POLITECHNIKA POZNAŃSKA
WYDZIAŁ INFORMATYKI

Tomasz PREJZENDANC

Wykorzystanie kontrolowanych języków naturalnych do modelowania systemów dynamicznych w bioinformatyce

STRESZCZENIE ROZPRAWY DOKTORSKIEJ

Promotor:
dr hab. inż. Szymon Wąsik

Poznań 2019

Modelowanie systemów dynamicznych w biologii jest silnie rozwijającą się gałęzią współczesnej nauki. Ponieważ jest to obszar interdyscyplinarny, łączy różne dziedziny naukowe, takie jak matematyka, informatyka i biologia. Matematycy i informatycy są odpowiedzialni za przetwarzanie ogromnych ilości danych i znajdowanie w nich wzorców. Z drugiej strony biologia dostarcza wyzwań naukowych, danych i pozwala zweryfikować empirycznie zaprojektowane modele teoretyczne. Niestety podczas współpracy naukowej, w której biologowie są odpowiedzialni za analizę procesów biologicznych, które mają znaleźć odzwierciedlenie w modelach teoretycznych, a matematycy i informatycy powinni budować struktury, które powinny opisywać takie procesy, bardzo często pojawiają się problemy komunikacyjne.

Biolodzy nie są w stanie skutecznie analizować formalnych opisów modeli matematycznych. Ich kluczową umiejętnością jest zrozumienie procesów biologicznych. Z drugiej strony matematycy i informatycy nie mają wystarczającej wiedzy biologicznej, która pozwalałaby na prawidłową interpretację zjawisk w formie matematycznej. Problemy te powodują, że proces modelowania jest mało skuteczny.

Rozwiązanie powyższych problemów możliwe jest między innymi poprzez opracowanie innowacyjnej metodologii modelowania. W niniejszej rozprawie opisano poszukiwanie najbardziej optymalnego podejścia do tego problemu. Jednak aby taką metodologię opracować, trzeba było osiągnąć kilka celów. Najważniejsze z nich to:

1. Klasyfikacja dostępnych metod modelowania i formatów zapisu systemów dynamicznych w bioinformatyce w celu analizy potencjału rozwoju nowych technik modelowania (rozdział **Błąd! Nie można odnaleźć źródła odwołania.**).
2. Projekt i rozwój nowego języka, który pozwoliłby modelować systemy dynamiczne w celu wyeliminowania wad obecnych rozwiązań i wykorzystania nowoczesnych technik w celu zwiększenia efektywności procesu modelowania (rozdział **Błąd! Nie można odnaleźć źródła odwołania.**).
3. Eksperymentalna weryfikacja funkcjonalności, które zapewni nowo zaproponowany język w procesie modelowania systemów dynamicznych w bioinformatyce (rozdział **Błąd! Nie można odnaleźć źródła odwołania.**).
4. Budowanie i weryfikacja środowiska pozwalającego analizować i konwertować zapisy w nowym języku do formatu SBML, który jest popularny w oprogramowaniu bioinformatycznym i biochemicznym (rozdział **Błąd! Nie można odnaleźć źródła odwołania.**).
5. Zbudowanie środowiska, które pozwala przechowywać i udostępniać dostępne modele systemów biologicznych oraz dzielić się wynikami eksperymentów naukowych (rozdział **Błąd! Nie można odnaleźć źródła odwołania.**).

Kolejne akapity opisują podsumowanie rozdziałów opisujących realizację celów wymienionych i opisanych powyżej.

Klasyfikacja metod modelowania systemów dynamicznych w bioinformatyce

Badania obecnego stanu wiedzy doprowadziły do klasyfikacji, która podzieliła kluczowe typy dostępnych metod na trzy klasy. Następne trzy paragrafy opisują zidentyfikowane klasy.

Formaty z niejawną strukturą to pierwsza klasa formatów, która została zidentyfikowana w trakcie badań. Celem tych formatów jest maksymalizacja ułatwienia przetwarzania danych przez systemy komputerowe. Typowym zastosowaniem tych formatów jest wykorzystanie ich jako typu pamięci dla

oprogramowania z graficznym interfejsem użytkownika. Oprogramowanie należące do tej klasy formatów jest obecnie najbardziej popularne i tylko w formacie SBML dostępnych jest ponad 250 narzędzi (Prejzendanc et al. 2016). Popularność formatów SBML powoduje, że wiele narzędzi zawiera funkcje umożliwiające importowanie z SBML i eksportowanie do formatu SBML. SBML został zaprojektowany w celu umożliwienia wymiany informacji w biologii systemów i pokrewnych obszarach nauki. Istnieje wiele przykładów narzędzi wykorzystujących SBML, w tym System Biology Workbench z modułem NOM, który został wdrożony w celu wykorzystania wymiany informacji między różnymi komponentami (Bergmann and Sauro 2006). Innym przykładem jest SYCAMORE (Weidemann et al. 2008), który został zaprojektowany w celu wspierania tworzenia modeli. Istnieje również program o nazwie CellDesigner (Funahashi et al. 2008), który pozwala edytować modele prezentowane na graficznym interfejsie użytkownika w postaci diagramów przedstawiających modele sieci genetycznych i biochemicznych. Innymi przykładami są SBML ODE Solver (Machné et al. 2006) oraz Systems Biology Simulation Core Library (Keller et al. 2013). Popularne są również formaty związane z inicjatywą COMBINE. Są one wspierane przez społeczność COMBINE. Przykłady takich formatów obejmują języki takie jak SBGN, SED-ML, CellML lub BioPAX (Hucka et al. 2015). Istnieją również programy, które używają dedykowanych języków, ale umożliwiają import z SBML i eksport do funkcji SBML w celu integracji z popularną społecznością. Przykłady takich programów obejmują CopasiML (Hoops et al. 2006), zaprojektowany dla symulatora COPASI, BNX wykorzystywany przez BiologicalNetworks (Baitaluk et al. 2006) lub CelleratorML (Shapiro et al. 2003), który jest zastosowaniem MathML do modelowania biologicznego.

Druga klasa, języki specjalizowane są przeznaczone do realizacji dedykowanych potrzeb technologicznych. Najczęściej są to języki dedykowane do konkretnego oprogramowania do modelowania, które zostały stworzone w celu spełnienia określonych potrzeb, czego nie można było osiągnąć za pomocą ogólnych formatów. Prowadzi to do oryginalnej charakterystyki specjalistycznych języków programowania. Przede wszystkim potrzebują specjalnego szkolenia dla użytkowników, aby zacząć z nich korzystać, ponieważ wąskie użycie takich formatów sprawia, że są one nieznane szerokiemu gronu użytkowników. Wyjątek stanowią języki używane w różnych obszarach, dla których utworzono biblioteki specjalizowane. To rozwiązanie pozwala wykorzystać powszechną wiedzę i bazę znanego i sprawdzonego standardu. Po drugie, wszystkie wyspecjalizowane języki programowania są przeznaczone dla określonego oprogramowania. Oznacza to, że języki te są edytowane i obsługiwane przez oprogramowanie, które używa ich do celów wewnętrznych. Trzecia część specjalistycznych języków programowania została zaprojektowana do określonych celów. Przykładem jest modelowanie systemów dynamicznych w biologii. Popularne jest stosowanie opisów prawa kinetycznego do definiowania reakcji między czynnikami w takich systemach. Oznacza to, że stworzenie języka, który pozwoli intuicyjnie opisać reakcje znane z praw kinetycznych, może uprościć pisanie, czytanie i interpretację takich modeli. Przykładem wyspecjalizowanych języków programowania jest Jarnac (Bergmann and Sauro 2006), który jest językiem modelowania systemów metabolicznych, który został stworzony w oparciu o BASIC. Innym przykładem jest Antymony, który został stworzony na podstawie Jarnaca. Dla symulatora JSim (Raymond et al. 2003) utworzono język skryptowy o nazwie Mathematical Model Language. PySCeS (Olivier et al. 2005) to oparty na Pythonie symulator dla systemów komórkowych, który ma swój własny język o nazwie Model Description Language. Istnieje również framework symulacyjny o nazwie NetLogo (Tisue and Wilensky 2004), który ma swój własny wbudowany język oparty na Logo, którego można używać do definiowania bardziej złożonych symulacji. Istnieją również frameworki, które pozwalają tworzyć modele oparte na znanych językach imperatywnych. Przykładami takich jest RePast (Recursive Porous Agent Simulation Toolkit) (Collier 2001), oparty na Javie. Innymi przykładami języków formalnych są SPiM (Phillips and Cardelli 2007), Bio-PEPA (Ciocchetta and Hillston 2009), który stanowi ramy do modelowania i analizy systemów biologicznych, Cyto-Sim (Sedwards and Mazza 2007), który jest językiem i symulatorem procesów biochemicznych oraz Kappa (Danos et al. 2008),

który jest językiem opartym na regułach, używanym do modelowania białek i sieci interakcji. Istnieją również możliwości zdefiniowania układów równań różniczkowych, które są często wykorzystywane w bioinformatyce. Składają się z systemów takich jak Berkley-Madonna (Macey et al. 2000), MathWorks MATLAB, a także alternatywa dla nich, którą jest Octave (Leros et al. 2010). Ponadto istnieje VCell (Loew and Schaff 2001), który pozwala bezpośrednio opisywać układy równań różniczkowych zamiast robić to za pomocą graficznego interfejsu użytkownika. Istnieje również dobry przykład specjalistycznego języka programowania o nazwie RePast, który jest strukturą umożliwiającą tworzenie skryptów modeli przy użyciu programowania imperatywnego. Został on między innymi wykorzystany w pracy Yanga (Yang Yong et al. 2007) do symulacji przenoszenia chorób zakaźnych w populacji. W tym celu badacze zastosowali indywidualny model oparty na aktywności czasoprzestrzennej. Było to możliwe dzięki integracji RePast z pakietem Java Topology Suite. Kluczową zaletą specjalistycznego języka programowania jest możliwość edycji modelu bez potrzeby korzystania z dodatkowego oprogramowania. Bezpośrednia kontrola nad modelem może być bardzo pomocna, ponieważ ręczne modyfikowanie ustawień może być znacznie szybsze i daje większą kontrolę nad efektem końcowym. W przypadku prostych formatów tekstowych zdecydowanie łatwiej jest zaimplementować skrypty do automatycznego testowania zmian w modelu. Ta klasa formatów jest zdecydowanie mniej intuicyjna niż formaty o niejawnej strukturze, którymi najczęściej zarządza graficzny interfejs użytkownika. Ponadto ręczna edycja niesie ryzyko błędów, które mogą powodować, że będzie dochodzić do błędów odczytu lub nieprawidłowych symulacji. Ważne jest również to, że używanie specjalistycznych języków programowania do zarządzania modelami składającymi się z tysięcy interakcji bez możliwości korzystania z graficznego interfejsu użytkownika może być problematyczne.

Trzecią grupą są kontrolowane języki naturalne, które są podzbiorem języków naturalnych. Ich konstrukcja zapewnia dodatkowe ograniczenia dla jednoznacznej interpretacji przez systemy komputerowe. To nowe podejście w bioinformatyce. Modelowanie za pomocą kontrolowanych języków naturalnych polega na pisaniu tekstu w oparciu o podzbiór języka naturalnego, najczęściej angielskiego. Takie podejście jest zdecydowanie bardziej elastyczne niż w poprzednich klasach wymienionych powyżej, ponieważ reguły podzbioru języka naturalnego są zdecydowanie mniej restrykcyjne niż reguły formatów komputerowych. Przygotowany model można interpretować za pomocą dedykowanego oprogramowania. Aby zrozumieć opis modelu, oprogramowanie musi zawierać specjalistyczną bazę wiedzy na temat terminów używanych w opisach modelu. Najczęściej definiuje się go w formie ontologii, czyli formalnej reprezentacji wiedzy. Większość istniejących kontrolowanych języków naturalnych można wykorzystać do przygotowania modeli bioinformatycznych. Jedynym ograniczeniem jest pozyskanie lub przygotowanie odpowiednich ontologii. Istniejące rozwiązania obejmują ACE (Fuchs and Schwitter 1996), który mocno ogranicza użycie języka angielskiego i ma już przykłady zastosowań bioinformatycznych. Innym kontrolowanym językiem naturalnym jest PENG (White and Schwitter 2009), którego ograniczenia są nawet szersze niż ACE. Innym przykładem jest CPL (Clark et al. 2010), który jest mniej restrykcyjny i wykorzystuje różne metody heurystyczne do wspierania procesu analizy. W końcu opisany w niniejszej rozprawie ModeLang (Wasik et al. 2013) jest nowym podejściem w grupie kontrolowanych języków naturalnych. Zawiera on słowniki wiedzy do wspierania modelowania systemów dynamicznych. Najważniejsze zalety kontrolowanych języków naturalnych to ich elastyczność i zdolność do opisu modelu przy użyciu różnych słów i różnych składni. W rezultacie ekspert, który opisuje model, ma możliwość wyboru słów i składni i może skupić się na opisie ważnych cech, zamiast skupiać się na zrozumieniu struktury kodu i reguł definiujących język. Jest to również uważane za ich wadę, ponieważ może powodować sytuację, w której trudniej będzie zidentyfikować podobieństwa między dwoma modelami. Kolejną ważną zaletą jest ułatwienie współpracy między specjalistami z różnych dziedzin nauki. Kontrolowane języki naturalne są ograniczeniem dla zapewnienia prawidłowej identyfikacji reguł przez systemy komputerowe. Z drugiej strony reguły są napisane w postaci podzbioru języka naturalnego, co

powoduje, że są one intuicyjne i łatwe do interpretacji. Kolejną zaletą kontrolowanych języków naturalnych jest ich zdolność do samodzielnej dokumentacji. Ta funkcja kodu jest bardzo ceniona przez inżynierów oprogramowania i jest jedną z kluczowych cech kontrolowanych języków naturalnych. Czytanie modelu napisanego przy użyciu kontrolowanych języków naturalnych przez naukowców zajmujących się opisywaną tematyką badawczą pozwala na natychmiastowe rozumienie treści co czyni modele bardziej dostępnymi dla użytkowników, którzy nie są inżynierami oprogramowania. Aby opisać kod, potrzeba znacznie mniej komentarzy i dokumentacji, ponieważ użytkownicy mogą znaleźć jego opis w podzbiorze języka naturalnego.

Projektowanie i rozwój języka ModeLang

ModeLang został stworzony w celu rozwiązywania problemów komunikacyjnych w zespołach interdyscyplinarnych i jednocześnie jako narzędzie do symulacji systemów dynamicznych. Zaprojektowanie języka było związane z poszukiwaniem narzędzia, które umożliwi intuicyjne modelowanie i nie wymaga dużo czasu na zdobycie niezbędnej wiedzy, czy to w dziedzinie modelowania matematycznego czy informatyki.

Głównym założeniem podczas projektowania nowego języka było zaproponowanie rozwiązania, które byłoby całkowicie intuicyjne, tak aby koszty czasu, który muszą ponieść naukowcy uczący się nowych rozwiązań w zakresie modelowania, został maksymalnie ograniczony. Z tej cechy języka ModeLang wynikała inna, mianowicie powinien on być jak najbardziej elastyczny. W tym celu zaproponowano szereg funkcji opisanych w tym rozdziale. Przede wszystkim wszystkie opisy zachowań i interakcji zostały napisane w formie reguł odpowiadających zdaniom znanym z języków naturalnych, takich jak na przykład angielski. Wszystkie opisy mogą być budowane w ModeLang zarówno w stronie czynnej, jak i biernej, dzięki czemu naturalność opisów nie jest ograniczona. Kolejnym elementem jest wykrycie powtarzalnych cech zdania, które można zignorować ze względu na brak wartości merytorycznej, na przykład zakończenie zdania kropką nie wpływa na rozpoznanie ostatniego słowa użytego w opisie reguły.

ModeLang to kontrolowany język naturalny. Oznacza to, że jego zapis jest podzbiorem języka naturalnego. Składa się z części, które można jednoznacznie interpretować na podstawie składni i słownictwa. Istnieje również deklaratywna część języka, która jest zapisem deklaracji wartości parametrów, podobnie do deklaracji znanych z języków empirycznych. Możliwe jest zbudowanie modelu bez części deklaratywnej, ale nie jest możliwe wykonywanie obliczeń i symulacji bez podania później wartości parametrów, na podstawie których należy wykonać obliczenia. Część języka, w której opisano modele w postaci kontrolowanego języka naturalnego, składa się z reguł.

Część deklaratywna pozwala przypisywać wartości. Elementy języka, które można zdefiniować, to początkowe wartości wielkości populacji i parametry. Parametry mogą określać szybkość reakcji, czas opóźnienia reakcji, parametr reakcji warunkowej i pojemność środowiska. Wszystkie deklaracje można zdefiniować przypisując im wartość. W przypadku parametrów można również wskazać zakresy wartości, aby umożliwić ich późniejsze oszacowanie na podstawie danych eksperymentalnych.

Część opisująca model pozwala określić, jakie interakcje występują między agentami w opisywanym modelu. Reguły pozwalają zapisywać modele przy użyciu stron czynnych i biernych. Ponadto pozwalają kończyć zdania dowolną liczbą sufiksów, czyli częścią zdania podaną na końcu reguły określającą szybkość reakcji, warunkującą wydajność reakcji w zależności od stanu i określającą opóźnienie w wykonaniu reakcji.

ModeLang składa się obecnie z dziesięciu reguł, w tym dwóch o specyficznym charakterze. Reguła 6 określa pojemność środowiska. Reguła 7 określa potencjalny stan agenta. Oprócz reguł 6 i 7 pozostałe reguły określają interakcje między agentami tego modelu. Dwie ostatnie reguły to tak zwane reguły „wiele do wielu", które pozwalają na nieograniczone zapisywanie modelu pod względem liczby zarówno reagentów jak i produktów.

Każda reguła opisująca interakcję między agentami wymaga określenia szybkości odpowiedzi. Nie można powiedzieć, że na przykład jeden agent żeruje na drugim, jeśli skala takiego zachowania nie jest ograniczona, ponieważ obliczenia nie będą możliwe. Dlatego jednym z trzech typów wyrażeń kończących regułę są definicje prędkości reakcji. Niektóre reakcje, oprócz szybkości wykonania, charakteryzują się również zależnością wykonania od stanu, w którym aktualnie znajduje się system. W ModeLang można zapisać takie reakcje w formie reguł specyficznych dla warunków. Są też reakcje opóźnione. Opisanie reakcji z opóźnieniem w języku ModeLang jest możliwe przy użyciu sufiksu reguły ModeLang, który odpowiada za określenie opóźnienia w wykonaniu reakcji opisanych w danej regule. Pełna lista reguł obsługiwana przez język ModeLang jest następująca:

- Regułę 1 można wykorzystać do opisania utworzenia nowego agenta.
- Regułę 2 można wykorzystać do opisania zniszczenia agenta przez innego agenta.
- Reguła 3 opisuje przekształcenie agenta w innego agenta.
- Reguła 4 opisuje śmierć agenta.
- Reguła 5 opisuje połączenie dwóch różnych agentów w trzeciego.
- Reguła 6 opisuje limity wzrostu dla jednostek wskazanego agenta lub agentów razem, zarówno w minimalnej, jak i maksymalnej liczbie.
- Reguła 7 opisuje potencjalny stan, w którym może znajdować się agent.
- Z reguły 8 można korzystać, gdy agenty są dostarczane do modelowanego systemu z zewnątrz lub powstają w środowisku bez potrzeby interakcji z innymi agentami.
- Reguła 9 została stworzona, aby umożliwić modelowanie interakcji dla dowolnego zestawu reagentów i produktów reakcji, czyli czynników, które reagują i czynników, które otrzymujemy w wyniku reakcji (stąd nazwa pomocnicza kreatywnej reguły N:N).
- Reguła 10 stanowi rozwinięcie reguły 9 do zastosowań destrukcyjnych.

Ponadto istnieje kilka rodzajów wyrażeń, których można użyć do uzupełnienia definicji modelu:

- tempo wykonywania reakcji,
- reguły warunkowe,
- możliwości środowiskowe,
- reakcje z opóźnieniem.

Weryfikacja eksperymentalna języka ModeLang

Po zaprojektowaniu języka ModeLang i wdrożeniu środowiska eksperymentalnego stało się możliwe sprawdzenie poprawności języka i zapisanych w nim modeli. Środowisko zostało zbudowane w formie aplikacji internetowej, która umożliwiła weryfikację języka ModeLang z dowolnego komputera za pośrednictwem przeglądarki internetowej. Została również stworzona integracja z narzędziami zewnętrznymi, umożliwiając przekształcenie ModeLang do formatu SBML. Ponadto utworzono konwerter modeli w celu prezentacji układu równań różniczkowych, który pozwala sprawdzić, czy wprowadzone modele są zgodne z ich znaną reprezentacją matematyczną opracowaną dla najpopularniejszych modeli biologicznych. Po utworzeniu środowiska konieczne było zbudowanie odpowiedniego zestawu modeli testowych, które umożliwiłyby przetestowanie rozwiązania. Model testowy rozumiany

jest tutaj jako słowny opis modelowanego systemu i jego właściwego modelu matematycznego służącego do weryfikacji wyników uzyskanych w języku ModeLang. W tym celu stworzono dwie klasyfikacje modeli: tematyczne i oparte na złożoności. Pierwszy pozwala zweryfikować jedno z pierwszych założeń ModeLang, takie jak intuicyjność i uniwersalność języka. Druga stworzona została w celu sprawdzenia czy notacje matematyczne o różnych poziomach złożoności można przenieść na obecną formę języka, czy też konieczne będzie jego rozszerzenie. Kolejnym ważnym elementem weryfikacji był zestaw prostych i skomplikowanych modeli, aby dozować poziom trudności użytkownikom biorącym udział w eksperymencie.

Najpierw przeprowadzono eksperyment, którego celem była weryfikacja podstawowych założeń i możliwości związanych z zapisywaniem modeli językowych ModeLang w celu weryfikacji możliwości samego języka. Rezultatem tych działań była seria sugestii ulepszeń i zmian, które zostały następnie wdrożone i przygotowane do następnego eksperymentu. Drugim eksperymentem było sprawdzenie poprawności naukowej zbudowanych modeli. Wykorzystano w nim takie elementy, jak podgląd formy układów równań różniczkowych i konwersję do języka SBML. Przekształcone modele zostały przeanalizowane w CellDesigner, gdzie można było je również zasymulować. W rezultacie powstał kompletny proces, który obejmował weryfikację eksperymentalną od etapu budowy modelu do jego ostatecznej weryfikacji, co pozwoliło na przetestowanie języka i związanych z nim założeń.

Budowanie zapisu w języku ModeLang jest ważną częścią procesu modelowania. Cały proces składa się z kilku faz. Utworzenie zapisu w ModeLang pozwala zautomatyzować przejście między fazą, w której przygotowywany jest model biologiczny, a fazą modelu matematycznego. Jest to ważna zmiana, ponieważ model biologiczny jest opisem rzeczywistych zjawisk i opiera się na obserwacjach empirycznych. Dlatego jest budowany przez specjalistów w dziedzinie biologii. Model matematyczny jest zapisem, na podstawie którego można wykonać obliczenia. Wymaga dalszego rozwoju poprzez uzupełnienie go o wartości parametrów i wielkości populacji agentów, jednak stanowi solidną podstawę do eksperymentalnych prac obliczeniowych. Budowa języka ModeLang miała na celu rozwiązanie problemów komunikacyjnych między specjalistami przygotowującymi model biologiczny i model matematyczny. Dlatego kluczowym aspektem weryfikacji poprawności eksperymentalnej ModeLang było sprawdzenie, czy przejście między modelami biologicznymi i matematycznymi odbywa się prawidłowo. Ze względu na fakt, że jest to praca interdyscyplinarna, weryfikacja została przeprowadzona przy użyciu specjalistów w tej dziedzinie, którzy są odpowiedzialni za codzienne budowanie modeli biologicznych. Na podstawie ich wiedzy na temat przeprowadzanych przez nich zjawisk biologicznych możliwa była weryfikacja poprawności modelu, który powinien odzwierciedlać zjawiska obserwowane w naturze. Ponadto zastosowane modele były znane i naukowo zweryfikowane, więc ich postać matematyczna była znana przed rozpoczęciem eksperymentu. To pozwoliło zweryfikować zapis matematyczny, otrzymany w wyniku działania narzędzi zbudowanych dla eksperymentalnej weryfikacji środowiska języka ModeLang.

Po sprawdzeniu poprawności formy matematycznej uzyskanej podczas drugiego eksperymentu można było skupić się na kolejnym kroku, jakim jest zbudowanie modelu statystycznego. Prawidłowość opisu zjawisk opiera się nie tylko na zastosowaniu odpowiednich notacji matematycznych, ale ważne są również parametry obliczeniowe, które mogą znacząco wpłynąć na przebieg obliczeń. Mając świadomość, że postać matematyczna modelu jest poprawnie zbudowana, można było eksperymentować z zastosowanymi parametrami, aby uzyskać prawidłowy model statystyczny, który zgodnie ze schematem modelowania opisanym przez Harveya Thomasa Banksa jest podstawą do porównania wyników z zachowaniami obserwowanymi w świecie rzeczywistym. Dlatego eksperymenty nie tylko potwierdziły prawidłowość procesu budowania modeli matematycznych przy użyciu języka ModeLang, ale także pozwoliły na skuteczną analizę wpływu wybranych parametrów w modelu statystycznym, aby

jak najlepiej odzwierciedlić rzeczywiste zjawiska. Po zastosowaniu odpowiednich parametrów, które zostały zaczerpnięte z publikacji opisujących prace badawcze nad wskazanymi modelami, możliwe było osiągnięcie oczekiwanych wyników, które potwierdziły poprawność interpretacji zapisu utworzonego w języku ModeLang.

Konwersja ModeLang do SBML

Projektowanie języka ModeLang od samego początku było powiązane z planem, aby umożliwić użytkownikom wykonywanie symulacji i obliczeń w oparciu o modele tworzone przy użyciu tego języka. Na pierwszym etapie implementacji autor tej rozprawy opracował język jako bibliotekę dedykowaną dla symulatora, która została stworzona do wykonywania obliczeń modelu, zapisanego w postaci układów równań różniczkowych. Kiedy udało się zbudować pierwszą wersję języka, rozpoczęto testy i ulepszenie rozwiązania. Autor tej rozprawy zaczął budować platformę zawierającą takie narzędzia, jak edytor modeli zapisanych w języku ModeLang, analizator składni i narzędzia komunikacyjne, które umożliwiają wykonywanie pracy za pomocą aplikacji sieciowych i zapisywanie modeli w bazie danych za jej pomocą.

Podczas badań jednym z obszarów była opisana wcześniej analiza dostępnych rozwiązań oraz weryfikacja ich przydatności i możliwości wymiany informacji między nimi. Podczas badań zaobserwowano, że kluczowym elementem łączącym większość narzędzi był format SBML. Wśród programów obsługujących SBML znalazły się najpopularniejsze aplikacje bioinformatyczne. Dlatego pojawił się pomysł, że integracja języka ModeLang z narzędziami zewnętrznymi może być wykonana przez zapewnienie mechanizmu konwersji ModeLang do formatu SBML.

Prace związane z przekształceniem ModeLang do SBML dotyczyły trzech obszarów, które wymagały weryfikacji. Przede wszystkim ważne było, jak przechowywać główne elementy ModeLang w SBML takie jak agenty, reguły ModeLang i wartości parametrów. Zapis wartości parametrów musiał być wykonany w sposób, który z matematycznego punktu widzenia zachowałby logikę ich wpływu na reakcje, które z ich pomocą musiały zostać wykonane. Kolejnym elementem było określenie, które elementy ModeLang nie mogą zostać zapisane w SBML i czy wpływa to na poprawność modeli przekonwertowanych na SBML. Istniały również elementy, których zapis był ważny dla poprawności modelu lub spójności dostarczonych informacji, które zostały uzupełnione po stronie języka ModeLang. Dla tych elementów językowych, które nie były kluczowe dla poprawności definicji modelu, zdefiniowano wartości domyślne. Zdefiniowano również domyślną strukturę wynikowego języka SBML, który odzwierciedla grupę informacji, które przekazujemy z języka ModeLang. Zdefiniowanie domyślnej struktury było ważne, ponieważ SBML musiał oprzeć wymaganą treść na informacjach przekazywanymi z zapisów w języku ModeLang.

Podstawową informacją niezbędną do konwersji była informacja o agentach, reakcjach między agentami i parametrach opisujących zachodzące reakcje. Informacje te mogły zostać bezpośrednio przekształcone. Druga grupa to informacje, których nie można zapisać w SBML, i przechowujemy je w ModeLang. Najważniejsze były zakresy wartości parametrów. W ModeLang możemy zapisać wartość parametru bezpośrednio lub jako zakres wartości. Pozwala to następnie na dopasowywanie wartości parametrów na podstawie załadowanych danych laboratoryjnych przy użyciu oprogramowania, takiego jak COPASI (Hoops et al. 2006). SBML nie pozwala na przechowywanie informacji o parametrach w postaci zakresów wartości. Inną informacją, której nie jesteśmy w stanie wykorzystać jest pojemność środowiskowa. Prawdopodobieństwo zachodzenia reakcji to kolejny aspekt. Istnieje ograniczenie prędkości reakcji, które opisuje prawdopodobieństwo zdarzenia. Z powodu braku jednoznacznych sposobów zapisania tego rodzaju ograniczenia prawdopodobieństwo zapisywane jest jako parametr, który

następnie podlega operacjom matematycznym w zależności od wartości prawdopodobieństwa, w sposób podobny do obliczania prędkości reakcji. Ostatnia grupa to informacje wymagane w formacie SBML, które zostały dodane do ModeLang. Istnieje duża grupa informacji, która jest częścią formatu SBML i nie znajduje się w ModeLang ponieważ SBML jest znacznie bardziej uniwersalnym formatem. Nie było jednak ważne zajmowanie się wszystkimi możliwymi aspektami SBML. Proces standaryzacji ModeLang w celu przekształcenia do formatu SBML, oprócz kilku czysto implementacyjnych modyfikacji, rozszerzył także funkcjonalność języka ModeLang. Z uwagi na fakt, że ModeLang jako język został zaprojektowany do definiowania modeli systemów dynamicznych, nie było konieczne podawanie szczegółów związanych z samym procesem symulacji. Dlatego nie można było zainicjować początkowych wartości agenta. Podczas pracy nad konwersją do SBML ten element został dodany do języka ModeLang.

Weryfikacja modeli i wyników eksperymentów poprzez bazę VirDB

Praca nad projektem bazy modeli infekcji wirusowych została zainspirowana ideą crowdsourcingu, która cieszyła się ogromną popularnością w ostatnich latach. Jest to podejście oparte na podstawach poszanowania praw autorskich, przy jednoczesnym upoważnieniu do dzielenia się i wspólnego wykorzystywania wyników osiąganych przez poszczególnych członków społeczności. Podczas pracy nad rozwiązaniem polegającym na udostępnianiu materiałów badawczych należy wziąć pod uwagę wiele ważnych kwestii. Przede wszystkim treść nie powinna być modyfikowana przez innych użytkowników ze względu na krytyczny charakter wyników badań i możliwość wnioskowania na ich podstawie. Inną ważną kwestią była możliwość jednoczesnego dostępu każdego z autorów do treści udostępnianych przez innych użytkowników. Ważne było również, aby treść była publicznie dostępna i dlatego każdy, kto był zarejestrowanym użytkownikiem, bez względu na swój wkład badawczy, mógł skorzystać z dostarczonych informacji.

W przypadku uruchamiania narzędzi, które miały analizować zbiory wyników dla poszczególnych modeli pod względem statystycznym, istotna była uniwersalna metoda nadawania znaczników etykietujących wyniki. Ważne było również, czy w ramach wynikowego narzędzia powinna istnieć możliwość zamieszczania komentarzy pod wynikami badań, które umożliwiłyby dyskusję i poddawanie wzajemnej krytyce i ocenie.

Ostatnim ważnym elementem związanym z budowaniem rozwiązania opartego na idei Crowdsourcingu była kwestia nadzoru i administracji wprowadzonymi danymi. Możliwość wprowadzania danych bez nadzoru i procesu weryfikacji niesie ryzyko popełnienia błędów, powielania znaczników i etykiet, co później utrudnia użytkownikom analizę wprowadzonych wyników lub w przypadku maszyn przetwarzających wprowadzone informacje, istnieje ryzyko nieprawidłowego katalogowania i indeksowania treści. Dlatego ważne jest, aby móc nadzorować, czy wprowadzana treść jest zgodna z pewnymi założeniami, czy wyniki dla tych samych przypadków mają prawidłowe etykiety, które pozwolą na porównanie wyników tego samego typu ze sobą, a na koniec czy wszystkie wprowadzone informacje są w odpowiedniej postaci, aby platforma była czytelna i wartościowa. Istnieje ryzyko, że treść wprowadzona bez zachowania określonej postaci od początku działania rozwiązania opartego na idei crowdsourcingu zostanie doprowadzona przez użytkowników do stanu, w którym nie będzie możliwe skuteczne uzyskanie wartości z wprowadzonych informacji.

Podstawowym celem bazy wirusów było umożliwienie naukowcom dzielenia się wynikami badań. Projekt obejmował wykorzystanie bazy w obszarze związanym z eksperymentami badawczymi przeprowadzonymi w ramach prac nad ModeLang. Podstawowym obszarem zastosowania bazy wiru-

sów jest wirusologia. W kontekście języka ModeLang możliwe jest przechowywanie w niej opracowanych w nim modeli oraz ich ewaluacja w oparciu o standardowe benchmarki. Modele te mogą być następnie wykorzystane w trakcie prac biologicznych lub medycznych w zakresie wirusologii lub immunologii. Przykładowo eksperymenty zaprojektowane w celu potwierdzenia poprawnego działania języka ModeLang i konwersji ModeLang do formatu SBML przeprowadzono na podstawie przykładów z różnych systemów biologicznych. Dziedziną, która była pierwotną motywacją do pracy nad ModeLang była wirusologia. Dlatego modele związane z zakażeniami wirusowymi HCV i HIV były znaczącym elementem badań eksperymentalnych. Dlatego zebrane informacje mogłyby zostać bezpośrednio wykorzystane do uzupełnienia bazy danych i porównania wyników eksperymentów.

Bibliografia

Baitaluk M, Sedova M, Ray A, Gupta A (2006) BiologicalNetworks: visualization and analysis tool for systems biology. Nucleic Acids Res 34:W466–W471. https://doi.org/10.1093/nar/gkl308

Bergmann FT, Sauro HM (2006) SBW - A Modular Framework for Systems Biology. In: Proceedings of the 2006 Winter Simulation Conference. Monterey, CA, pp 1637–1645

Ciocchetta F, Hillston J (2009) Bio-PEPA: A framework for the modelling and analysis of biological systems. Theor Comput Sci 410:3065–3084. https://doi.org/10.1016/j.tcs.2009.02.037

Clark P, Murray WR, Harrison P, Thompson J (2010) Naturalness vs. Predictability: A Key Debate in Controlled Languages. In: Fuchs NE (ed) International Workshop on Controlled Natural Language. Springer Berlin Heidelberg, pp 65–81

Collier N (2001) Repast: An extensible framework for agent simulation. Nat Resour Environ Issues 8:4

Danos V, Feret J, Fontana W, et al (2008) Rule-Based Modelling, Symmetries, Refinements. In: Formal Methods in Systems Biology. Springer Berlin Heidelberg, pp 103–122

Fuchs NE, Schwitter R (1996) Attempto Controlled English (ACE). In: Proceedings of The First International Workshop On Controlled Language Applications. Katholieke Universiteit Leuven, Belgium, pp 124–136

Funahashi A, Matsuoka Y, Jouraku A, et al (2008) CellDesigner 3.5: A Versatile Modeling Tool for Biochemical Networks. Proc IEEE 96:1254–1265. https://doi.org/10.1109/JPROC.2008.925458

Hoops S, Sahle S, Gauges R, et al (2006) COPASI—a COmplex PAthway SImulator. Bioinformatics 22:3067–3074. https://doi.org/10.1093/bioinformatics/btl485

Hucka M, Nickerson DP, Bader GD, et al (2015) Promoting Coordinated Development of Community-Based Information Standards for Modeling in Biology: The COMBINE Initiative. Front Bioeng Biotechnol 3:19. https://doi.org/10.3389/fbioe.2015.00019

Keller R, Dörr A, Tabira A, et al (2013) The systems biology simulation core algorithm. BMC Syst Biol 7:55. https://doi.org/10.1186/1752-0509-7-55

Leros A, Andreatos A, Zagorianos A (2010) Matlab-Octave science and engineering benchmarking and comparison. In: ICCOMP'10 Proceedings of the 14th WSEAS international conference on Computers: part of the 14th WSEAS CSCC multiconference. Corfu, Greece, pp 746–754

Loew LM, Schaff JC (2001) The Virtual Cell: a software environment for computational cell biology. Trends Biotechnol 19:401–406. https://doi.org/10.1016/S0167-7799(01)01740-1

Macey R, Oster G, Zahnley T (2000) Berkeley Madonna user's guide. University of California, Berkeley, CA

Machné R, Finney A, Müller S, et al (2006) The SBML ODE Solver Library: a native API for symbolic and fast numerical analysis of reaction networks. Bioinformatics 22:1406–1407. https://doi.org/10.1093/bioinformatics/btl086

Olivier BG, Rohwer JM, Hofmeyr J-HS (2005) Modelling cellular systems with PySCeS. Bioinformatics 21:560–561. https://doi.org/10.1093/bioinformatics/bti046

Phillips A, Cardelli L (2007) Efficient, Correct Simulation of Biological Processes in the Stochastic Pi-calculus. In: Calder M, Gilmore S (eds) Computational Methods in Systems Biology. Springer Berlin Heidelberg, pp 184–199

Prejzendanc T, Wasik S, Blazewicz J (2016) Computer Representations of Bioinformatics Models. Curr Bioinforma 11:551–560

Raymond G, Butterworth E, Bassingthwaighte J (2003) JSIM: Free software package for teaching phyiological modeling and research. J Exp Biol 280:102–107

Sedwards S, Mazza T (2007) Cyto-Sim: a formal language model and stochastic simulator of membrane-enclosed biochemical processes. Bioinformatics 23:2800–2802. https://doi.org/10.1093/bioinformatics/btm416

Shapiro BE, Levchenko A, Meyerowitz EM, et al (2003) Cellerator: extending a computer algebra system to include biochemical arrows for signal transduction simulations. Bioinforma Oxf Engl 19:677–678. https://doi.org/10.1093/bioinformatics/btg042

Tisue S, Wilensky U (2004) NetLogo: A simple environment for modeling complexity. In: International Conference on Complex Systems. Boston, MA, pp 16–21

Wasik S, Prejzendanc T, Blazewicz J (2013) ModeLang: A New Approach for Experts-Friendly Viral Infections Modeling. Comput Math Methods Med 2013:1–8. https://doi.org/10.1155/2013/320715

Weidemann A, Richter S, Stein M, et al (2008) SYCAMORE—a systems biology computational analysis and modeling research environment. Bioinformatics 24:1463–1464. https://doi.org/10.1093/bioinformatics/btn207

White C, Schwitter R (2009) An Update on PENG Light. In: Proceedings of the Australasian Language Technology Association Workshop 2009. Sydney, pp 80–88

Yang Yong, Atkinson Peter, Ettema Dick (2007) Individual space–time activity-based modelling of infectious disease transmission within a city. J R Soc Interface 5:759–772. https://doi.org/10.1098/rsif.2007.1218