizować prawdopodobieństwo wystąpienia skutków ubocznych.

Rezultaty przedstawione w tym rozdziale są w całości wynikiem prac badawczych autora, prowadzonych pod kierownictwem promotora, ze wsparciem profesora Jacka Krawczyka. Wyniki te zostały opublikowane w pracach [WJK$^+$10, WJK$^+$09] i zaprezentowane na licznych konferencjach krajowych i międzynaro-

dowych. Konsultantami biologicznej części prac byli profesor Marek Figlerowicz oraz doktor Paulina Jackowiak z Instytutu Chemii Bioorganicznej PAN w Poznaniu.

7.1 Dane wejściowe

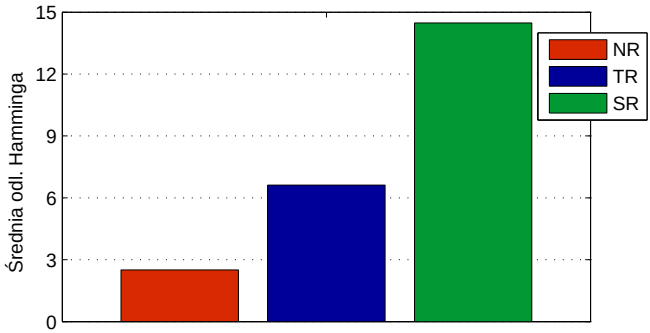
Dane wejściowe zostały zebrane w trakcie badań klinicznych przeprowadzonych przez Uniwersytet Medyczny w Poznaniu we współpracy z Instytutem Chemii Bioorganicznej PAN [KFF+05, FJA+09]. Zbiór ten składa się ze 109 pacjentów, dla których oznaczono poziom RNA wirusa we krwi w punktach czasu przełomowych dla terapii infekcji HCV (patrz sekcja 2.3.3), czyli na początku terapii oraz po 24, 48 i 72 tygodniach od jej rozpoczęcia. W rozdziale tym punkty te oznaczane będą formalnie jako $T0$, $T24$, $T48$ oraz $T72$. Niestety nie dla wszystkich pacjentów oznaczono poziom RNA wirusa w punktach $T48$ i $T72$. Dla części pacjentów zapisano jedynie informację, że wirus był w tym momencie obecny we krwi.

Dodatkowo dla 15 z powyższego zbioru pacjentów, wykonano również badania genetyczne wirusa, obliczono średnią odległość Hamminga oraz skonstruowano drzewa filogenetyczne w momencie rozpoczynania terapii. Ponieważ są to badania dużo kosztowniejsze i bardziej skomplikowane od wyznaczania poziomu RNA, liczba pacjentów, dla których je wykonano, jest zdecydowanie mniejsza. Dla tych 15 pacjentów oznaczono również wartość poziomu alatów (patrz sekcja 2.3.2).

Szczegółowe wartości poziomu RNA wirusa we krwi zostały umieszczone w dodatku w tablicy B.1, natomiast dane o średniej odległości Hamminga w tablicy B.2.

7.2 Wyniki analiz przeprowadzonych w przeszłości

Autorzy badań klinicznych zaprezentowanych w rozdziale 7.1 po ich zebraniu dokonali kilku ciekawych analiz powyższych danych [KFF+05, FJA+09]. Autorzy pod wpływem wcześniejszych doniesień [FSA+02, PGN+98] zastanawiali się jak różnorodność genetyczna wirusa wpływa na szansę wyleczenia pacjenta. W trakcie analizy zebranych danych dokonano dwóch ważnych obserwacji:

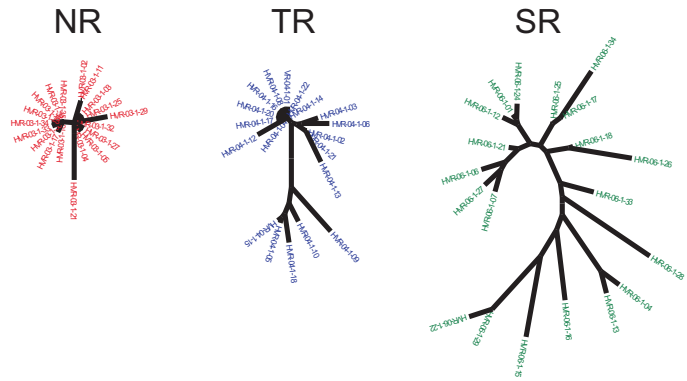


Rysunek 7.1: Przykładowe wartości średniej odległości Hamminga dla pacjentów należących do grup SR, TR i NR.

1. Spośród trzech przebadanych współczynników, których definicja znajduje się w rozdziale 2.3.3, czyli złożoności pseudotypów wirusa, średniej odległości Hamminga (MHD) i struktury drzewa filogenetycznego to te dwa ostanie najlepiej przewidują szansę pacjenta na wyleczenie.
2. Do prognozowania prawdopodobieństwa wyleczenia pacjenta najlepiej użyć sekwencji genetycznej kodującej białko E1 oraz E2.

Zgodnie z analizami przeprowadzonymi w artykule im mniejsza średnia odległość Hamminga lub bardziej skoncentrowana struktura drzewa filogenetycznego tym trudniej usunąć z organizmu wirusa za pomocą aktualnie stosowanej terapii. W zbiorze testowym pacjentów użytym przez autorów zależność ta była bardzo dobrze widoczna. Dla pacjentów tych można było precyzyjnie zdefiniować dwie wartości krytyczne średniego dystansu Hamminga - niższy, poniżej którego wszyscy pacjenci wykazywali brak odpowiedzi na leczenie (NR) oraz wyższy, powyżej którego wszyscy pacjenci wykazywali trwałą odpowiedź na leczenie (SR). Pacjenci pomiędzy niższym i wyższym progiem prezentowali odpowiedź przejściową (TR). Przykładowe wartości średniej odległości Hamminga dla pacjentów z każdej z grup SR, TR i NR przedstawione są na rysunku 7.1, natomiast przykładowe drzewa filogenetyczne prezentowane są na rysunku 7.2.

Interesującym faktem jest to, że mniejsza wartość MHD powoduje, że pacjent jest trudniejszy do wyleczenia. Intuicyjnie mogłoby się wydawać, że gdy w organizmie występuje tylko jeden, podobny wariant wirusa, to łatwiej będzie wytworzyć zwalczające go przeciwciała. W rzeczywistości jest jednak od-



Rysunek 7.2: Przykładowe drzewa filogenetyczne dla pacjentów należących do grup SR, TR i NR.

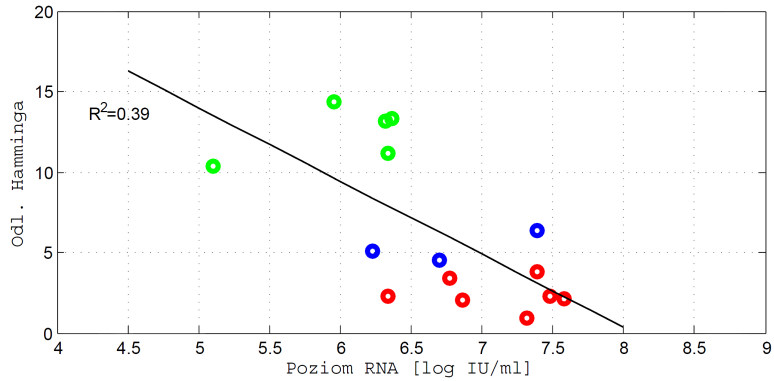
wrotnie. Prawdopodobnym wytłumaczeniem jest to, że ponieważ infekcja HCV zanim zostanie wykryta często trwa już od kilku lat, to przez ten czas wirus mógł znaleźć jeden optymalny wariant genetyczny, który gwarantuje mu przetrwanie. Natomiast w przypadku dużych odległości Hamminga poszukiwanie takiego wariantu z użyciem mechanizmów ewolucyjnych ciągle trwa.

7.3 Analiza statystyczna danych

Na podstawie badań przedstawionych w rozdziale 7.2 zdefiniowane zostały trzy grupy pacjentów, każda o innej średniej odległości Hamminga sekwencji kodującej białka E1 i E2. W celu umożliwienia bardziej czytelnej wizualizacji, każdej z tych grup przyporządkowany został inny kolor:

- pacjenci o niskiej różnorodności genetycznej (grupa czerwona) - średnia odległość Hamminga poniżej 4,5,
- pacjenci o średniej różnorodności genetycznej (grupa niebieska) - średnia odległość Hamminga pomiędzy 4,5 oraz 6,61,
- pacjenci o wysokiej różnorodności genetycznej (grupa zielona) - średnia odległość Hamminga powyżej 6,61.

Wyniki badań przedstawionych w rozdziale 7.2 pomimo iż unikalne i ważne, opierały się bardziej na obserwacjach eksperckich niż na analizie matematycz-



Rysunek 7.3: Regresja liniowa pomiędzy średnią odległością Hamminga, a poziomem RNA wirusa we krwi w chwili $T0$. Kolory reprezentują grupy pacjentów ze względu na wartość MHD.

nej. Dlatego też konstruowanie modelu prognozującego efektywność terapii rozpoczęto od dokładnej analizy statystycznej danych.

7.3.1 Korelacja pomiędzy wskaźnikami

Pierwszym krokiem było sprawdzenie korelacji pomiędzy różnorodnością genetyczną wirusa mierzoną średnią odległością Hamminga, a poziomem RNA wirusa we krwi. Wyniki prezentowane są na rysunku 7.3 w postaci wykresu regresji liniowej.

Wartość współczynnika R^2 dla regresji wynosi 0,39. Oznacza to, że około 40% zmienności wartości średniej odległości Hamminga można wyjaśnić za pomocą poziomu RNA wirusa we krwi. Nie jest to wartość bardzo wysoka, nie mniej jest uważana za wystarczająco wysoką, aby w dalszych analizach założyć istnienie korelacji pomiędzy powyższymi wartościami. Jedynym problemem jest niska liczba przypadków testowych, wynosząca tylko 15 pacjentów. Dlatego, aby upewnić się co do istotności statystycznej otrzymanego wyniku, przeprowadzono dodatkowe testy statystyczne. Model regresji liniowej został zweryfikowany przy pomocy testu F, natomiast współczynniki przy pomocy testu t, oba na poziomie ufności równym 95%. Otrzymane wyniki pokazują, że na tym poziomie ufności, w obu przypadkach, nie ma podstaw do odrzucenia hipotezy zerowej:

$$F = 9{,}72 > F_{crit} = 4{,}67 \tag{7.1}$$

oraz:

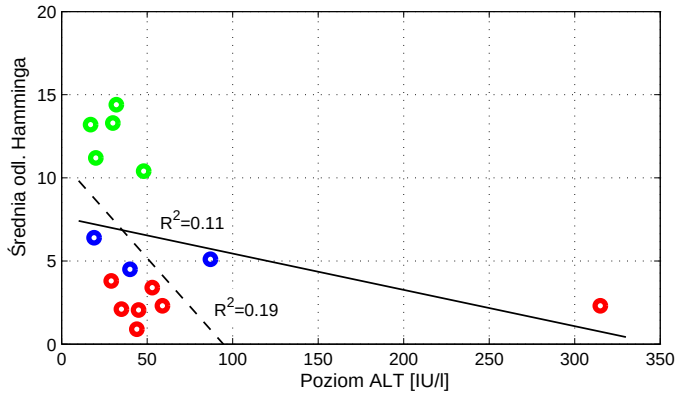
$$t = -3{,}11 < -t_{crit} = -2{,}16\,. \tag{7.2}$$

Przeprowadzona analiza statystyczna potwierdza, że ponieważ pomiędzy średnią odległością Hamminga, a poziomem RNA wirusa istnieje znacząca zależność, w dalszych rozważaniach można oprzeć się na poziomie RNA wirusa we krwi, a nie na współczynniku MHD jak to sugerują autorzy w [KFF$^+$05, FJA$^+$09]. Jest to istotny wniosek, gdyż poziom RNA wirusa jest dużo prostszy do ustalenia, a także na etapie prowadzenia tych prac posiadano dane o poziomie RNA wirusa dla zdecydowanie większej liczby pacjentów.

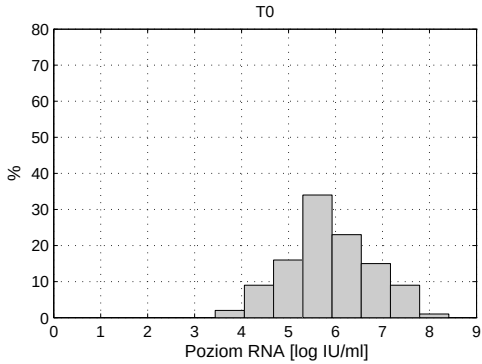
Dla porównania przeprowadzono również badanie korelacji pomiędzy różnorodnością genetyczną wirusa oraz poziomem alatów we krwi pacjenta, który można zmierzyć w sposób jeszcze prostszy niż poziom RNA wirusa. Ponieważ jeden z pacjentów (P2-4) miał kilkukrotnie wyższy poziom alatów od pozostałych uznano, że może to być błąd pomiarowy i dokonano dwóch wersji analizy — uwzględniając i nie uwzględniając powyższego pacjenta. Wyniki prezentowane są na rysunku 7.4. Wartość współczynnika R^2 wynosiła, w zależności od uwzględnienia skrajnego pacjenta, 0,11 lub 0,19. W obu przypadkach wynik jest zdecydowanie gorszy niż 0,39 dla zanalizowanego wcześniej w tej sekcji poziomu RNA, co potwierdza że wybór poziomu RNA jako wskaźnika szansy wyleczenia jest słuszny.

7.3.2 Rozkład danych

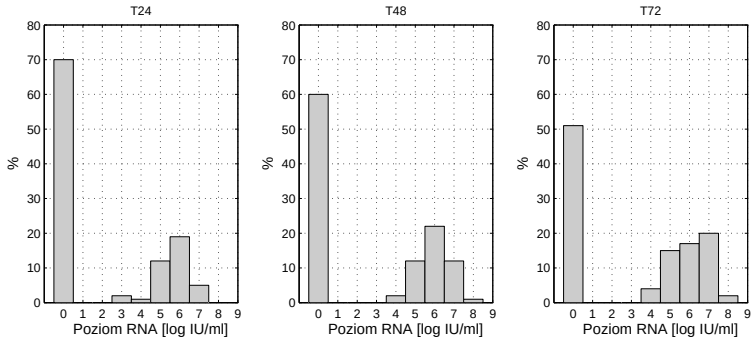
W celu skonstruowania histogramu obrazującego rozkład danych w chwili $T0$, pacjenci zostali podzieleni na 8 grup według wartości opisującej poziom RNA wirusa. Szerokość każdego przedziału wynosiła 0,62 $\frac{\log \text{IU}}{\text{ml}}$, a wynikowy histogram prezentowany jest na rysunku 7.5. Ważną cechą charakterystyczną punktu $T0$ jest to, że nie występują w nim pacjenci o zerowym poziomie wirusa HCV, ponieważ z oczywistych względów do terapii kwalifikowane są tylko osoby chore. Dla punktów $T24$ i dalszych występują już osoby o zerowym poziomie wirusa RNA (te o pozytywnej odpowiedzi na leczenie), co znacząco zwiększa różnicę wartości tego wskaźnika pomiędzy pacjentem o minimalnym (zerowym) poziomie RNA wirusa oraz pacjentem z jego maksymalną wartością. Dlatego też dla kolejnych punktów czasu zwiększono liczbę przedziałów do 10 oraz ich szerokość do $1{,}0\frac{\log \text{IU}}{\text{ml}}$. Wynikowe histogramy prezentowane są na rysunku 7.6.



Rysunek 7.4: Regresja liniowa pomiędzy średnią odległością Hamminga, a poziomem alatów we krwi w chwili $T0$. Kolory reprezentują grupy pacjentów ze względu na wartość MHD. Linia ciągła to regresja liniowa uwzględniająca wszystkich pacjentów, linia przerywana to regresja liniowa z pominięciem zdecydowanie odstającego od pozostałych pacjenta (skrajnie prawy punkt).



Rysunek 7.5: Rozkład wartości poziomu RNA wirusa w punkcie $T0$.



Rysunek 7.6: Rozkłady wartości poziomu RNA wirusa w punktach $T24$, $T48$ i $T72$.

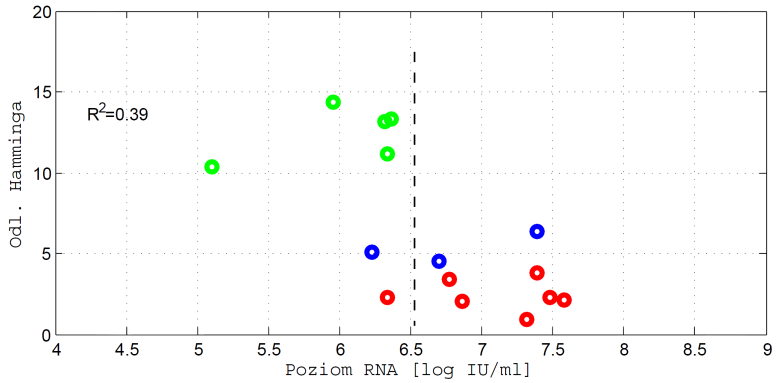
Zarówno w przypadku punktu $T0$, jak i w przypadku kolejnych punktów czasowych, po usunięciu pacjentów z zerowym poziomem RNA wirusa, rozkład wartości jest normalny. Zostało to zweryfikowane za pomocą testu Lilliforsa na poziomie ufności wynoszącym 95%. Dla punktu $T0$ średnia oraz odchylenie standardowe wynoszą odpowiednio $m = 5{,}84$ oraz $\sigma = 0{,}88$. Wartości te zostaną wykorzystane w kolejnym rozdziale do podziału pacjentów na grupy.

7.4 Analiza danych

7.4.1 Klasy zdrowia pacjentów

Na podstawie analizy rozkładu wartości poziomu RNA wirusa we krwi podzielono pacjentów na trzy grupy:

1. Grupa N, bez RNA (ang. *no RNA*) - grupa pacjentów u których poziom RNA wirusa jest niewykrywalny za pomocą metod stosowanych w czasie prowadzenia badań klinicznych, czyli poniżej $N_{max} = 2{,}43\frac{\log \text{ IU}}{\text{ml}}$. Pacjenci ci uważani są za niezainfekowanych.
2. Grupa M, ze średnim poziomem RNA (ang. *medium RNA*) - grupa pacjentów u których poziom RNA wirusa jest pomiędzy $N_{max} = 2{,}43\frac{\log \text{ IU}}{\text{ml}}$ oraz $M_{max} = 6{,}53\frac{\log \text{ IU}}{\text{ml}}$. Poziom $6{,}53\frac{\log \text{ IU}}{\text{ml}}$ został dobrany tak, aby w punkcie $T0$ jak najlepiej oddzielać pacjentów z grupy SR (z trwałą od-



Rysunek 7.7: Podział pomiędzy pacjentów w grupie M oraz H opisany szczegółowo w sekcji 7.4.1.

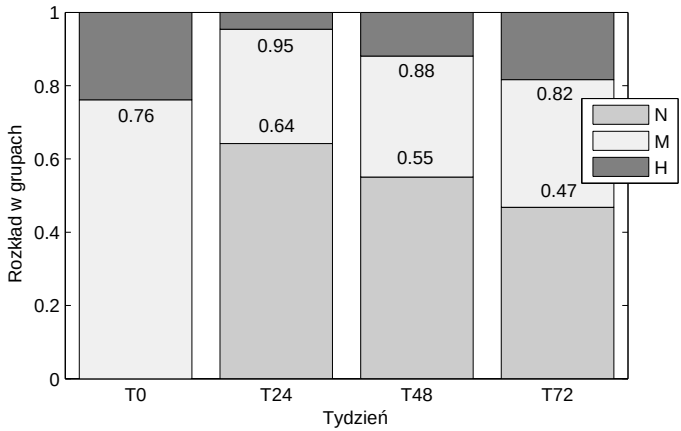
powiedzią na leczenie) od pacjentów z odpowiedzią przejściową lub jej brakiem (TR i NR), dodatkowo wartość ta jest zbliżona do $m + \sigma$. Poziom ten prezentowany jest na rysunku 7.7.

3. Grupa H, z wysokim poziomem RNA (ang. *high RNA*) - grupa pacjentów u których poziom RNA wirusa wynosi powyżej $M_{max} = 6{,}53 \frac{\log \text{ IU}}{\text{ml}}$.

Rozkład pacjentów na powyższe grupy w kolejnych punktach charakterystycznych terapii przedstawiony jest na rysunku 7.8. Na rysunku tym wyraźnie widać, że o ile po 24 tygodniach leczenia liczba pacjentów uznawanych za zdrowych wynosi ponad 60% i jest stosunkowo wysoka, to w kolejnych tygodniach wartość ta systematycznie spada. Jednocześnie pojawia się coraz większa liczba pacjentów z wysokim poziomem RNA wirusa we krwi, co sygnalizuje nawrót choroby.

7.4.2 Macierze przejść

Wykres przedstawiony na rysunku 7.8 daje dobry, ogólny pogląd na przebieg infekcji, jednak możliwość przeprowadzania na jego podstawie bardziej złożonych analiz jest ograniczona. Nie można na przykład stwierdzić, czy za przyrost pacjentów w grupie H odpowiedzialni są pacjenci z grupy M, u których z czasem nastąpił przyrost poziomu RNA wirusa, czy głównie pacjenci z grupy N, u których nastąpił nawrót choroby. Aby umożliwić przeprowadza-



Rysunek 7.8: Rozkład pacjentów w grupach w kolejnych tygodniach terapii.

nie bardziej złożonych analiz zdefiniowana została macierz przejść pomiędzy stanami:

$$\mathcal{T}_{i,j} = \begin{bmatrix} p_{N,N}^{(i,j)} & p_{N,M}^{(i,j)} & p_{N,H}^{(i,j)} \\ p_{M,N}^{(i,j)} & p_{M,M}^{(i,j)} & p_{M,H}^{(i,j)} \\ p_{H,N}^{(i,j)} & p_{H,M}^{(i,j)} & p_{H,H}^{(i,j)} \end{bmatrix} . \tag{7.3}$$

W powyższej definicji i oraz j określają tygodnie, pomiędzy którymi nastąpiło przejście ($i,j \in \{0;24;48;72\}$), natomiast $p_{g,h}^{(i,j)}$ określa prawdopodobieństwo, że pacjent będący w tygodniu i w grupie g znajdzie się w tygodniu j w grupie h ($g,h \in \{N;M;H\}$). Macierz $\mathcal{T}_{i,j}$ jest z definicji macierzą stochastyczną, a przy założeniu braku pamięci oraz skróceniu odstępów między kolejnymi punktami czasowymi $j-i$ może być macierzą przejść łańcuchu Markova modelującego rozwój infekcji HCV.

Na podstawie powyższej definicji obliczone zostały wartości elementów macierzy reprezentujących przejścia pomiędzy kolejnymi etapami terapii, czyli macierze $\mathcal{T}_{0,24}$, $\mathcal{T}_{24,48}$, $\mathcal{T}_{48,72}$ reprezentujące przejścia $T0 \to T24$, $T24 \to T48$ oraz $T48 \to T72$. Obliczone zostały również dwie dodatkowe macierze zawierające prawdopodobieństwa przejść pomiędzy początkiem terapii i jej końcem ($T0 \to T72$) oraz pomiędzy końcem pierwszej fazy oraz końcem terapii ($T24 \to T72$). Obliczone wartości elementów macierzy są następujące:

$$\mathcal{T}_{0,24} = \begin{bmatrix} - & - & - \\ 0{,}711 & 0{,}277 & 0{,}012 \\ 0{,}423 & 0{,}423 & 0{,}154 \end{bmatrix}, \tag{7.4}$$

$$\mathcal{T}_{24,48} = \begin{bmatrix} 0{,}843 & 0{,}143 & 0{,}014 \\ 0{,}029 & 0{,}765 & 0{,}206 \\ 0{,}0 & 0{,}0 & 1{,}0 \end{bmatrix}, \tag{7.5}$$

$$\mathcal{T}_{48,72} = \begin{bmatrix} 0{,}85 & 0{,}1 & 0{,}05 \\ 0{,}0 & 0{,}722 & 0{,}278 \\ 0{,}0 & 0{,}462 & 0{,}538 \end{bmatrix}, \tag{7.6}$$

$$\mathcal{T}_{0,72} = \begin{bmatrix} - & - & - \\ 0{,}566 & 0{,}337 & 0{,}096 \\ 0{,}154 & 0{,}385 & 0{,}461 \end{bmatrix}, \tag{7.7}$$

$$\mathcal{T}_{24,72} = \begin{bmatrix} 0{,}714 & 0{,}214 & 0{,}071 \\ 0{,}029 & 0{,}647 & 0{,}324 \\ 0{,}0 & 0{,}2 & 0{,}8 \end{bmatrix}. \tag{7.8}$$

Dla macierzy opisujących przejścia ze stanu z początku terapii nie można oczywiście wyliczyć prawdopodobieństw dla pierwszego wiersza, gdyż są to przejścia z grupy pacjentów z niewykrywalnym wirusem, która w punkcie $T0$ jest pusta.

Jako uzupełnienie definicji macierzy przejścia zdefiniujmy wektor wartości początkowych dla tygodnia i jako:

$$P_i = [\, P_{i,N} \; P_{i,M} \; P_{i,H} \,], \tag{7.9}$$

gdzie $P_{i,g}$ oznacza liczbę pacjentów w grupie g, w tygodniu i ($i \in \{0; 24; 48; 72\}$, $g \in \{N; M; H\}$). W opisywanym badaniu mamy na przykład:

$$P_0 = [0 \quad 83 \quad 26]. \tag{7.10}$$

Dla tak zdefiniowanego wektora, wykorzystując powyższe definicje możemy łatwo obliczyć liczbę pacjentów w każdej z grup w tygodniu j korzystając ze wzoru:

$$P_j = P_i \cdot \mathcal{T}_{i,j}. \tag{7.11}$$

7.4.3 Skuteczność terapii

Jednym z najważniejszych wskaźników służących do oceny terapii jest jej skuteczność (ang. *efficiency*). Formalnie można zdefiniować skuteczność terapii

ε jako:

$$\varepsilon = \frac{N_c}{N_t}\,, \quad \varepsilon \in [0,1]\,, \tag{7.12}$$

gdzie N_c oznacza liczbę pacjentów wyleczonych (ang. *cured*), natomiast N_t liczbę wszystkich pacjentów poddanych terapii (ang. *treated*). Zakładamy przy tym, że liczba pacjentów leczonych jest większa od 0. W przypadku terapii o potencjalnie bardzo niekorzystnych skutkach ubocznych [MH00], a przy tym nie gwarantującej wysokich szans na wyleczenie, warto zastanowić się którym pacjentom zaaplikować tą konkretną terapię tak, aby nie pogorszyć ich ogólnego stanu zdrowia. Dlatego też poniżej przedstawiona jest analiza w jaki sposób można zweryfikować skuteczność terapii dla wirusa HCV.

Jeżeli uzależnimy zakwalifikowanie pacjenta do terapii od poziomu RNA wirusa w jego krwi oznaczanego przez μ, to uszczegółowiony wzór na skuteczność terapii będzie następujący:

$$\varepsilon(\mu) = \frac{N_c}{N_t(\mu)}\,, \quad \varepsilon \in [0,1]\,, \tag{7.13}$$

gdzie $N_t(\mu)$ oznacza, że terapii poddajemy tylko pacjentów o poziomie RNA wirusa we krwi nie przewyższającym μ. W przypadku danych rozpatrywanych w tym rozdziale mamy $\mu \in [3{,}0; 8{,}5]$, a skuteczność terapii, gdy leczeni są wszyscy pacjenci, czyli μ jest maksymalne, wynosi:

$$\varepsilon(8{,}5) = 47\%. \tag{7.14}$$

Ciekawą obserwację można poczynić po dokładnym zanalizowaniu grupy pacjentów wyleczonych N_c. Okazuje się, że przy powyższych warunkach, 92% wyleczonych pacjentów pochodzi z grupy M o średnim poziomie RNA wirusa we krwi. Właśnie ta obserwacja w trakcie analizowania rezultatów badań zasugerowała, aby sprawdzić jak warunek kwalifikacji do leczenia wpływa na skuteczność terapii. Opracowano w tym celu algorytm, który zaprezentowany jest w kolejnej sekcji.

7.4.4 Wpływ początkowego poziomu RNA na skuteczność terapii

Załóżmy, że poziom wirusa w kolejnym mierzonym punkcie zależy tylko i wyłącznie od poziomu wirusa w poprzednim punkcie pomiarowym. Na podstawie wiedzy biologicznej o przebiegu infekcji HCV oraz tego, że pomiary dokonywane są stosunkowo rzadko (co 24 tygodnie) można przyjąć, że jest to

założenie realistyczne. Zauważmy również, że zgodnie z aktualnym schematem leczenia, żaden pacjent nie jest leczony pomiędzy tygodniem 48 i 72. Dlatego, jeżeli przyjmiemy powyższe założenie, można przyjąć, że macierz $\mathcal{T}_{48,72}$ określa prawdopodobieństwa zmiany stanów dla pacjenta, który nie jest leczony. Na tej podstawie można zaproponować następujący eksperyment obliczeniowy. Aktualnie wszyscy pacjenci zakwalifikowani do terapii poddawani są leczeniu, a prawdopodobieństwo wyeliminowania bądź rozwoju wirusa w początkowym okresie terapii określa macierz $\mathcal{T}_{0,24}$. Gdyby w macierzy $\mathcal{T}_{0,24}$ zamienić ostatni wiersz na wiersz pochodzący z macierzy $\mathcal{T}_{48,72}$ otrzymalibyśmy macierz określającą prawdopodobieństwa zmiany stanów pacjentów jeżeli przez pierwsze 24 tygodnie leczeni będą tylko pacjenci ze średnim poziomem RNA wirusa (należący do grupy M). Oznaczmy tak zmodyfikowaną macierz jako $\mathcal{T}'_{0,24}$. Na jej podstawie można obliczyć oczekiwany rozkład pacjentów w grupach w 72 tygodniu, korzystając ze wzoru:

$$P'_{72} = P_0 \cdot \mathcal{T}'_{0,24} \cdot \mathcal{T}_{24,48} \cdot \mathcal{T}_{48,72} = [\,43\ 41\ 25\,]\,. \tag{7.15}$$

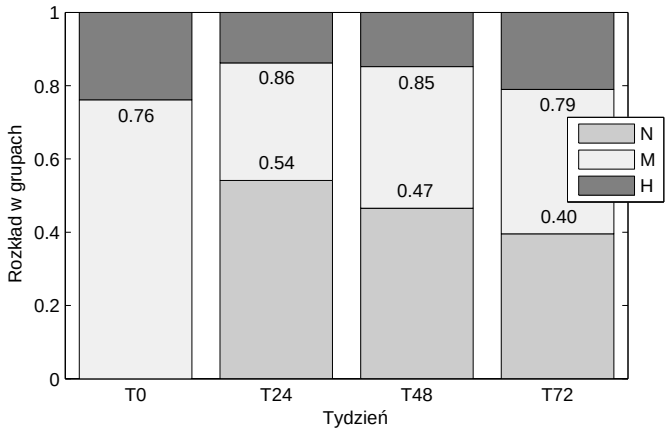
Rozkład pacjentów w grupach dla tak zmodyfikowanej terapii zaprezentowany został na rysunku 7.9. Natomiast skuteczność tak zmodyfikowanej terapii wynosi:

$$\varepsilon(M_{max}) = \varepsilon(6{,}53) = 52\%\,, \tag{7.16}$$

a więc wzrosła ona w stosunku do początkowego schematu terapii (patrz sekcja 7.4.3) o 5%.

Łatwo zauważyć, że jeżeli istnieje korelacja pomiędzy poziomem RNA wirusa, a prawdopodobieństwem wyleczenia, skuteczność terapii będzie tym większa, im niższy poziom RNA wirusa będzie wymagany do zakwalifikowania się do leczenia. Można by tą obserwację zastosować do kwalifikowania pacjentów do terapii, co w wielu przypadkach mogłoby być dla nich korzystne. Niesie to jednak poważne zagrożenie, że część pacjentów z wyższym poziomem RNA wirusa, co prawda uniknie skutków ubocznych terapii, ale również nie zostanie wyleczona, mimo że potencjalnie miałaby taką możliwość. W związku z tym, aby ocenić czy obniżanie poziomu kwalifikującego do leczenia jest korzystne dla pacjentów zaproponowano algorytm analizy zmian skuteczności terapii w zależności od poziomu RNA wirusa w momencie jej rozpoczynania oznaczony jako algorytm 7.1. W dalszej części pracy założono, że R oznacza zbiór wszystkich wartości poziomów RNA wirusa zmierzonych u pacjentów w chwili $T0$.

Funkcja *ObliczT*(i,j,M_{max}) oblicza macierz przejść pomiędzy stanami pomiędzy tygodniami i oraz j, jeżeli podział pomiędzy grupy M i H zostanie przeprowadzony według wartości M_{max} (patrz dodatek A.4). Algorytm 7.1 dla



Rysunek 7.9: Rozkład pacjentów w grupach w kolejnych tygodniach terapii prowadzonej według zmodyfikowanego schematu.

Algorytm 7.1: Algorytm obliczania wartości funckji $\varepsilon(M_{max})$ opracowany przez autorów badań.

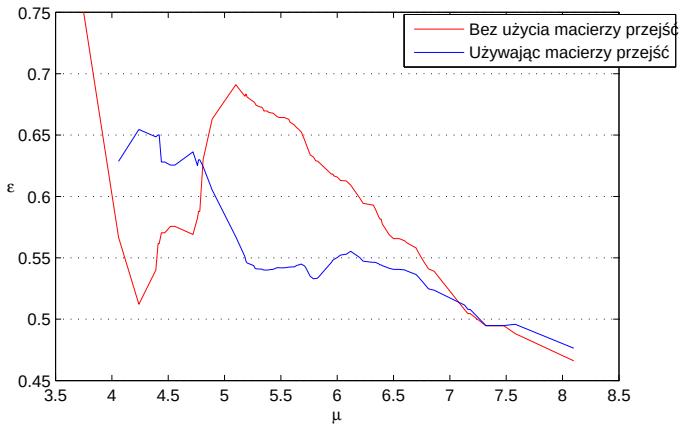
1: **foreach** $M_{max} \in R$ **do**
2: **foreach** $(i,j) \in \{(0;24);(24;48);(48;72)\}$ **do**
3: $\mathcal{T}_{i,j} = ObliczT(i,j,M_{max})$
4: **end for**
5: $\mathcal{T}_{0,24}[H,L] = \mathcal{T}_{48,72}[H,L]$
6: $\mathcal{T}_{0,24}[H,M] = \mathcal{T}_{48,72}[H,M]$
7: $\mathcal{T}_{0,24}[H,H] = \mathcal{T}_{48,72}[H,H]$
8: $P_{72} = P_0 \cdot \mathcal{T}_{0,24} \cdot \mathcal{T}_{24,48} \cdot \mathcal{T}_{48,72}$
9: $\varepsilon(M_{max}) = \frac{P_{72}[N]}{P_0[M]}$
10: **end for**

każdej wartości M_{max} będącej poziomem RNA wirusa u któregoś z pacjentów oblicza wartość $\varepsilon(M_{max})$ dla tego poziomu RNA korzystając z opisanej wcześniej metody modyfikacji macierzy $\mathcal{T}_{0,24}$. Algorytm ten został porównany z prostym algorytmem 7.2, który nie wykorzystuje macierzy $\mathcal{T}$, tylko proste przeszukanie danych wejściowych.

Funkcja $Odp(x)$ informuje jaka była odpowiedź na leczenie pacjenta, u którego zmierzono dany poziom RNA wirusa x. Algorytm 7.2 opiera się wyłącznie na sprawdzeniu liczby pacjentów posiadających pewien poziom RNA

Algorytm 7.2: Algorytmy obliczania wartości funckji $\varepsilon(M_{max})$ bez użycia macierzy przejść zdefiniowany przez autorów badań.

1: **foreach** $M_{max} \in R$ **do**
2: $P = \{x \in R : x < M_{max}\}$
3: $P_{SR} = \{x \in P : Odp(x) = SR\}$
4: $\varepsilon\left(M_{max}\right) = \frac{|P_{SR}|}{|P|}$
5: **end for**



Rysunek 7.10: Skuteczność terapii w zależności od poziomu RNA wirusa kwalifikującego do leczenia.

wirusa oraz trwałą odpowiedź na leczenie. Porównanie wyników obu algorytmów przedstawia rysunek 7.10. Rysunek ten bardzo dobrze prezentuje przewagę opracowanego algorytmu opartego o macierze przejść (niebieska linia) nad standardowym algorytmem (czerwona linia). Dla niskich poziomów RNA wirusa (poniżej 5 $\frac{\log \text{IU}}{\text{ml}}$) w zbiorze danych znajduje się bardzo mało przypadków pacjentów. Przez to dane te są bardzo podatne na zaburzenia. W rezultacie na wykresie można zaobserwować bardzo wyraźne minimum lokalne na poziomie RNA równym 4,25 $\frac{\log \text{IU}}{\text{ml}}$. Spowodowane jest ono przypadkiem, w skutek którego wśród bardzo ograniczonej liczby pacjentów posiadających poziom RNA wirusa poniżej 4,5 $\frac{\log \text{IU}}{\text{ml}}$ znalazło się kilku, którzy wykazali odpowiedź na leczenie inną niż SR. Algorytm oparty o macierze przejść, ponieważ w celu wyznaczenia skuteczności terapii wykorzystuje statystyki obliczone na podstawie

wszystkich pacjentów, jest dużo mniej podatny na powyższe zakłócenia. Jak można zaobserwować na wykresie, algorytm ten prawidłowo obrazuje zaobserwowaną wcześniej zależność, że skuteczność terapii spada wraz ze spadkiem poziomu RNA wirusa we krwi. Zachowanie takie jest również zgodne z innymi badaniami [LDK+93]. Na podstawie wykresu można uznać wartość 5,25 $\frac{\log \text{ IU}}{\text{ml}}$ jako wartość krytyczną, po przekroczeniu której skuteczność terapii jest zdecydowanie mniejsza niż dla niższych poziomów RNA wirusa. Na tej podstawie można by sugerować, że optymalna wartość M_{max} kwalifikująca pacjenta do leczenia wynosi około:

$$M^*_{max} = 5{,}25 \, \frac{\log \text{ IU}}{\text{ml}}. \tag{7.17}$$

7.5 Podsumowanie

W badaniach poprzedzających prace autora [KFF+05, FJA+09] wykazane zostało, że różnorodność genetyczna wirusa HCV we krwi pacjenta może dobrze prognozować typ odpowiedzi na leczenie. Ponieważ badanie różnorodności genetycznej jest procesem długim i kosztownym, w rozdziale tym zaproponowano i wykazano możliwość stosowania poziomu RNA wirusa we krwi jako dobrego zamiennika różnorodności genetycznej. Na tej podstawie opracowano następnie metody analizy infekcji HCV w populacji pacjentów oraz algorytm analizy skuteczności leczenia. Wykorzystując wyniki tych badań należy jednak zachować dużą ostrożność, aby wyniki analizy skuteczności terapii nie zostały wykorzystane do niezakwalifikowania do terapii z powodów ekonomicznych i organizacyjnych, a jedynie medycznych. W trakcie analiz skuteczność terapii mogłyby pojawić się argumenty za niefinansowaniem leczenia dla pacjentów mających bardzo niską szansę na wyleczenie. Jednak to nie one powinny decydować o podjęciu terapii, ale analiza potencjalnych skutków ubocznych u danego pacjenta oraz dostępność innych metod leczenia, które mogłyby w danym przypadku być bardziej skuteczne.

8
Podsumowanie

Infekcje wirusowe są bardzo ważnym i aktualnym problemem medycznym nie tylko ze względu na istniejące choroby, na które cierpią miliony osób. Dużym zagrożeniem są również nowo ewoluujące wirusy oraz nowe warianty istniejących wirusów, które uodparniają się na stosowane leczenie. Potwierdzić to mogą choćby takie wydarzenia z minionego dziesięciolecia jak pandemia tak zwanej świńskiej grypy z 2009 roku, wywołana przez szczep wirusa A/H1N1 [Cha09] oraz niebezpieczeństwo epidemii wśród ludzi spowodowanej wirusem A/H5N1, czyli tak zwaną ptasią grypą w roku 2003 [PYL+04].

W niniejszej pracy przedstawione zostały modele i algorytmy, które mogą przyczynić się do rozwoju wirusologii poprzez zaproponowanie biologom nowych, sprawniejszych narzędzi służących analizowaniu infekcji wirusowych. W tym celu w pierwszej kolejności dokonano obszernego przeglądu i porównania aktualnie stosowanych metod. Ważnym zadaniem zrealizowanym w czasie przeglądu była szczegółowa analiza jednego z popularniejszych aktualnie modeli infekcji HCV, zwanego modelem klasycznym [RDP09]. Analiza tego modelu oraz innych stosowanych narzędzi umożliwiła zdefiniowanie wad i niedogodności w aktualnie stosowanych metodach. Na ich podstawie opracowane zostały wytyczne dla dalszych badań, aby ich rezultaty były jak najbardziej przydatne w trakcie prowadzenia badań biologicznych.

Na podstawie przeprowadzonego rozpoznania oraz własnych doświadczeń ze współpracy z biologami zidentyfikowano, że jednym z głównych problemów występujących w trakcie modelowania infekcji wirusowych jest brak rozumia-

nego przez biologów języka, w którym można by je opisać. W trakcie przeprowadzania przeglądu metod do opisu modeli biologicznych potwierdzono tą diagnozę poprzez opisanie klasycznego modelu infekcji HCV w najpopularniejszych obecnie językach oraz programach służących do modelowania systemów biologicznych. Następnie na podstawie zaobserwowanych wad tych programów zaprojektowano nowy język opisu modeli biologicznych ModeLang. Ostatecznie język ten został przetestowany, a jego przydatność udowodniona w oparciu o dwa istniejące modele infekcji wirusami HIV i HCV.

Kolejna część pracy prezentuje model wieloagentowy infekcji HCV, który ma wiele zalet w stosunku do modelu klasycznego opartego o równania różniczkowe. Wraz z nim w oparciu o metodę odwróconej symulacji opracowany został algorytm wyznaczania parametrów występujących w modelu. Algorytm ten z pewnością zasługuje na uwagę, gdyż jest to pierwsza próba rozwiązania powyższego problemu dla modeli wieloagentowych stosowanych w biologii. Część pracy poświęcona symulacjom wieloagentowym kończy się opisem eksperymentu obliczeniowego, poprzez który zademonstrowane zostały możliwości zaprojektowanego modelu.

Praca kończy się prezentacją analizy statystycznej infekcji HCV w zainfekowanej populacji ludzi. Od wcześniej przeprowadzanych analiz i opracowanych faktów wyróżniają ją przede wszystkim dwa elementy. Po pierwsze przedstawiona analiza wykorzystuje zaniedbywane wcześniej wskaźniki, którymi są struktura drzewa filogenetycznego i średnia odległość Hamminga. Po drugie w niespotykany wcześniej sposób wykorzystuje macierze stochastyczne w celu modelowania i przewidywania efektywności terapii.

W trakcie prac nad doktoratem udało się zrealizować wszystkie zaprezentowane we wstępie do pracy cele. Nie oznacza to oczywiście, że nie istnieją dalsze ciekawe kierunki rozwoju. W zakresie opisywania modeli biologicznych warto na pewno zrobić szersze badania jak język ModeLang jest postrzegany w środowisku biologicznym i na tej podstawie zaproponować jego ulepszenia. Warto też zbadać możliwość jego zastosowania w innych, niż biologia, dziedzinach wiedzy. Model wieloagentowy może być dalej rozwijany, aby mógł modelować bardziej złożone systemy, a jego wydajność zwiększana. Ponieważ aktualnie przeprowadzenie pojedynczego eksperymentu obliczeniowego trwa około 6 godzin, istnieje duża potrzeba jego optymalizacji. Natomiast w przypadku badań populacyjnych na pewno najciekawszym kierunkiem dalszych badań byłoby wyznaczenie drzew filogenetycznych dla większej liczby pacjentów i zweryfikowanie dzięki temu prezentowanych wniosków na szerszej populacji ludzi. Warto również rozważyć stworzenie modelu dla infekcji wirusowej na poziomie komórkowym, zgodnie z rozgraniczeniem podanym w sekcji 3.1.

Zaprezentowane w pracy wyniki badań, wraz z potencjalnymi kierunkami

ich kontynuacji mogą z pewnością pozytywnie wpłynąć na rozwój wirusologii, nie tylko w obszarach najszerzej poruszanych w tej pracy, czyli infekcji HIV i HCV, ale również w przypadku innych wirusów. Mogą one być dużą szansą dla milionów osób chorych i zarażonych wirusami na całym świecie poprzez uchronienie ich od śmierci, poprawę ich stanu zdrowia, a nawet całkowite wyleczenie.

A Skrypty

A.1 Przykładowy model zapisany w SBML

Listing A.1: Pełny zapis w języku SBML modelu infekcji HCV opisanego w sekcji 4.1.2 zdefiniowanego przez autora pracy w celu demonstracji i analizy tego języka.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Created by libAntimony version v2.2 on 2012-10-27 13:24 with
    libSBML version 5.6.0. -->
<sbml xmlns="http://www.sbml.org/sbml/level3/version1/core" level=
    "3" version="1">
  <model id="__main" name="__main">
    <listOfCompartments>
      <compartment sboTerm="SBO:0000410" id="default_compartment"
          spatialDimensions="3" size="1" constant="true"/>
    </listOfCompartments>
    <listOfSpecies>
      <species id="Healthy_hepatocytes" compartment="
          default_compartment" hasOnlySubstanceUnits="false"
          boundaryCondition="false" constant="false"/>
      <species id="Infected_hepatocytes" compartment="
          default_compartment" hasOnlySubstanceUnits="false"
          boundaryCondition="false" constant="false"/>
      <species id="Blastic_cells" compartment="default_compartment
          " hasOnlySubstanceUnits="false" boundaryCondition="false"
```

```
            constant="false"/>
12        <species id="Dead_hepatocytes" compartment="
              default_compartment" hasOnlySubstanceUnits="false"
              boundaryCondition="false" constant="false"/>
13        <species id="Virions" compartment="default_compartment"
              hasOnlySubstanceUnits="false" boundaryCondition="false"
              constant="false"/>
14        <species id="Dead_virions" compartment="default_compartment"
              hasOnlySubstanceUnits="false" boundaryCondition="false"
              constant="false"/>
15      </listOfSpecies>
16      <listOfParameters>
17        <parameter id="rHU" constant="false"/>
18        <parameter id="rHI" constant="false"/>
19        <parameter id="s" constant="false"/>
20        <parameter id="dHU" constant="false"/>
21        <parameter id="dHI" constant="false"/>
22        <parameter id="beta" constant="false"/>
23        <parameter id="pH" constant="false"/>
24        <parameter id="cv" constant="false"/>
25        <parameter id="cI" constant="false"/>
26      </listOfParameters>
27      <listOfReactions>
28        <reaction id="HH_Reproduction" reversible="true" fast="false
              ">
29          <listOfReactants>
30            <speciesReference species="Healthy_hepatocytes"
                  stoichiometry="1" constant="true"/>
31          </listOfReactants>
32          <listOfProducts>
33            <speciesReference species="Healthy_hepatocytes"
                  stoichiometry="1" constant="true"/>
34          </listOfProducts>
35          <kineticLaw>
36            <math xmlns="http://www.w3.org/1998/Math/MathML">
37              <ci> rHU </ci>
38            </math>
39          </kineticLaw>
40        </reaction>
41        <reaction id="IH_Reproduction" reversible="true" fast="false
              ">
42          <listOfReactants>
43            <speciesReference species="Infected_hepatocytes"
                  stoichiometry="1" constant="true"/>
44          </listOfReactants>
45          <listOfProducts>
46            <speciesReference species="Infected_hepatocytes"
                  stoichiometry="1" constant="true"/>
47          </listOfProducts>
48          <kineticLaw>
49            <math xmlns="http://www.w3.org/1998/Math/MathML">
```

```
50            <ci> rHI </ci>
51          </math>
52        </kineticLaw>
53      </reaction>
54      <reaction id="BC_Upgrade" reversible="true" fast="false">
55        <listOfReactants>
56          <speciesReference species="Blastic_cells" stoichiometry=
                "1" constant="true"/>
57        </listOfReactants>
58        <listOfProducts>
59          <speciesReference species="Healthy_hepatocytes"
                stoichiometry="1" constant="true"/>
60        </listOfProducts>
61        <kineticLaw>
62          <math xmlns="http://www.w3.org/1998/Math/MathML">
63            <ci> s </ci>
64          </math>
65        </kineticLaw>
66      </reaction>
67      <reaction id="HH_Death" reversible="true" fast="false">
68        <listOfReactants>
69          <speciesReference species="Healthy_hepatocytes"
                stoichiometry="1" constant="true"/>
70        </listOfReactants>
71        <listOfProducts>
72          <speciesReference species="Dead_hepatocytes"
                stoichiometry="1" constant="true"/>
73        </listOfProducts>
74        <kineticLaw>
75          <math xmlns="http://www.w3.org/1998/Math/MathML">
76            <ci> dHU </ci>
77          </math>
78        </kineticLaw>
79      </reaction>
80      <reaction id="IH_Death" reversible="true" fast="false">
81        <listOfReactants>
82          <speciesReference species="Infected_hepatocytes"
                stoichiometry="1" constant="true"/>
83        </listOfReactants>
84        <listOfProducts>
85          <speciesReference species="Dead_hepatocytes"
                stoichiometry="1" constant="true"/>
86        </listOfProducts>
87        <kineticLaw>
88          <math xmlns="http://www.w3.org/1998/Math/MathML">
89            <ci> dHI </ci>
90          </math>
91        </kineticLaw>
92      </reaction>
93      <reaction id="Infection" reversible="true" fast="false">
94        <listOfReactants>
```

```
95          <speciesReference species="Healthy_hepatocytes"
                stoichiometry="1" constant="true"/>
96          <speciesReference species="Virions" stoichiometry="1"
                constant="true"/>
97        </listOfReactants>
98        <listOfProducts>
99          <speciesReference species="Infected_hepatocytes"
                stoichiometry="1" constant="true"/>
100       </listOfProducts>
101       <kineticLaw>
102         <math xmlns="http://www.w3.org/1998/Math/MathML">
103           <ci> beta </ci>
104         </math>
105       </kineticLaw>
106     </reaction>
107     <reaction id="Virus_emission" reversible="true" fast="false"
            >
108       <listOfReactants>
109         <speciesReference species="Infected_hepatocytes"
                stoichiometry="1" constant="true"/>
110       </listOfReactants>
111       <listOfProducts>
112         <speciesReference species="Infected_hepatocytes"
                stoichiometry="1" constant="true"/>
113         <speciesReference species="Virions" stoichiometry="1"
                constant="true"/>
114       </listOfProducts>
115       <kineticLaw>
116         <math xmlns="http://www.w3.org/1998/Math/MathML">
117           <ci> pH </ci>
118         </math>
119       </kineticLaw>
120     </reaction>
121     <reaction id="Virus_death" reversible="true" fast="false">
122       <listOfReactants>
123         <speciesReference species="Virions" stoichiometry="1"
                constant="true"/>
124       </listOfReactants>
125       <listOfProducts>
126         <speciesReference species="Dead_virions" stoichiometry="
                1" constant="true"/>
127       </listOfProducts>
128       <kineticLaw>
129         <math xmlns="http://www.w3.org/1998/Math/MathML">
130           <ci> cv </ci>
131         </math>
132       </kineticLaw>
133     </reaction>
134     <reaction id="Cure" reversible="true" fast="false">
135       <listOfReactants>
```

```
136           <speciesReference species="Infected_hepatocytes"
                  stoichiometry="1" constant="true"/>
137         </listOfReactants>
138         <listOfProducts>
139           <speciesReference species="Healthy_hepatocytes"
                  stoichiometry="1" constant="true"/>
140         </listOfProducts>
141         <kineticLaw>
142           <math xmlns="http://www.w3.org/1998/Math/MathML">
143             <ci> cI </ci>
144           </math>
145         </kineticLaw>
146       </reaction>
147     </listOfReactions>
148   </model>
149 </sbml>
```

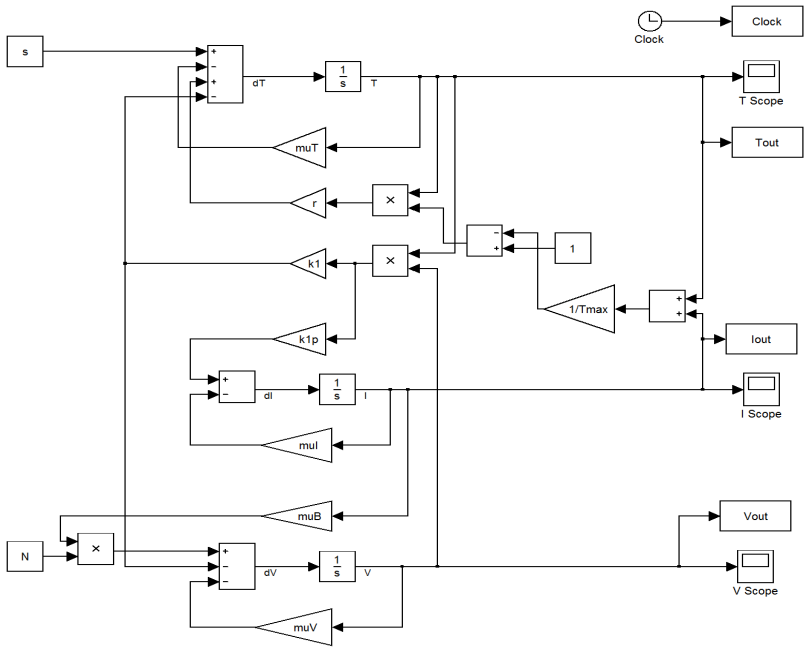
A.2 Przykładowy plik SBGN-ML

Listing A.2: Przykładowy model infekcji HCV, którego graficzna reprezentacja w języku SBGN została przygotowana przez autora i przedstawiona w sekcji 4.1.3 zapisany w formacie SBGN-ML.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<sbgn xmlns="http://sbgn.org/libsbgn/0.2">
    <map language="process description">
        <glyph id="glyph2" class="unspecified entity">
            <label text="Dead cells"/>
            <bbox y="400.0" x="396.0" h="60.0" w="108.0"/>
        </glyph>
        <glyph id="glyph4" class="association">
            <bbox y="328.0" x="458.0" h="24.0" w="24.0"/>
        </glyph>
        <glyph id="glyph5" class="association">
            <bbox y="178.0" x="628.0" h="24.0" w="24.0"/>
        </glyph>
        <glyph id="glyph6" class="association">
            <bbox y="268.0" x="268.0" h="24.0" w="24.0"/>
        </glyph>
        <glyph id="glyph7" class="association">
            <bbox y="378.0" x="298.0" h="24.0" w="24.0"/>
        </glyph>
        <glyph id="glyph8" class="association">
            <bbox y="278.0" x="158.0" h="24.0" w="24.0"/>
        </glyph>
        <glyph id="glyph0" class="unspecified entity">
            <label text="Healthy hepatocytes"/>
            <bbox y="180.0" x="80.0" h="60.0" w="180.0"/>
        </glyph>
        <glyph id="glyph1" class="unspecified entity">
            <label text="Infected hepatocytes"/>
            <bbox y="330.0" x="120.0" h="60.0" w="160.0"/>
        </glyph>
        <glyph id="glyph3" class="unspecified entity">
            <label text="Virions"/>
            <bbox y="240.0" x="436.0" h="60.0" w="108.0"/>
        </glyph>
        <arc target="glyph1" source="glyph1" id="arc0" class="
            production">
            <start y="390.0" x="158.0"/>
            <next y="410.0" x="130.0"/>
            <next y="520.0" x="160.0"/>
            <next y="450.0" x="230.0"/>
            <end y="390.0" x="210.0"/>
        </arc>
```

```
42          <arc target="glyph0" source="glyph0" id="arc1" class="
                production">
43              <start y="180.0" x="148.18182"/>
44              <next y="100.0" x="90.0"/>
45              <next y="120.0" x="260.0"/>
46              <end y="180.0" x="200.0"/>
47          </arc>
48          <arc target="glyph3" source="glyph1" id="arc2" class="
                production">
49              <start y="335.17242" x="280.0"/>
50              <end y="286.7586" x="436.0"/>
51          </arc>
52      </map>
53  </sbgn>
```

A.3 Model infekcji HIV w Mathworks Simulink



Rysunek A.1: Model infekcji HIV zaprojektowany przez autora pracy w programie Mathworks Simulink uzupełniającym środowisko Matlab.

A.4 Wyliczanie macierzy $\mathcal{T}$ (funkcja $ObliczT$)

Listing A.3: Zaimplementowany przez autorów badań kod w języku Matlab prezentujący funkcję wyliczającą macierz $\mathcal{T}$ wykorzystywaną w rozdziale 7.4.4.

```
% sort RNA_T0 into 3 groups and prepare a plot
%A = mT0-sT0;
%B = mT0+sT0;

A=2.7;
B=6.1;

kA0=1;
kB0=1;
kC0=1;
for k=1:109
    if RNA_T0(k)<A
T0A(kA0)=RNA_T0(k);
kA0=kA0+1;
    end
if (RNA_T0(k)>A) & (RNA_T0(k)<B)
T0B(kB0)=RNA_T0(k);
kB0=kB0+1;
    end
if (RNA_T0(k)>B)
T0C(kC0)=RNA_T0(k);
kC0=kC0+1;
end
end

kA24=1;
kB24=1;
kC24=1;
for k=1:109
    if RNA_T24(k)<A
T24A(kA24)=RNA_T24(k);
kA24=kA24+1;
    end
if (RNA_T24(k)>A) & (RNA_T24(k)<B)
T24B(kB24)=RNA_T24(k);
kB24=kB24+1;
    end
if (RNA_T24(k)>B)
T24C(kC24)=RNA_T24(k);
kC24=kC24+1;
end
end

```

```
kA48=1;
kB48=1;
kC48=1;
for k=1:109
    if RNA_T48r(k)<A
T24A(kA48)=RNA_T48r(k);
kA48=kA48+1;
    end
if (RNA_T48r(k)>A) & (RNA_T48r(k)<B)
T48B(kB48)=RNA_T48r(k);
kB48=kB48+1;
    end
if (RNA_T48r(k)>B)
T48C(kC48)=RNA_T48r(k);
kC48=kC48+1;
end
end

kA72=1;
kB72=1;
kC72=1;
for k=1:109
    if RNA_T72r(k)<A
T24A(kA72)=RNA_T72r(k);
kA72=kA72+1;
    end
if (RNA_T72r(k)>A) & (RNA_T72r(k)<B)
T72B(kB72)=RNA_T72r(k);
kB72=kB72+1;
    end
if (RNA_T72r(k)>B)
T72C(kC72)=RNA_T72r(k);
kC72=kC72+1;
end
end

figure
bar([kA0,kB0,kC0; kA24,kB24,kC24; kA48,kB48,kC48; kA72,kB72,kC72
    ]-1,'stacked')
```

A.5 Definicja słowników wiedzy eksperckiej

Listing A.4: Definicja XML Schema słowników wiedzy eksperckiej używanych w języku ModeLang zaimplementowana jako element przykładowego parsera tego języka.

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema id="parser"
    targetNamespace="http://tempuri.org/parser.xsd"
    elementFormDefault="qualified"
    xmlns="http://tempuri.org/parser.xsd"
    xmlns:mstns="http://tempuri.org/parser.xsd"
    xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <!-- Root element. -->
  <xs:element name="constraints">
    <xs:complexType>
      <xs:sequence>

        <!-- Each constrainsGroup is element containing unbounded
             number of synonyms for single constraint type.
             Containts ID parameter which informs about type. -->
        <xs:element maxOccurs="5" name="constraintsGroup">
          <xs:complexType>


            <xs:sequence>

              <!-- Element describing single constraint for
                   specific type of constraints. -->
              <xs:element maxOccurs="unbounded" name="constraint">
                <xs:complexType>
                  <xs:sequence>

                    <!-- This element required for every
                         constraint element. Contains string which
                         will be searched in input file. -->
                    <xs:element name="value" type="xs:string"
                         minOccurs="1" maxOccurs="1" />

                  </xs:sequence>

                </xs:complexType>
              </xs:element>

            </xs:sequence>
            <xs:attribute name="id" type="xs:integer" use="
                 required" />
```

```
36
37            </xs:complexType>
38          </xs:element>
39        </xs:sequence>
40      </xs:complexType>
41    </xs:element>
42
43    <!-- ANOTHER ELEMENT -->
44
45    <!-- Root element. -->
46    <xs:element name="keywords">
47      <xs:complexType>
48        <xs:sequence>
49
50          <!-- Keyword element contains set of parts of speech with
                unbounded number of examples. Parameter ruleType
                contains id numbers of rules for which parts of speech
                are defined. -->
51          <xs:element maxOccurs="unbounded" name="keyword">
52            <xs:complexType>
53
54
55              <xs:sequence>
56                <!-- START OF BASE -->
57                <!-- Base is set of passive voice parts of speech.
                       -->
58                <xs:element maxOccurs="1" minOccurs="0" name="base">
59                  <xs:complexType>
60                    <xs:sequence>
61
62                      <!-- Set of verbs for defined in parameter of
                             keyword element rule types. -->
63                      <xs:element maxOccurs="1" minOccurs="0" name="
                             verbs">
64                        <xs:complexType>
65                          <xs:sequence>
66
67                            <!-- Definition of single verb. Number
                                   of this elements in verbs element is
                                   unbounded. -->
68                            <xs:element maxOccurs="unbounded" name="
                                   verb">
69                              <xs:complexType>
70                                <xs:sequence>
71
72                                  <!-- Value element contains verb
                                         string. There must be exactly 1
                                          element of this type in single
                                          verb element. -->
73                                  <xs:element name="value" type="
                                         xs:string" minOccurs="1"
```

```
                                    maxOccurs="1" />
74
75                                </xs:sequence>
76                              </xs:complexType>
77                            </xs:element>
78
79                          </xs:sequence>
80                        </xs:complexType>
81                      </xs:element>
82
83                      <!-- Set of adjectives for defined in
                            parameter of keyword element rule types. --
                            >
84                      <xs:element maxOccurs="1" minOccurs="0" name="
                            adjectives">
85                        <xs:complexType>
86                          <xs:sequence>
87
88                            <!-- Definition of single adjective.
                                  Number of this elements in adjectives
                                   element is unbounded. -->
89                            <xs:element maxOccurs="unbounded" name="
                                  adjective">
90                              <xs:complexType>
91                                <xs:sequence>
92
93                                  <!-- Value element contains
                                        adjective string. There must be
                                         exactly 1 element of this type
                                         in single adjective element.
                                        -->
94                                  <xs:element name="value" type="
                                        xs:string" minOccurs="1"
                                        maxOccurs="1" />
95
96                                </xs:sequence>
97                              </xs:complexType>
98                            </xs:element>
99
100                         </xs:sequence>
101                       </xs:complexType>
102                     </xs:element>
103
104                     <!-- Set of prepositions for defined in
                            parameter of keyword element rule types. --
                            >
105                     <xs:element maxOccurs="1" minOccurs="0" name="
                            prepositions">
106                       <xs:complexType>
107                         <xs:sequence>
108
```

```
109                     <!-- Definition of single preposition.
                             Number of this elements in
                             prepositions element is unbounded. --
                             >
110                     <xs:element maxOccurs="unbounded" name="
                             preposition">
111                       <xs:complexType>
112                         <xs:sequence>
113
114                           <!-- Value element contains
                                  preposition string. There must
                                  be exactly 1 element of this
                                  type in single preposition
                                  element. -->
115                           <xs:element name="value" type="
                                  xs:string" minOccurs="1"
                                  maxOccurs="1"/>
116
117                         </xs:sequence>
118                       </xs:complexType>
119                     </xs:element>
120
121                   </xs:sequence>
122                 </xs:complexType>
123               </xs:element>
124
125               <!-- Set of nouns for defined in parameter of
                      keyword element rule types. -->
126               <xs:element maxOccurs="1" minOccurs="0" name="
                      nouns">
127                 <xs:complexType>
128                   <xs:sequence>
129
130                     <!-- Definition of single noun. Number
                            of this elements in nouns element is
                            unbounded. -->
131                     <xs:element maxOccurs="unbounded" name="
                            noun">
132                       <xs:complexType>
133                         <xs:sequence>
134
135                           <!-- Value element contains noun
                                  string. There must be exactly 1
                                   element of this type in single
                                   noun element. -->
136                           <xs:element name="value" type="
                                  xs:string" minOccurs="1"
                                  maxOccurs="1"/>
137
138                         </xs:sequence>
139                       </xs:complexType>
```

```
140                         </xs:element>
141
142                       </xs:sequence>
143                     </xs:complexType>
144                   </xs:element>
145
146                 </xs:sequence>
147               </xs:complexType>
148             </xs:element>
149             <!-- END OF BASE -->
150
151             <!-- START OF ALT -->
152             <!-- Alt is set of active voice parts of speech. -->
153             <xs:element maxOccurs="1" minOccurs="0" name="alt">
154               <xs:complexType>
155                 <xs:sequence>
156
157                   <!-- Set of verbs for defined in parameter of
                          keyword element rule types. -->
158                   <xs:element maxOccurs="1" minOccurs="0" name="
                          verbs">
159                     <xs:complexType>
160                       <xs:sequence>
161
162                         <!-- Definition of single verb. Number
                                of this elements in verbs element is
                                unbounded. -->
163                         <xs:element maxOccurs="unbounded" name="
                                verb">
164                           <xs:complexType>
165                             <xs:sequence>
166
167                               <!-- Value element contains verb
                                      string. There must be exactly 1
                                       element of this type in single
                                       verb element. -->
168                               <xs:element name="value" type="
                                      xs:string" maxOccurs="1"/>
169
170                             </xs:sequence>
171                           </xs:complexType>
172                         </xs:element>
173
174                       </xs:sequence>
175                     </xs:complexType>
176                   </xs:element>
177
178                   <!-- Set of adjectives for defined in
                          parameter of keyword element rule types. --
                          >
```

```
179                     <xs:element maxOccurs="1" minOccurs="0" name="
                            adjectives">
180                       <xs:complexType>
181                         <xs:sequence>
182
183                           <!-- Definition of single adjective.
                                  Number of this elements in adjectives
                                   element is unbounded. -->
184                           <xs:element maxOccurs="unbounded" name="
                                  adjective">
185                             <xs:complexType>
186                               <xs:sequence>
187
188                                 <!-- Value element contains
                                        adjective string. There must be
                                         exactly 1 element of this type
                                         in single adjective element.
                                        -->
189                                 <xs:element name="value" type="
                                        xs:string" maxOccurs="1"/>
190
191                               </xs:sequence>
192                             </xs:complexType>
193                           </xs:element>
194
195                         </xs:sequence>
196                       </xs:complexType>
197                     </xs:element>
198
199                     <!-- Set of prepositions for defined in
                            parameter of keyword element rule types. --
                            >
200                     <xs:element maxOccurs="1" minOccurs="0" name="
                            prepositions">
201                       <xs:complexType>
202                         <xs:sequence>
203
204                           <!-- Definition of single preposition.
                                  Number of this elements in
                                  prepositions element is unbounded. --
                                  >
205                           <xs:element maxOccurs="unbounded" name="
                                  preposition">
206                             <xs:complexType>
207                               <xs:sequence>
208
209                                 <!-- Value element contains
                                        preposition string. There must
                                        be exactly 1 element of this
                                        type in single preposition
                                        element. -->
```

```
210                                    <xs:element name="value" type="
                                           xs:string" maxOccurs="1"/>
211
212                                  </xs:sequence>
213                                </xs:complexType>
214                              </xs:element>
215
216                            </xs:sequence>
217                          </xs:complexType>
218                        </xs:element>
219
220                      </xs:sequence>
221                    </xs:complexType>
222                  </xs:element>
223                  <!-- END OF ALT -->
224                </xs:sequence>
225                <xs:attribute name="ruleType" use="required"/>
226
227              </xs:complexType>
228            </xs:element>
229          </xs:sequence>
230        </xs:complexType>
231      </xs:element>
232
233 </xs:schema>
```

A.6 Przykładowy słownik wiedzy dziedzinowej

Listing A.5: Przykładowy plik *keywords.xml* wykorzystywany w języku ModeLang do reprezentacji wiedzy dziedzinowej zdefiniowany w czasie prac nad językiem w celach testowych i demonstracyjnych. Przykład zawiera wiedzę umożliwiającą opisywanie modeli infekcji wirusowych.

```
<?xml version="1.0" encoding="utf-8"?>
<keywords xmlns="http://tempuri.org/parser.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://tempuri.org/parser.xsd parser.xsd">

  <keyword ruleType="2,3,4,5">

    <base>
      <verbs>
        <verb>
          <value>is</value>
        </verb>
        <verb>
          <value>are</value>
        </verb>
    </verbs>
    </base>

  </keyword>

  <!-- This starts new keyword section. -->

  <keyword ruleType="2,3">

    <base>
      <prepositions>
        <preposition>
          <value>by</value>
        </preposition>
      </prepositions>
    </base>

  </keyword>

  <!-- This starts new keyword section. -->

  <keyword ruleType="2,4">

    <base>
      <adjectives>
        <adjective>
```

```
42            <value>created</value>
43          </adjective>
44          <adjective>
45            <value>emitted</value>
46          </adjective>
47        </adjectives>
48      </base>
49
50    </keyword>
51
52    <!-- This starts new keyword section. -->
53
54    <keyword ruleType="2">
55      <alt>
56        <verbs>
57          <verb>
58            <value>creates</value>
59          </verb>
60          <verb>
61            <value>emits</value>
62          </verb>
63        </verbs>
64      </alt>
65
66    </keyword>
67
68    <!-- This starts new keyword section. -->
69
70    <keyword ruleType="3">
71
72      <base>
73
74        <adjectives>
75          <adjective>
76            <value>destroyed</value>
77          </adjective>
78          <adjective>
79            <value>killed</value>
80          </adjective>
81        </adjectives>
82
83
84    </base>
85
86      <alt>
87        <verbs>
88          <verb>
89            <value>destroys</value>
90          </verb>
91          <verb>
92            <value>kills</value>
```

```
93          </verb>
94        </verbs>
95      </alt>
96
97    </keyword>
98
99    <!-- This starts new keyword section. -->
100
101   <keyword ruleType="4">
102
103     <base>
104       <prepositions>
105         <preposition>
106           <value>from</value>
107         </preposition>
108       </prepositions>
109     </base>
110
111     <alt>
112       <verbs>
113         <verb>
114           <value>transforms</value>
115         </verb>
116         <verb>
117           <value>changes</value>
118         </verb>
119       </verbs>
120       <prepositions>
121         <preposition>
122           <value>to</value>
123         </preposition>
124         <preposition>
125           <value>into</value>
126         </preposition>
127       </prepositions>
128     </alt>
129
130   </keyword>
131
132   <!-- This starts new keyword section. -->
133
134   <keyword ruleType="5">
135
136     <base>
137
138       <adjectives>
139         <adjective>
140           <value>killed</value>
141         </adjective>
142         <adjective>
143           <value>disabled</value>
```

```
144         </adjective>
145         </adjectives>
146
147       </base>
148
149     <alt>
150       <verbs>
151         <verb>
152           <value>die</value>
153         </verb>
154         <verb>
155           <value>dies</value>
156         </verb>
157       </verbs>
158     </alt>
159
160   </keyword>
161
162   <!-- This starts new keyword section. -->
163
164   <keyword ruleType="6">
165
166     <base>
167
168       <verbs>
169         <verb>
170           <value>merge</value>
171         </verb>
172         <verb>
173           <value>connects</value>
174         </verb>
175         <verb>
176           <value>generates</value>
177         </verb>
178       </verbs>
179
180       <prepositions>
181         <preposition>
182           <value>to</value>
183         </preposition>
184         <preposition>
185           <value>into</value>
186         </preposition>
187         <preposition>
188           <value>the</value>
189         </preposition>
190       </prepositions>
191     </base>
192
193
194   </keyword>
```

```
195
196   <!-- This starts new keyword section. -->
197
198   <keyword ruleType="7">
199     <!-- Another type of grammar. -->
200
201     <base>
202       <nouns>
203         <noun>
204           <value>Number of</value>
205         </noun>
206         <noun>
207           <value>Sum of</value>
208         </noun>
209       </nouns>
210     </base>
211
212   </keyword>
213
214   <!-- Last keyword. -->
215
216
217 </keywords>
```

A.7 Przykładowy słownik z definicją ograniczeń

Listing A.6: Przykładowy plik *constraints.xml* wykorzystywany w języku ModeLang do reprezentacji wiedzy opisującej sposób definiowania ograniczeń zdefiniowany w czasie prac nad językiem w celach testowych i demonstracyjnych.

```
<?xml version="1.0" encoding="utf-8"?>
<constraints xmlns="http://tempuri.org/parser.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://tempuri.org/parser.xsd parser.xsd">
  <constraintsGroup id="1">
  <constraint>
    <value>at speed</value>
  </constraint>
    <constraint>
      <value>at rate</value>
    </constraint>
  </constraintsGroup>
  <constraintsGroup id="2">
    <constraint>
      <value>with probability</value>
    </constraint>
  </constraintsGroup>
  <constraintsGroup id="3">
    <constraint>
      <value>with mean time</value>
    </constraint>
  </constraintsGroup>
  <constraintsGroup id="4">
    <constraint>
      <value>less than</value>
    </constraint>
  </constraintsGroup>
  <constraintsGroup id="5">
    <constraint>
      <value>greater than</value>
    </constraint>
  </constraintsGroup>
</constraints>
```

B
Tablice uzupełniające

Tablica B.1: Poziom RNA wirusa HCV w 0, 24, 48 oraz 72 tygodniu po rozpoczęciu leczenia podany w jednostkach międzynarodowych na mililitr. '+' oznacza, że RNA zostało wykryte, ale jego dokładny poziom nie jest znany. Pacjenci, których numer rozpoczynają się od 'P' to Ci, dla których znana jest średnia odległość Hamminga populacji wirusa. Dane o MHD dla tych pacjentów zebrano w tablicy B.2. Oznaczenia SR, TR i NR oznaczają typ odpowiedzi na leczenie i zostały zdefiniowane w rozdziale 2.3.3.

Pacjent	T0	T24	T48	T72	Odpowiedź
6.	3.75	0.00	0.00	0.00	SR
68.	4.06	6.00	+	+	NR
38.	4.24	0.00	0.00	0.00	SR
74.	4.39	5.59	+	+	NR
47.	4.41	0.00	0.00	4.29	TR
51.	4.42	0.00	0.00	+	TR
22.	4.44	0.00	0.00	0.00	SR
15.	4.47	0.00	0.00	0.00	SR
45.	4.52	3.04	0.00	0.00	SR
2.	4.54	0.00	0.00	0.00	SR
43.	4.56	0.00	0.00	0.00	SR

Kontynuacja na następnej stronie...

Tablica B.1 – kontynuacja z poprzedniej strony

Patient	T0	T24	T48	T72	Response
16.	4.72	0.00	0.00	0.00	SR
41.	4.76	0.00	0.00	0.00	SR
44.	4.77	0.00	0.00	0.00	SR
70.	4.78	4.54	+	+	NR
93.	4.81	5.53	6.25	5.70	NR
23.	4.89	0.00	0.00	0.00	SR
35.	5.10	0.00	0.00	0.00	SR
P1-10	5.10	0.00	0.00	0.00	SR
5.	5.18	0.00	0.00	0.00	SR
57.	5.18	0.00	4.96	+	TR
46.	5.19	0.00	0.00	0.00	SR
24.	5.20	0.00	0.00	0.00	SR
8.	5.26	0.00	0.00	0.00	SR
66.	5.27	4.76	+	+	NR
7.	5.28	0.00	0.00	0.00	SR
13.	5.28	0.00	0.00	0.00	SR
3.	5.32	0.00	0.00	0.00	SR
21.	5.33	0.00	0.00	0.00	SR
28.	5.33	0.00	0.00	0.00	SR
19.	5.34	0.00	0.00	0.00	SR
10.	5.35	0.00	0.00	0.00	SR
55.	5.36	0.00	5.00	+	TR
87.	5.37	5.51	+	+	NR
64.	5.38	4.66	+	+	NR
17.	5.39	0.00	0.00	0.00	SR
50.	5.42	0.00	0.00	5.44	TR
1.	5.43	0.00	0.00	0.00	SR
37.	5.47	0.00	0.00	0.00	SR
48.	5.47	0.00	0.00	5.39	TR
73.	5.47	6.21	+	+	NR
86.	5.47	5.02	+	+	NR
53.	5.49	0.00	0.00	5.51	TR
26.	5.50	0.00	0.00	0.00	SR
36.	5.50	0.00	0.00	0.00	SR
72.	5.53	4.68	+	+	NR

Kontynuacja na następnej stronie...

Tablica B.1 – kontynuacja z poprzedniej strony

Patient	T0	T24	T48	T72	Response
94.	5.53	5.40	+	+	NR
76.	5.57	5.71	+	+	NR
9.	5.58	0.00	0.00	0.00	SR
18.	5.62	0.00	0.00	0.00	SR
52.	5.64	0.00	0.00	+	TR
29.	5.65	0.00	0.00	0.00	SR
30.	5.68	0.00	0.00	0.00	SR
32.	5.71	0.00	0.00	0.00	SR
34.	5.72	0.00	0.00	0.00	SR
60.	5.72	0.00	+	+	TR
42.	5.76	0.00	0.00	0.00	SR
40.	5.79	0.00	0.00	0.00	SR
79.	5.81	2.99	+	+	NR
67.	5.82	4.65	5.83	+	NR
4.	5.83	0.00	0.00	0.00	SR
78.	5.95	5.14	+	+	NR
P1-6	5.96	0.00	0.00	0.00	SR
25.	5.97	0.00	0.00	0.00	SR
39.	5.97	0.00	0.00	0.00	SR
49.	6.00	0.00	0.00	+	TR
27.	6.03	0.00	0.00	0.00	SR
63.	6.08	0.00	5.03	5.93	TR
82.	6.12	5.80	+	+	NR
12.	6.20	0.00	0.00	0.00	SR
P1-7	6.23	0.00	6.33	6.27	TR
65.	6.31	4.42	+	+	NR
77.	6.31	5.54	+	+	NR
P2-2	6.32	0.00	0.00	0.00	SR
P1-3	6.34	5.88	6.29	6.38	NR
P1-9	6.34	0.00	0.00	0.00	SR
P1-8	6.37	0.00	0.00	0.00	SR
61.	6.38	0.00	+	+	TR
75.	6.39	6.42	+	+	NR
20.	6.40	0.00	0.00	0.00	SR
85.	6.46	5.35	+	+	NR

Kontynuacja na następnej stronie...

Tablica B.1 – kontynuacja z poprzedniej strony

Patient	T0	T24	T48	T72	Response
81.	6.50	5.70	+	+	NR
89.	6.52	6.59	+	+	NR
62.	6.53	0.00	5.39	+	TR
84.	6.55	6.20	+	+	NR
31.	6.60	0.00	0.00	0.00	SR
71.	6.60	4.91	+	+	NR
14.	6.62	0.00	0.00	0.00	SR
P2-8	6.70	0.00	0.00	6.90	TR
58.	6.72	0.00	+	+	TR
88.	6.72	5.63	+	+	NR
59.	6.77	0.00	5.76	+	TR
P2-5	6.77	6.89	6.92	7.29	NR
33.	6.81	0.00	0.00	0.00	SR
80.	6.81	6.18	+	+	NR
83.	6.81	5.63	+	+	NR
P2-10	6.86	6.69	6.79	6.81	NR
90.	7.13	5.23	5.39	6.19	NR
11.	7.16	0.00	0.00	0.00	SR
56.	7.18	0.00	4.92	5.92	TR
P1-1	7.32	6.21	7.16	7.23	NR
91.	7.33	5.69	5.77	7.15	NR
54.	7.34	0.00	0.00	6.53	TR
69.	7.35	4.55	+	+	NR
P1-5	7.39	7.45	7.41	7.48	NR
P1-4	7.39	0.00	6.26	7.06	TR
P2-4	7.48	7.42	7.52	7.91	NR
P1-2	7.58	6.49	7.29	7.36	NR
92.	8.10	6.32	4.47	5.08	NR

Tablica B.2: Poziom RNA wirusa HCV we krwi w momencie rozpoczęcia terapii oraz po 24, 42 i 78 tygodniach, podany w jednostkach międzynarodowych na mililitr krwi. Poniżsi pacjenci stanowią podzbiór pacjentów wyszczególnionych w tablicy B.1.

Pacjent	MHD	RNA $\left[\frac{\log \text{ IU}}{\text{ml}}\right]$	ALT $\left[\frac{\text{IU}}{\text{l}}\right]$
P1-1	0.9	7.32	44
P1-2	2.1	7.58	35
P1-3	2.3	6.34	59
P1-5	3.8	7.39	29
P2-4	2.3	7.48	315
P2-5	3.4	6.77	53
P2-10	2.05	6.86	45
P1-4	6.4	7.39	19
P1-7	5.1	6.23	87
P2-8	4.5	6.7	40
P1-6	14.4	5.96	32
P1-8	13.3	6.36	30
P1-9	11.2	6.34	20
P1-10	10.4	5.1	48
P2-2	13.2	6.32	17

Bibliografia

[ABLS06] Bree B. Aldridge, John M. Burke, Douglas A. Lauffenburger, and Peter K. Sorger. Physicochemical modelling of cell signalling pathways. *Nat Cell Biol*, 8(11):1195–1203, Nov 2006.

[ACB99] A. Alberti, L. Chemello, and L. Benvegnu. Natural history of hepatitis C. *Journal of Hepatology*, 31 Supplement 1:17–24, 1999.

[AJL+02] Bruce Alberts, Alexander Johnson, Julian Lewis, Martin Raff, Keith Roberts, and Peter Walter. *Molecular Biology of the Cell, Fourth Edition*. Garland Science, 2002.

[AMDMV09] Gary An, Qi Mi, Joyeeta Dutta-Moscato, and Yoram Vodovotz. Agent-based models in translational systems biology. *Wiley Interdisciplinary Reviews: Systems Biology and Medicine*, 1(2):159–171, 2009.

[And12] AndroMeta LLC. *AndroMeta 2.0 User's Guide*, 2012.

[Arf85] G. Arfken. *Mathematical Methods for Physicists, 3rd ed.*, chapter Differential Equations, pages 437–496. Academic Press, Orlando, FL, 1985.

[ASU86] Alfred V. Aho, Ravi Sethi, and Jeffrey D. Ullman. *Compilers: Principles, Techniques, and Tools*. Addison Wesley, 1986.

[Bar57] N.A. Barricelli. Symbiogenetic evolution processes realized by artificial methods. *Methods*, 9:35–36, 1957.

[BC99] Jan Barciszewski and Brian F.C. Clark. *RNA Biochemistry and Biotechnology (NATO Science Partnership Subseries: 3 (closed))*. Springer, 1999.

[Ber12] Herbert Bernstein. *Regelungstechnik: Theorie und Praxis mit WinFACT und Multisim*. Elektor Verlag, 2012.

[BG07] Łukasz Bolikowski and Anna Gambin. New metrics for phylogenies. *Fundam. Inf.*, 78(2):199–216, April 2007.

[BHvR05] Rimon Barr, Zygmunt J. Haas, and Robbert van Renesse. JiST: an efficient approach to simulation using virtual machines: Research articles. *Softw. Pract. Exper.*, 35(6):539–576, May 2005.

[BOB07] C.P.D. Birch, S.P. Oom, and J.A. Beecham. Rectangular and hexagonal grids used for observation, experiment and simulation in ecology. *ecological modelling*, 206(3):347–359, 2007.

[Bro90] Rodney A. Brooks. Elephants don't play chess. *Robotics and Autonomous Systems*, 6:3–15, 1990.

[CCHK07] Anthony Campbell, Victoria Cardine, David Hines, and Stephen Kerns. *University of Michigan Chemical Engineering Process Dynamics and Controls Open Textbook*, chapter Fitting ODE parameters to data using Excel. College of Engineering, University of Michigan, 2007.

[CCL+91] C. L. Celum, R. W. Coombs, W. Lafferty, T. S. Inui, P. H. Louie, C. A. Gates, B. J. McCreedy, R. Egan, T. Grove, and S. Alexander. Indeterminate human immunodeficiency virus type 1 western blots: seroconversion risk, specificity of supplemental tests, and an algorithm for evaluation. *J Infect Dis*, 164(4):656–664, Oct 1991.

[CDH+10] Anthony L. Cunningham, Heather Donaghy, Andrew N. Harman, Min Kim, and Stuart G. Turville. Manipulation of dendritic cell function by viruses. *Curr Opin Microbiol*, 13(4):524–529, Aug 2010.

[CDK10] Krzysztof Ciesielski and Barbara Dunin-Kęplicz. *Systemy wieloagentowe: skrypt*. Uniwersytet Warszawski, 2010.

[Cen01] Centers for Disease Control and Prevention. Revised guidelines for HIV counseling, testing, and referral. *MMWR Recomm Rep*, 50(RR-19):1–57, Nov 2001.

[CH04] Bretonnel K. Cohen and Lawrence Hunter. Natural Language Processing and Systems Biology. In *Artificial Intelligence Methods and Tools for Systems Biology*, pages 147–173. Springer, 2004.

[Cha09] Margaret Chan. World now at the start of 2009 influenza pandemic. Technical report, World Health Organization, 2009.

[CKS10] Tobias Czauderna, Christian Klukas, and Falk Schreiber. Editing, validating and translating of SBGN maps. *Bioinformatics*, 26(18):2340–2341, Sep 2010.

[CM06] Stephen L. Chen and Timothy R. Morgan. The natural history of hepatitis C virus (HCV) infection. *Int J Med Sci*, 3(2):47–52, 2006.

[CNBC+06] A. Csikasz-Nagy, D. Battogtokh, K. Chen, B. Novak, and J. J. Tyson. Analysis of a generic model of eukaryotic cell cycle regulation. *Biophysical Journal*, 90:4361–4379, 2006.

[CR00] R. V. Culshaw and S. Ruan. A delay-differential equation model of HIV infection of $CD4^+$ T-cells. *Mathematical Biosciences*, 165:27–39, 2000.

[CR11] Kamila Caraballo Cortes and Marek Radkowski. The influence of hepatitis C virus (HCV) genetic varability on the outcome of antiviral therapy. *Postępy Mikrobiologii*, 50(2):131–140, 2011.

[Dah08] Ralf Dahm. Discovering DNA: Friedrich Miescher and the early years of nucleic acid research. *Hum Genet*, 122(6):565–581, Jan 2008.

[De 99] R. De Francesco. Molecular virology of the hepatitis C virus. *J Hepatol*, 31 Suppl 1:47–53, 1999.

[DEL07] N.J Dimmock, Andrew J Easton, and Keith Leppard. *Introduction to Modern Virology*. Blackwell Publishing, 6th edition, 2007.

[DGPL11] Harel Dahari, Jeremie Guedj, Alan Perelson, and Thomas Layden. Hepatitis C viral kinetics in the era of direct acting antiviral agents and interleukin-28b. *Current Hepatitis Reports*, 10:214–227, 2011. 10.1007/s11901-011-0101-7.

[DMZ+05] Harel Dahari, Marian Major, Xinan Zhang, Kathleen Mihalik, Charles M. Rice, Alan S. Perelson, Stephen M. Feinstone, and Avidan U. Neumann. Mathematical modeling of primary hepatitis C infection: noncytolytic clearance and early blockage of virion production. *Gastroenterology*, 128(4):1056–1066, Apr 2005.

[DPF09] M. M. Dabrowska, A. Panasiuk, and R. Flisiak. HCV entry as a new therapeutic target in chronic hatitits C. *Pol. Merk. Lek.*, 27(158):140–143, 2009.

[DRK09] Daniel C. Douek, Mario Roederer, and Richard A. Koup. Emerging concepts in the immunopathogenesis of AIDS. *Annu Rev Med*, 60:471–484, 2009.

[dVHL+06] Gerda de Vries, Thomas Hillen, Mark Lewis, Birgitt Schonfisch, and Johannes Muller. *A Course in Mathematical Biology: Quantitative Modeling with Mathematical and Computational (Monographs on Mathematical Modeling and Computation)*. SIAM, 2006.

[EKWR12] A. C. El Khoury, W. K. Klimack, C. Wallace, and H. Razavi. Economic burden of hepatitis C-associated diseases in the United States. *J Viral Hepat*, 19(3):153–160, Mar 2012.

[EP87] A. Engler and K. A. E. Prantl. *Die Naturlichen Pflanzenfamilien*. 1887.

[Eps06] Joshua M. Epstein. *Generative Social Science: Studies in Agent-Based Computational Modeling (Princeton Studies in Complexity)*. Princeton University Press, 2006.

[Fae11] James R. Faeder. Toward a comprehensive language for biological systems. *BMC Biol*, 9:68, 2011.

[FAKKF03] M. Figlerowicz, M. Alejska, A. Kurzynska-Kokorniak, and M. Figlerowicz. Genetic variability: the key problem in the prevention and therapy of RNA-based virus infections. *Medicinal Research Reviews*, 23:488–518, 2003.

[FBH09] James R. Faeder, Michael L. Blinov, and William S. Hlavacek. Rule-based modeling of biochemical systems with BioNetGen. *Methods Mol Biol*, 500:113–167, 2009.

[FCP+11] Alejandro F. Frangi, Jean-Louis Coatrieux, Grace C Y. Peng, David Z. D'Argenio, Vasilis Z. Marmarelis, and Anushka Michailova. Editorial: Special issue on multiscale modeling and analysis in computational biology and medicine–part-1. *IEEE Trans Biomed Eng*, 58(10):2936–2942, Oct 2011.

[FH03] A. Finney and M. Hucka. Systems biology markup language: Level 2 and beyond. *Biochem Soc Trans*, 31(Pt 6):1472–1473, Dec 2003.

[FJA+09] M. Figlerowicz, P. Jackowiak, M. Alejska, N. Malinowska, A. Kowala-Piaskowska, P. Kedziora, P. Formanowicz, J. Blazewicz, and M. Figlerowicz. Two types of viral quasispecies identified in children suffering from chronic hepatitis C. *Journal of Hepatology*, 50:S127, 2009.

[FSA+02] P. Farci, R. Strazzera, H. J. Alter, S. Farci, D. Degioannis, A. Coiana, G. Peddis, F. Usai, G. Serra, L. Chessa, Diaz G., A. Balestrieri, and R. H. Purcell. Early changes in hepatitis C viral quasispecies during interferon therapy predict the therapeutic outcome. *Proceedings of the National Academy of Sciences*, 99:3081–3086, 2002.

[GBR+99] F. Gao, E. Bailes, D. L. Robertson, Y. Chen, C. M. Rodenburg, S. F. Michael, L. B. Cummins, L. O. Arthur, M. Peeters, G. M. Shaw, P. M. Sharp, and B. H. Hahn. Origin of HIV-1 in the chimpanzee Pan troglodytes troglodytes. *Nature*, 397(6718):436–441, Feb 1999.

[Gil07] Daniel T. Gillespie. Stochastic simulation of chemical kinetics. *Annu Rev Phys Chem*, 58:35–55, 2007.

[GKP08] A. Gambin, P. Krzyżanowski, and P. Pokarowski. Aggregation algorithms for perturbed Markov chains with applications to networks modeling. *SIAM Journal on Scientific Computing*, 31(1):45–73, 2008.

[GME+03] Peter B. Gilbert, Ian W. McKeague, Geoffrey Eisen, Christopher Mullins, Aissatou Guéye-NDiaye, Souleymane Mboup, and Phyllis J. Kanki. Comparison of HIV-1 and HIV-2 infectivity from a prospective cohort study in Senegal. *Stat Med*, 22(4):573–593, Feb 2003.

[Gol80] Herman H. Goldstine. *The Computer from Pascal to von Neumann*. Princeton University Press, 1980.

[Gol89] David E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley Professional, 1989.

[GP01] Anna Gambin and Piotr Pokarowski. A new combinatorial algorithm for large markov chains (extended abstract). In *COMPUTER ALGEBRA IN SCIENTIFIC COMPUTING (CASC 2001*, pages 195–212. Springer-Verlag, 2001.

[GSG+83] R. C. Gallo, P. S. Sarin, E. P. Gelmann, M. Robert-Guroff, E. Richardson, V. S. Kalyanaraman, D. Mann, G. D. Sidhu, R. E. Stahl, S. Zolla-Pazner, J. Leibowitch, and M. Popovic. Isolation of human T-cell leukemia virus in acquired immune deficiency syndrome (AIDS). *Science*, 220(4599):865–867, May 1983.

[HBK+10] M. Hucka, F.T. Bergmann, S.M. Keating, J.C. Schaff, and L.P. Smith. *The Systems Biology Markup Language (SBML): Language Specification for Level 3 Version*, 2010.

[HFB+06] William S. Hlavacek, James R. Faeder, Michael L. Blinov, Richard G. Posner, Michael Hucka, and Walter Fontana. Rules for modeling signal-transduction systems. *Sci STKE*, 2006(344):re6, Jul 2006.

[HFS+03] M. Hucka, A. Finney, H. M. Sauro, H. Bolouri, J. C. Doyle, H. Kitano, A. P. Arkin, B. J. Bornstein, D. Bray, A. Cornish-Bowden, A. A. Cuellar, S. Dronov, E. D. Gilles, M. Ginkel, V. Gor, I. I. Goryanin, W. J. Hedley, T. C. Hodgman, J-H. Hofmeyr, P. J. Hunter, N. S. Juty, J. L. Kasberger, A. Kremling, U. Kummer, N. Le Novere, L. M. Loew, D. Lucio, P. Mendes, E. Minch, E. D. Mjolsness, Y. Nakayama, M. R. Nelson, P. F. Nielsen, T. Sakurada, J. C. Schaff, B. E. Shapiro, T. S. Shimizu, H. D. Spence, J. Stelling, K. Takahashi, M. Tomita, J. Wagner, J. Wang, and S. B. M. L Forum . The systems biology markup language (SBML): a medium for representation and exchange of biochemical network models. *Bioinformatics*, 19(4):524–531, Mar 2003.

[HH52] A. L. Hodgkin and A. F. Huxley. A quantitative description of membrane current and its application to conduction and excitation in nerve. *J Physiol*, 117(4):500–544, Aug 1952.

[Hol75] J. H. Holland. *Adaptation in Natural and Artificial Systems.* University of Michigan Press, Ann Arbor, MI, USA, 1975.

[IKI+10] Jun Itakura, Masayuki Kurosaki, Yoshie Itakura, Sinya Maekawa, Yasuhiro Asahina, Namiki Izumi, and Nobuyuki Enomoto. Reproducibility and usability of chronic virus infection model using agent-based simulation; comparing with a mathematical model. *Biosystems*, 99(1):70–78, 2010.

[Jen96] F Jensen. *An Introduction To Bayesian Networks.* CRC Press, 1996.

[JKS06] Björn H. Junker, Christian Klukas, and Falk Schreiber. VANTED: a system for advanced data analysis and visualization in the context of biological networks. *BMC Bioinformatics*, 7:109, 2006.

[KA11] Harold Klee and Randal Allen. *Simulation of Dynamic Systems with MATLAB and Simulink, Second Edition.* CRC Press, 2011.

[KB12] Franziska Klügl and Ana L. C. Bazzan. Agent-based modeling and simulation. *AI Magazine*, 33(3):29–40, 2012.

[KCS05] Abdullah Konak, David W. Coit, and Alice E. Smith. *Multi-Objective Optimization Using Genetic Algorithms: A Tutorial*, 2005.

[KFF+05] P. Kedziora, M. Figlerowicz, P. Formanowicz, M. Alejska, P. Jackowiak, N. Malinowska, A. Fratczak, J. Blazewicz, and M. Figlerowicz. Computational methods in diagnostics of chronic hepatitis C. *Bulletin of the Polish Academy of Sci.*, Tech 53:273–281, 2005.

[Kit01] H. Kitano. *Foundations of Systems Biology.* The MIT Press, 2001.

[KTN04] S. Kumar, K. Tamura, and M. Nei. MEGA3: Integrated software for molecular evolutionary genetics analysis and sequence alignment. *Briefings in Bioinformatics*, 5(2):150–163, 2004.

[KW01] J. Kim and T. Warnow. *Tutorial on Phylogenetic Tree Estimation*, 2001.

[Laz02] Yuri Lazebnik. Can a biologist fix a radio? - or, what i learned while studying apoptosis. *Cancer Cell*, 2(3):179 – 182, 2002.

[LDK+93] J. Y. N. Lau, G. L. Davis, J. Kniffen, K. P. Qian, M. S. Urdea, M. Chan, C. S. abd Mizokami, P. D. Neuwald, and J. C. Wilber. Significance of serum hepatitis C virus RNA levels in chronic hepatitis C. *Lancet*, 341:1501–1504, 1993.

[LHM+09] Nicolas Le Novere, Michael Hucka, Huaiyu Mi, Stuart Moodie, Falk Schreiber, Anatoly Sorokin, Emek Demir, Katja Wegner, Mirit I. Aladjem, Sarala M. Wimalaratne, Frank T. Bergman, Ralph Gauges, Peter Ghazal, Hideya Kawaji, Lu Li, Yukiko Matsuoka, Alice Villeger, Sarah E. Boyd, Laurence Calzone, Melanie Courtot, Ugur Dogrusoz, Tom C. Freeman, Akira Funahashi, Samik Ghosh, Akiya Jouraku, Sohyoung Kim, Fedor Kolpakov, Augustin Luna, Sven Sahle, Esther Schmidt, Steven Watterson, Guanming Wu, Igor Goryanin, Douglas B. Kell, Chris Sander, Herbert Sauro, Jacky L. Snoep, Kurt Kohn, and Hiroaki Kitano. The Systems Biology Graphical Notation. *Nat Biotechnol*, 27(8):735–741, Aug 2009.

[Lia00] T. Jake Liang. *Hepatitis C*, chapter Therapy of Chronic Hepatitis C, pages 203–239. Academic Press, 2000.

[LSX+09] Yuqiong Liang, Tuya Shilagard, Shu-Yuan Xiao, Ned Snyder, Daryl Lau, Luca Cicalese, Heidi Weiss, Gracie Vargas, and Stanley M. Lemon. Visualizing hepatitis C virus infections in human liver by two-photon microscopy. *Gastroenterology*, 137(4):1448–1458, Oct 2009.

[MAW12] Alex Mogilner, Jun Allard, and Roy Wollman. Cell polarity: Quantitative modeling as a tool in cell biology. *Science*, 336(6078):175–179, 2012.

[MCGS02] Susan L. McGovern, Emilia Caselli, Nikolaus Grigorieff, and Brian K. Shoichet. A common mechanism underlying promiscuous inhibitors from virtual and high-throughput screening. *J Med Chem*, 45(8):1712–1722, Apr 2002.

[McN10] Donald G. McNeil. Precursor to H.I.V. was in monkeys for millennia. *New York Times*, 09.16, 2010.

[Mes68] Mihajlo Mesarovic. *Systems Theory and Biology*. Springer Verlag, 1968.

[MH00] J. G. McHutchinson and J. H. Hoofnagle. *Hepatitis C*, chapter Therapy of chronic hepatitis C, pages 203–239. San Diego, California, USA, Academic Press, 2000.

[MI11] Margaret T. May and Suzanne M. Ingle. Life expectancy of HIV-positive adults: a review. *Sex Health*, 8(4):526–533, Dec 2011.

[MLF+08] Faheem Mitha, Timothy Lucas, Feng Feng, Thomas Kepler, and Cliburn Chan. The Multiscale Systems Immunology project: software for cell-based immunological simulation. *Source Code for Biology and Medicine*, 3(1):6–6, 2008.

[MOZ09] Robert Macey, George Oster, and Tim Zahnley. *Berkeley Madonna User's Guide*. University of California, Berkeley, 2009.

[MP99] J. G. McHutchison and T. Poynard. Combination therapy with interferon plus ribavirin for the initial treatment of chronic hepatitis C. *Seminars in Liver Disease*, 19 (Suppl. 1):57–65, 1999.

[MP04] S. Mottelet and A. Pauss. Xmllab : a pluridisciplinary simulation tool based on xml and scilab. In *Scilab 2004 International Conference, Rocquencourt, France, 2-3 December 2004*, 2004.

[MRU11] Carsten Maus, Stefan Rybacki, and Adelinde M. Uhrmacher. Rule-based multi-level modeling of cell biological systems. *BMC Syst Biol*, 5:166, 2011.

[MT08] Thomas Maiwald and Jens Timmer. Dynamical modeling and multi-experiment fitting with PottersWheel. *Bioinformatics*, 24(18):2037–2043, Sep 2008.

[Mur03] J.D. Murray. *Mathematical Biology II*. Springer, 2003.

[MV03] Klaus Muller and Tony Vignaux. SimPy: Simulating systems in Python. *ONLamp.com Python DevCenter*, February 2003.

[NLL+12] Tatsunori Nakano, Gillian M G. Lau, Grace M L. Lau, Masaya Sugiyama, and Masashi Mizokami. An updated analysis of hepatitis C virus genotypes and subtypes based on the complete coding region. *Liver Int*, 32(2):339–345, Feb 2012.

[OD03] Anne Op De Beeck and Jean Dubuisson. Topology of hepatitis C virus envelope glycoproteins. *Rev Med Virol*, 13(4):233–241, 2003.

[oH11] Joint United Nations Programme on HIV/AIDS. Unaids world aids day report. Technical report, United Nations, 2011.

[ONU+00] Tuncer I. Ören, S. K. Numrich, Adelinde M. Uhrmacher, Linda F. Wilson, and Erol Gelenbe. Agent-directed simulation: challenges to meet defense and civilian requirements. In *Proceedings of the 32nd conference on Winter simulation*, WSC '00, pages 1757–1762, San Diego, CA, USA, 2000. Society for Computer Simulation International.

[PBMW98] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bring order to the web. Technical report, Stanford University, 1998.

[Per89] Alan S. Perelson. Modeling the interaction of the immune system with HIV. pages 350–370, 1989.

[PGN+98] J. M. Pawlotsky, G. Germanidis, A. U. Neumann, M. Pellerin, P. O. Frainais, and D. Dhumeaux. Interferon resistance of hepatitis C virus genotype 1b: relationship to nonstructural 5A gene quasispecies mutations. *Journal of Virology*, 72:2795–2805, 1998.

[PH01] M. Pawłowska and W. Halota. Postęp w leczeniu wzw C - interferon pegylowany. *Przegl. Epidemiologiczny*, 55(supl 3):169–173, 2001.

[Pol07] Polska Grupa Ekspertów HCV. Epidemiologia HCV, 2007.

[PS10] Anna Piwonska and Franciszek Seredynski. Discovery by genetic algorithm of cellular automata rules for pattern reconstruction task. In *Proceedings of the 9th international conference on Cellular automata for research and industry*, ACRI'10, pages 198–208, Berlin, Heidelberg, 2010. Springer-Verlag.

[PYL+04] J S M. Peiris, W. C. Yu, C. W. Leung, C. Y. Cheung, W. F. Ng, J. M. Nicholls, T. K. Ng, K. H. Chan, S. T. Lai, W. L. Lim, K. Y. Yuen, and Y. Guan. Re-emergence of fatal human influenza A subtype H5N1 disease. *Lancet*, 363(9409):617–619, Feb 2004.

[RD02] Jacqueline D. Reeves and Robert W. Doms. Human immunodeficiency virus type 2. *J Gen Virol*, 83(Pt 6):1253–1265, Jun 2002.

[RDP09] Timothy C. Reluga, Harel Dahari, and Alan S. Perelson. Analysis of hepatitis C virus infection models with hepatocyte homeostasis. *SIAM Journal on Applied Mathematics*, 69(4):999–1023, 2009.

[RHS95] D. L. Robertson, B. H. Hahn, and P. M. Sharp. Recombination in AIDS viruses. *J Mol Evol*, 40(3):249–259, Mar 1995.

[Ros11] Hugo R. Rosen. Clinical practice. chronic hepatitis C infection. *N Engl J Med*, 364(25):2429–2438, Jun 2011.

[RPCH04] Andrew Rambaut, David Posada, Keith A. Crandall, and Edward C. Holmes. The causes and consequences of HIV evolution. *Nat Rev Genet*, 5(1):52–61, Jan 2004.

[SBC+05] Peter Simmonds, Jens Bukh, Christophe Combet, Gilbert Deléage, Nobuyuki Enomoto, Stephen Feinstone, Phillippe Halfon, Geneviève Inchauspé, Carla Kuiken, Geert Maertens, Masashi Mizokami, Donald G. Murphy, Hiroaki Okamoto, Jean-Michel Pawlotsky, François Penin, Erwin Sablon, Tadasu Shin-I, Lieven J. Stuyver, Heinz-Jürgen Thiel, Sergei Viazov, Amy J. Weiner, and Anders Widell. Consensus proposals for a unified system of nomenclature of hepatitis C virus genotypes. *Hepatology*, 42(4):962–973, Oct 2005.

[SBCS09] Lucian P. Smith, Frank T. Bergmann, Deepak Chandran, and Herbert M. Sauro. Antimony: a modular model definition language. *Bioinformatics*, 25(18):2452–2454, Sep 2009.

[SBZ04] Franciszek Seredynski, Pascal Bouvry, and Albert Y. Zomaya. Cellular automata computations and secret key cryptography. *Parallel Computing*, 30(5–6):753 – 766, 2004. ¡ce:title¿Parallel and nature-inspired computational paradigms and applications¡/ce:title¿.

[SC88] A. Saxon and V. Campen. AIDS: state of the art, spring 1988. *J Allergy Clin Immunol*, 81(5 Pt 1):796–802, May 1988.

[Sch02] K. Schittkowski. EASY-FIT: a software system for data fitting in dynamical systems. *Structural and Multidisciplinary Optimization*, 23:153–169, 2002.

[SCvH03] Maarten Sierhuis, William J. Clancey, and Ron van Hoof. Brahms – a multiagent modeling environment for simulating social phenomena. In *First conference of the European Social Simulation Association (SIMSOC VI), Groningen, The Netherlands*, 2003.

[See02] L. B. Seeff. Natural history of chronic hepatitis C. *Hepatology*, 36:S35–46, 2002.

[Ser97] Franciszek Seredynski. Competitive coevolutionary multi-agent systems: the application to mapping and scheduling problems. *J. Parallel Distrib. Comput.*, 47(1):39–57, November 1997.

[SH08] Vana Sypsa and Angelos Hatzakis. Modelling of viral dynamics in hepatitis B and hepatitis C clinical trials. *Stat Med*, 27(30):6505–6521, Dec 2008.

[Sho08] T. Shors. *Understanding Viruses*. Jones and Bartlett Publishers, 2008.

[Sim04] P. Simmonds. Genetic diversity and evolution of hepatitis C virus – 15 years on. *Journal of General Virology*, 85:3173–3188, 2004.

[SLB08] Yoav Shoham and Kevin Leyton-Brown. *Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations*. Cambridge University Press, 2008.

[SRAE+09] Julio Saez-Rodriguez, Leonidas G. Alexopoulos, Jonathan Epperlein, Regina Samaga, Douglas A. Lauffenburger, Steffen Klamt, and Peter K. Sorger. Discrete logic modelling as a means to link protein signalling networks with functional analysis of mammalian signal transduction. *Mol Syst Biol*, 5:331, 2009.

[Sto01] Julian Stolarczyk. Wartość ilościowej biopsji wątroby w pzw b i c. In *Warsztaty Hepatologiczne. Bielsko-Biała*, 2001.

[Str01] S.H. Strogatz. *Nonlinear dynamics and Chaos: Applications to Physics, Biology, Chemistry, and Engineering*. Perseus, 2001.

[SW03] George A. F. Seber and C. J. Wild. *Nonlinear Regression (Wiley Series in Probability and Statistics)*. Wiley-Interscience, 2003.

[Syc98] Katia P. Sycara. Multiagent systems. *AI Magazine*, 19(2):79–92, 1998.

[Ter07] Takao Terano. Exploring the Vast Parameter Space of Multi-Agent Based Simulation. In Luis Antunes and Keiki Takadama, editors, *Multi-Agent-Based Simulation VII*, volume 4442 of *Lecture Notes in Computer Science*, pages 1–14. Springer Berlin / Heidelberg, 2007.

[THT+97] Tomita, Hashimoto, Takahashi, Shimizu, Matsuzaki, Miyoshi, Saito, Tanida, Yugi, Venter, and Hutchison. E-CELL: Software environment for whole cell simulation. *Genome Inform Ser Workshop Genome Inform*, 8:147–155, 1997.

[TJ05] Joc Cing Tay and Atul Jhavar. CAFISS: a complex adaptive framework for immune system simulation. In *SAC '05: Proceedings of the 2005 ACM symposium on Applied computing*, pages 158–164, New York, NY, USA, 2005. ACM.

[TJW11] Joseph Torresi, Doug Johnson, and Heiner Wedemeyer. Progress in the development of preventive and therapeutic vaccines for hepatitis C virus. *J Hepatol*, 54(6):1273–1285, Jun 2011.

[TS10] Michael A. Tolle and Heidi L. Schwarzwald. Postexposure prophylaxis against human immunodeficiency virus. *Am Fam Physician*, 82(2):161–166, Jul 2010.

[TSB04] T. E. Turner, S. Schnell, and K. Burrage. Stochastic approaches for modelling in vivo reactions. *Comput Biol Chem*, 28(3):165–178, Jul 2004.

[WC53] J. D. Watson and F. H. Crick. Molecular structure of nucleic acids; a structure for deoxyribose nucleic acid. *Nature*, 171(4356):737–738, Apr 1953.

[Wei93] R. A. Weiss. How does HIV cause AIDS? *Science*, 260(5112):1273–1279, May 1993.

[Wil99] Uri Wilensky. Center for connected learning and computer-based modeling. Northwestern University, Evanston, IL, 1999.

[WJFB11] S. Wasik, P. Jackowiak, M. Figlerowicz, and J. Blazewicz. Modeling hcv infection using multi-agent simulation. In *Machine Learning Reports 01/2011*, Machine Learning Reports, pages 37–41, 2011.

[WJFB12] Szymon Wasik, Paulina Jackowiak, Marek Figlerowicz, and Jacek Blazewicz. Multi-agent model of HCV infection. *Artifficial Intelligence in Medicine*, in review, 2012.

[WJK+09] S. Wasik, P. Jackowiak, J. Krawczyk, P. Kedziora, P. Formanowicz, M. Figlerowicz, and J. Blazewicz. A certain model of hcv virus infection. Technical report, Institut fur Informatik, Technische Universitat Clausthal, November 2009. Proceedings of ICOLE'09: German-Polish Workshop on Computational Biology, Scheduling and Machine Learning, Lessach.

[WJK+10] S. Wasik, P. Jackowiak, J. B. Krawczyk, P. Kedziora, P. Formanowicz, M. Figlerowicz, and J. Blazewicz. Towards prediction of HCV therapy efficiency. *Computational and Mathematical Methods in Medicine*, 11(2):185–199, 2010.

[WN02] D. Wodarz and M. A. Nowak. Mathematical models of HIV pathogenesis and treatment. *Bioessays*, 24(12):1178–1187, 2002.

[Wor00] World Health Organization. Hepatitis C – global prevalence (update). *Weekly Epidemiological Record*, 75(3):18–19, 2000.

[Wot01] Garry Wotherspoon. *Who's Who in Contemporary Gay and Lesbian History : From World War II to the Present Day (Who's Who) (Vol 2)*. Routledge, 2001.

[WPB12] Szymon Wasik, Tomasz Prejzendanc, and Jacek Blazewicz. Modelang – experts-friendly language for describing viral infection models. *BMC Systems Biology*, in review, 2012.

[Wri91] A.H. Wright. Genetic algorithms for real parameter optimization. *Foundations of genetic algorithms*, 1:205–218, 1991.

[YC09] Ming-Lung Yu and Wan-Long Chuang. Treatment of chronic hepatitis C in Asia: when East meets West. *J Gastroenterol Hepatol*, 24(3):336–345, Mar 2009.

[ZKC05] GUO Zaiyi, HAN Hann Kwang, and TAY Joc Cing. Sufficiency verification of HIV-1 pathogenesis based on multi-agent simulation. In *GECCO '05: Proceedings of the 2005 conference on Genetic and evolutionary computation*, pages 305–312, New York, NY, USA, 2005. ACM.