Sapienza
Università di Roma

Artificial Intelligence and Model Checking Methods for In Silico Clinical Trials

Ph.D. Programme in Computer Science – XXXII Cycle

Candidate
Stefano Sinisi

Thesis Advisor
Prof. Toni Mancini

Co-Advisor
Prof. Enrico Tronci

September 2020

Artificial Intelligence and Model Checking Methods for In Silico Clinical Trials
Ph.D. thesis. Sapienza University of Rome

This thesis has been typeset by LaTeX.

Version: February 2021

Author's email: sinisi@di.uniroma1.it

To my grandparents,
for taught me true love and sacrifice.

Abstract

Model-based approaches to safety and efficacy assessment of pharmacological treatments (In Silico Clinical Trials, ISCT) hold the promise to *decrease* time and cost for the needed experimentations, *reduce* the need for animal and human testing, and enable *personalised* medicine, where treatments tailored for each single patient can be designed before being actually administered.

Research in Virtual Physiological Human (VPH) is harvesting such promise by developing *quantitative mechanistic models* of patient physiology and drugs. Depending on many *parameters*, such models define physiological differences among different individuals and different reactions to drug administrations. Value assignments to model parameters can be regarded as Virtual Patients (VPs). Thus, as *in vivo* clinical trials test relevant drugs against suitable candidate patients, ISCT simulate effect of relevant drugs against VPs covering possible behaviours that might occur *in vivo*.

Having a population of VPs *representative* of the whole spectrum of human patient behaviours is a *key enabler* of ISCT.

However, VPH models of practical relevance are typically too complex to be solved analytically or to be formally analysed. Thus, they are usually solved numerically within simulators. In this setting, Artificial Intelligence and Model Checking methods are typically devised. Indeed, a VP coupled together with a pharmacological treatment represents a closed-loop model where the VP plays the role of a physical subsystem and the treatment strategy plays the role of the control software. Systems with this structure are known as Cyber-Physical Systems (CPSs). Thus, simulation-based methodologies for CPSs can be employed within personalised medicine in order to compute representative VP populations and to conduct ISCT.

In this thesis, we advance the state of the art of simulation-based Artificial Intelligence and Model Checking methods for ISCT in the following directions.

First, we present a Statistical Model Checking (SMC) methodology based on hypothesis testing that, given a VPH model as input, computes a population of VPs which is *representative* (*i.e.*, large enough to represent all relevant phenotypes, with a given degree of statistical confidence) and *stratified* (*i.e.*, organised as a multi-layer hierarchy of homogeneous sub-groups). Stratification allows ISCT to adaptively focus on specific phenotypes, also supporting prioritisation of patient sub-groups in follow-up *in vivo* clinical trials.

Second, resting on a representative VP population, we design an ISCT aiming at *optimising* a complex treatment for a patient digital twin, that is the virtual counterpart of that patient physiology defined by means of a set of VPs. Our ISCT employs an *intelligent search* driving a VPH model simulator to seek the lightest but still effective treatment for the input patient digital twin.

Third, to enable interoperability among VPH models defined with different modelling and simulation environments and to increase efficiency of our ISCT, we also design an *optimised simulator driver* to speed-up backtracking-based search algorithms driving simulators.

Finally, we evaluate the effectiveness of our presented methodologies on state-of-the-art use cases and validate our results on retrospective clinical data.

Acknowledgments

First and foremost, I would like to thank my supervisors, Prof. Toni Mancini and Prof. Enrico Tronci, for their insightful suggestions and continuous support. Their invaluable advice has been and always will be extremely precious. Their plentiful expertise and extensive knowledge contributed to sharpen my research path and shape my future directions.

I would like to extend my thanks to the external reviewers, Prof. Radu Grosu and Prof. Paolo Zuliani. It was a pleasure to receive their grateful comments and meticulous suggestions that helped me to improve the quality of my work.

Next, I am deeply grateful to my colleague and friend, Prof. Federico Mari, for his valuable guidance, effort and assistance at every stage of my PhD path. His encouragements and practical suggestions played a decisive role.

I would like to extend my sincere gratitude to my colleague and friend, Dr Vadim Alimguzhin. His invaluable technical support and critical thinking have been crucial in shaping my method.

I would also like to thank team members of the Model Checking Laboratory of the Computer Science Department of Sapienza University of Rome. They made my life as a PhD student a wonderful time. It is a pleasure to be part of such an outstanding team.

I would like to offer my special thanks to my friend, Viviana, for her sympathetic ear. She is always there for me.

Last but not least, I am extremely grateful to my girlfriend, Stefania, for her endless love, her emotional support and her huge understanding. My appreciation also goes to my parents and my sister. Without their encouragements and love, it would have been impossible to overcome encountered obstacles.

Contents

1 Introduction **1**
- 1.1 Virtual Physiological Human . . . 3
- 1.2 Populations of virtual patients . . . 4
- 1.3 Individualised pharmacological treatments . . . 6
- 1.4 Interoperability and simulation efficiency . . . 7
- 1.5 Thesis outline and summary of contributions . . . 8

2 Background **13**
- 2.1 Formal Framework . . . 13
- 2.2 Case Studies . . . 15

3 Computation of complete populations of virtual patients **23**
- 3.1 Introduction . . . 23
- 3.2 Methods . . . 25
- 3.3 Experimental Results . . . 39
- 3.4 Related Work . . . 45
- 3.5 Conclusions . . . 47

4 Computation of optimal personalised treatments **49**
- 4.1 Introduction . . . 49
- 4.2 Methods . . . 50
- 4.3 Experimental Results . . . 56
- 4.4 Related Work . . . 66
- 4.5 Conclusions . . . 68

5 Reconciling interoperability and simulation efficiency **71**
- 5.1 Introduction . . . 71
- 5.2 Methods . . . 74
- 5.3 Experimental Results . . . 86
- 5.4 Related Work . . . 94
- 5.5 Conclusions . . . 97

6 Conclusions **99**

Bibliography **101**

List of Acronyms **123**

Chapter 1

Introduction

To bring a new biomedical product (*e.g.*, drug, treatment, vaccine, medical device) to the market is highly expensive in terms of time and money. Several studies through the years (see, *e.g.*, [64, 166, 177]) show that, in the current clinical practice, the development and approval of a new drug could cost *up to* 1 billion dollars and *up to* 15 years of hard work. This long process can be summarised in the following main phases: an initial R&D and pre-clinical activities (Phase 0) where laboratory studies, *i.e.*, *in vitro* and *in vivo* tests on animals, are conducted; a clinical trial (Phase I, II and III) where *in vivo* tests on patients and human volunteers are conducted and, finally, a post-market surveillance (Phase IV). Phase I, II and III, *i.e.*, clinical trials, are of paramount importance to assess safety and efficacy of the drug or device under test. In particular, during Phase I, a small group of patients (or volunteers) is recruited to test safety of the drug. Phase II is demanded to test the effectiveness of the drug on a larger group of patients. In Phase III, the drug under test is distributed to multiple hospitals in different countries and the safety and efficacy evaluation is conducted on a much larger group of patients. The latter phase is very important due to the large set of patients involved with the aim of representing *all* human phenotypes in terms of human variability regarding the possible reactions to administrations of the new drug under test. A recent study [63] estimates that the 75% of the total cost of drug development and approval is spent in these three phases. Unfortunately, due to several reasons such as, *e.g.*, significant differences when moving from *in vitro* to *in vivo*, scarce representativeness of the patients involved (see, *e.g.*, [34]) and/or complexity of diseases, the majority of clinical trials mostly fails at the last phase causing an important loss. Also, for some rare diseases the whole process is too costly with respect to the limited return and more hard to accomplish due to the lack of patients.

In this setting, a revolutionary and radical turn is promoted by the introduction of model-based approaches into medicine for the approval of a new biomedical product. Model-based approaches have been used for decades for the *development phase* of biomedical devices and also of drugs. For example, pharmaceutical companies use computational models to predict both the pharmacokinetics and the pharmacodynamics (PK/PD) of drugs that define changes of the drug concentration and the drug effect over time in the human body, respectively. In a similar way, biomedical devices are designed using model-based methods to simulate their behaviour and to verify their design requirements (*e.g.*, safety properties). However, as described above, such novel devices and drugs must then undergo to a slow approval process to assess their safety and efficacy, namely Phase I, II and III, where human patients enter in the loop. In these phases the use of modelling and simulation has been set aside for a long time. Recently, with the advent of computational models of patient physiology (*e.g.*, molecules, cells, organs and organ systems), clinical trials can be conducted *in silico*, *i.e.*, in a virtual setting, by exploiting such models of patients (Virtual Patients, VPs) together with models of biomedical devices and PK/PD of drugs. In this setting, effects of drug administrations as well as outcomes of biomedical devices can be simulated without going *in vivo*. The adoption of model-based approaches and In Silico Clinical Trials (ISCT) for safety and efficacy assessment of biomedical products hold the promise to: i) decrease time and costs for the needed experimentations; ii) reduce the need for animal and human testing; iii) enable personalised medicine, where individualised pharmacological treatments are optimised for a specific patient and can be designed before being actually administered. As envisioned by the Avicenna Consortium in 2016 [18]: *“ISCT aim at reducing, refining and partially replacing real clinical trials”*.

The ISCT revolutionary potential is currently recognised by companies (see, *e.g.*, [13]) and regulatory bodies such as, *e.g.*, U.S. Food & Drugs Administration (FDA) and European Medicines Agency (EMA), as an important advancement of the current clinical practice. Indeed, both FDA and EMA are promoting novel guidelines for the use of computer modelling and simulation during the approval process of biomedical products (see, *e.g.*, [208, 209, 210, 70]).

The Virtual Physiological Human (VPH) initiative is harvesting this vision of *in silico* medicine by providing those computational models that, by means of many parameters (*e.g.*, stoichiometric constants, rates and/or patient-specific quantities), define physiological difference among different individuals and different reactions to drug administrations. Each assignment of values to model parameter can be regarded as a VP. A representative population of VPs of the whole spectrum of human behaviours of interest is a key enabler to design ISCT. Indeed, as in vivo clinical trials test biomedical products on suitable candidate patients, ISCT simulate effects of biomedical products on VPs covering possible behaviours that might occur *in vivo*. However, the computation of such a population is the main obstacle to overcome to conduct ISCT and poses several research questions.

In this setting, Artificial Intelligence and Model Checking methods are typically devised. Indeed, the closed-loop model consisting of a VP coupled with a biomedical product model can be seen as a Cyber-Physical System (CPS) model where the patient represents the physical part and the biomedical product the cyber part. Much has been done in other sectors, such as, *e.g.*, aerospace, defence, automotive, smart grid, concerning the modelling and simulation of CPSs. Model-based approaches and, in particular, Model-based Systems Engineering (MBSE) methodology are extensively used to solve safety-critical and mission-critical problems similar to ISCT objectives. For example, in the healthcare context, safety and efficacy assessment can be done by simulating the effect of the biomedical product on, ideally, all VP, exactly as, ideally, CPS Verification and Validation (V&V) aims at evaluating system requirements (*e.g.*, safety properties) under all possible operational scenarios.

In the following sections we describe more in detail our research setting with a particular emphasis on research problems addressed in this thesis as well as our contributions.

1.1 Virtual Physiological Human

The VPH is an European initiative aiming at promoting the innovative vision of model-based medicine where predictive models of human physiology are exploited in all aspects of the prevention, diagnosis, prognostic assessment, treatment of a disease and development of a biomedical product (see https://www.vph-institute.org). Such an initiative has been introduced in 2007 by the European Commission [44] inspired by the Physiome Project (http://physiomeproject.org) focusing on building a comprehensive framework for modelling the whole human body through the integration of different quantitative computational models which, in turn, define molecules, cells, organs and organ systems [100].

VPH computational models are defined by encoding *qualitative* knowledge of the human physiology of interest (*e.g.*, from the literature or pathways databases like KEGG [111], Reactome [72] or MetaCyc [38]) as well as PK/PD of medicinal compounds (*e.g.*, [131]) into mathematical systems such as, *e.g.*, Ordinary Differential Equations (ODEs) or difference equations (see, *e.g.*, [21, 50, 102]). In particular, PharmacoKinetics (PK) and PharmacoDynamics (PD) models describe drug concentrations and effects over time in the human body (see, *e.g.*, [168]), respectively.

An important aspect of VPH computational models is the capability of defining VPs to take into account inter-subject variabilities, *i.e.*, the possible physiological difference between individuals and possible reactions to drug administrations. This is achieved by means of model parameters such as, stoichiometric constants, rates or other patient-specific quantities. Indeed, different assignments to model parameter (*i.e.*, VP) yield to different model behaviours and, also, to different reactions to drug administrations.

For this reason, during the last decades, VPH has acquired a central role in supporting and harvesting promises hold by ISCT as VPs predict reactions of human patients to therapies (*e.g.*, drug, treatments, biomedical devices). In particular, research in VPH provides mechanistic quantitative models of the human physiology at different levels of scale and for various medical areas (*e.g.*, cardiology, endocrinology, oncology and neurology, see [204, 51, 184, 182, 104]). For example, several models have been proposed for *molecules* (*e.g.*, [186]), cells (*e.g.*, [19, 101]) and single organs (*e.g.*, [223, 51, 169]). At upper levels in the hierarchy, we find models of human body *compartments*. Good examples are models in [110, 93, 53], which define the glucose regulation mechanism in patients with Type 1 Diabetes Mellitus, and the Hypothalamic–Pituitary–Gonadal (HPG) axis model presented in [184], which is specifically focused on hormones related to the human menstrual cycle. Finally, at the top level we find models of the *whole human* body, for example the Glucose-Insulin Model [189, 190], HumMod [94], and Physiomodel [154]. Independently of the level of scale used, most VPH models are defined by means of complex, highly non-linear differential equations. Due to their complexity, in order to provide quantitative information about the time course of the modelled biological quantities (*e.g.*, blood hormone or glucose concentrations) models of practical relevance cannot be analysed analytically, but need to be numerically simulated (*e.g.*, [99, 136]). VPH models at different positions in the above hierarchy are typically defined at different levels of abstraction. In order to find a trade-off between simulation accuracy and efficiency, *multi-scale* models can be built by integrating, interconnecting, and co-simulating models of the different organs or body compartments of interest, exploiting the different available levels of abstraction as required by the focus and scope of the *in silico* analysis to be performed.

1.2 Populations of virtual patients

As anticipated, one of the main enablers to perform an ISCT is the availability of a *population of virtual patients*, able to predict (via *simulation*) the impact of a therapy (*i.e.*, a pharmacological treatment or a medical device) by providing relevant clinical measurements (see, *e.g.*, [210, 70]). For an ISCT to provide *compelling evidence* of the safety and the efficacy of a therapy and to support its design and revision, such population must be *complete*, *i.e.*, representative of the *entire spectrum* of behaviours deemed of interest, from both physiology and drug PK/PD points of view.

Virtual Patients (VPs) are typically derived by *parametrising* quantitative mechanistic VPH models. Unfortunately, the computation of complete populations of VPs to be used for ISCT is a complex task for various reasons: First, as previously described, quantitative VPH models of practical relevance (*e.g.*, those encoding complex biological pathways) are often too complex to be analysed analytically, and *numerical simulation* is the only viable means to compute their evolution under any given parameter value (*e.g.*, [99]). Second, their parameter space is often *infinite*, as most parameters are real-valued. Third, most parameter values *might not* actually represent VPs, as their associated model evolutions are physiologically meaningless. This is due to, *e.g.*, over-parametrisation, presence of unknown (hence, not modelled) interdependency constraints among parameters, and use of parameters to define too complex or not-well-understood physiological mechanisms. Last, the values of most parameters are often not measurable through clinical assays in human patients.

In this setting, when the VPH model at hand is *identifiable*, *i.e.*, a model behaviour is uniquely determined from a parameter value assignment [122], a complete population of VPs can be computed by fitting the model (*e.g.*, [162, 184, 191, 49]) against a set of *in vivo* measurements (*e.g.*, [116, 131]) deemed representative of the entire spectrum of behaviours of interest. Conversely, in case of *non-identifiable* models (*e.g.*, [42]), although such techniques can still be used (*e.g.*, [202, 10, 183, 214] and citations thereof), the resulting population of VPs is *not* guaranteed to be complete, *regardless* of how large and representative are the available *in vivo* data used for fitting. Additionally, if small changes in such parameter values are expected to be physiologically meaningful as well, random perturbations can be exploited to generate additional VPs quite easily (*e.g.*, [214]). Hence, although such VPs can still be used to find cases (*counterexamples*) where the therapy under assessment is unsafe and/or ineffective, non-identifiability hinders the possibility to have a *comprehensive picture* of the cases where the therapy succeeds or fails, as there could be other (possibly very different) model parameter values (not selected through fitting) still matching experimental data, but leading to different model behaviours under the new therapy.

As a result, although such models are based on solid *scientific principles* (*e.g.*, biochemical reactions), their *non-identifiability* makes it hard to use them to qualify a therapy as safe and effective, no matter how large is the input dataset used to generate the population of VPs employed in the ISCT. In the recent EMA physiologically-based PK guidelines for reporting modelling and simulation [70] some considerations concerning the identifiability of the model are mandatory and must be eventually supported by in-vivo tests.

The above problem stemming from non-identifiability is quite common also in other areas. For example, models used in machine learning (*e.g.*, neural networks) are typically non-identifiable, and it is well known that, notwithstanding how large is the training dataset, it is possible to find (plausible) input data leading to complete wrong classifications (*e.g.*, [71]). Indeed, this is among the main obstacles in qualifying machine learning–based approaches within safety-critical applications (*e.g.*, autonomous driving) (see, *e.g.*, [73, 106]).

The above considerations motivate the development of methods and software that, possibly building on parameter estimation against *in vivo* data, for a possibly non-identifiable VPH model, can compute a finite set of VPs which are: i) *physiologically meaningful*, *i.e.*, they all define behaviours deemed possible *in vivo*; ii) pairwise *distinguishable*, *i.e.*, their associated model behaviours on the input scenarios relevant for the ISCT are different enough, according to some metrics, so that no computation is wasted; and iii) *representative* of the entire spectrum of behaviours defined by the given VPH model (*completeness*).

1.3 Individualised pharmacological treatments

The slow and costly clinical trial process for the approval of a new pharmacological treatment is among the main reasons why pharmacological treatments are often designed for the average patient. In fact, to tailor a treatment strategy for a single patient or a group of patients is currently infeasible as it should require a clinical trial for each possible variation of the treatment strategy under test and for each possible patient (or class of patients exhibiting similar reactions to drug administrations). Hence, current pharmacological treatments are often not effective for a large number of patients. Examples are female fertility treatments or lung cancer treatments which have a success rate of about 35% (see, *e.g.*, [207]) and 25% (see, *e.g.*, [76]), respectively. Personalised medicine, as also outlined by the International Consortium for Personalised Medicine (ICPerMed) (https://www.icpermed.eu), addresses the above challenges by aiming at developing pharmacological treatments *optimised* for any given individual, namely *personalised* treatments (see, *e.g.*, [18, 176]). Several optimisation criteria can be defined. A typical criterion is the minimisation of the overall amount of drug used also motivated by the fact that about 75% of adverse drug reactions are dose-related [47]. Also, minimising doses of drug employed typically decreases the treatment cost. With their amenability to define different individuals, VPH models of proved accuracy are a *key enabler* for personalised medicine. Indeed, VPs can be exploited to define human patient digital twins capturing the patient-specific physiology and the patient-specific reactions to drug administrations. In turn ISCT can be conducted on a human digital twin to test all the different possible variations of a pharmacological treatment in order to find the lightest-but-still-effective treatment for that specific patient.

The above considerations motivate the development of methods and software that, resting on a VPH model and a complete population of VPs, optimise a pharmaceutical treatment strategy by individualising the drug dosage, saving the overall amount of drug and keeping the treatment successful.

1.4 Interoperability and simulation efficiency

As previously described, in an ISCT setting, a VP coupled together with a pharmacological treatment strategy (or a biomedical device) represents a closed-loop model where the VP plays the role of the physical subsystem (plant) and the treatment strategy (or the biomedical device) plays the role of the control software (controller). As anticipated, systems with this structure are known as CPSs. Due to the complexity of CPSs, simulation-based approaches are typically used to support CPS design and V&V in several domains such as: aerospace, defence, automotive, smart grid and healthcare. Accordingly, many simulation-based tools are available to support CPS design. This, on one side, enables designers to choose the toolchain that best suits their needs, and on the other side poses huge interoperability challenges when one needs to simulate CPSs whose subsystems have been designed and modelled using different toolchains.

To overcome such an interoperability problem, in 2010 the Functional Mock-up Interface (FMI) has been proposed as an open standard to support both Model Exchange (ME) and Co-Simulation (CS) of simulation models created with different toolchains. FMI has been adopted by several modelling and simulation environments. Models adhering to such a standard are called Functional Mock-up Units (FMUs). Indeed FMUs play an essential role in defining complex CPSs through, *e.g.*, System Structure and Parametrization (SSP) standard. This is true also for the ISCT setting where similar interoperability issues are present. Indeed, different languages are traditionally used to define VPH models. Among them are SBML [99], Physiological Hierarchy Markup Language (PHML) (see, *e.g.*, [16]) and the Modelica language (www.modelica.org). Modelica is one of the most widely adopted open-standard general-purpose modelling languages for networks of dynamical systems in many areas of engineering and it has been proposed as a common modelling platform for the integration of VPH, PK/PD models (see, *e.g.*, [115]) in order to facilitate their co-simulation and analysis. Also, the equation-based and the acausal modelling paradigm of Modelica is perfectly suitable for physical systems having continuous-time dynamics such as VPH models of human physiology and drug PK/PD. On the other hand, Simulink (https://mathworks.com) is the *de facto* industrial standard for modelling and simulation of, *e.g.*, control systems, hardware architectures and sensors that exhibit a discrete-time or discrete-event dynamics. For these reasons, biomedical devices are usually defined using Simulink (see, *e.g.*, [15, 170, 108]).

In this setting, the FMI 2.0 standard [27] for model exchange enables the interoperability between Modelica and Simulink and, hence, the coupling of a biomedical device with a VPH model to conduct, via simulation, ISCT. Moreover, as for the ISCT setting, simulation-based V&V of CPSs typically requires exploring different simulation scenarios (*i.e.*, exogenous input sequences to the CPS under design). To avoid simulating many times shared prefixes of different simulation scenarios, the simulator state at the end of a shared prefix is saved and then restored on demand, as a start state. In this context, an important FMI feature is the capability to save and restore the internal FMU state on demand. This is crucial to increase efficiency of simulation-based V&V. Unfortunately, the implementation of this feature is not mandatory and it is available only within some commercial software.

As a result, the interoperability enabled by the FMI standard cannot be fully exploited when using open-source simulation environments. Furthermore, none of the currently available open-source Modelica environments implement the FMI 2.0 optional feature for saving and restoring FMU states. This motivates developing such a feature for open-source Modelica environments for CPSs.

1.5 Thesis outline and summary of contributions

In this thesis we focus on simulation-based artificial intelligence and model checking methods to perform ISCT. The following outline summarises thesis chapters and contributions.

Chapter 2 presents the needed background knowledge and the formal framework. Moreover, it describes our case studies used through this thesis. Namely, i) a quantitative VPH ODE-based model of human signalling pathways of the female HPG-axis defining the behaviour of 33 biological species (mostly hormones); ii) an assisted reproduction treatment currently in use at the Department of Reproductive Endocrinology of University Hospital Zurich; and iii) retrospective clinical data kindly made available by Hannover Medical School (a total of 35 clinical records), University Hospital of Lausanne (a total of 39 clinical records) and Pfizer (a total of 12 clinical records) and used to evaluate and assess the effectiveness and efficacy of methodology presented in this thesis.

Chapter 3 presents a methodology to compute a *population of VPs*. Such a methodology is based on a global search algorithm driven by a statistical model checking approach that exploits a quantitative VPH model of human physiology and drugs (PK/PD) and suitable biological and medical knowledge elicited from experts. The computed population of VPs yields behaviours that are i) *distinguishable* from each other (*i.e.*, different phenotypes); ii) *representative* of the whole phenotype spectrum entailed by the input VPH model; and iii) *stratified* at different levels to form a hierarchy of classes of behaviours. The latter enables a *full granularity control* on the size of the population to be employed in an ISCT, guaranteeing representativeness *while avoiding* over-representation of behaviours. We prove the effectiveness of our algorithm on our case study, *i.e.*, a complex and non-identifiable VPH model of the HPG-axis. In particular, we generate, exploiting 60 days of computation on HPC infrastructure, a population of 4 830 264 VPs stratified into a hierarchy of 7 levels. Also, we experimentally assess the representativeness of such a population of VPs using retrospective clinical data on 86 medical cases from Pfizer, Hannover Medical School and University Hospital of Lausanne. Finally, we show that datasets are covered by our VPs within Average Normalised Mean Absolute Error of 15%, 20%, and 35% (90% of the latter dataset is covered within 20% error), respectively.

Chapter 4 presents a methodology that, starting from a standard pharmacological treatment, automatically synthesises an individualised variation, optimised for a given human patient, by means of an ISCT. The employed optimisation criterion aims at reducing the overall treatment drug amount while keeping the treatment successful for that patient. As anticipated, minimising treatment drug doses is a way to reduce the probability of adverse drug reactions [47]. To this end, we exploit a quantitative VPH model of the physiology and PK/PD relevant for the standard pharmacological treatment to optimise. Also, given retrospective clinical data of a human patient and a representative population of VPs, we compute a patient digital twin, *i.e.*, a digital representation of that specific patient physiology. Formally, we define a digital twin as a set of VPs each one yielding a model behaviour that matches human patient clinical data within a given error bound. Given a patient digital twin, our automatic synthesis approach consists of performing extensive simulations (ISCT) guided by an intelligent (*i.e.*, AI-based) search taking as input the possible drug doses to be administered and a set of invariants and goals, *i.e.*, properties that must be *always* and *eventually* satisfied, respectively, which formalises clinical guidelines for a successful and safe treatment. Our algorithm explores the space of possible clinical action sequences while seeking for a treatment that it is safe and effective for the input patient digital twin, *i.e.*, if administered to the given patient digital twin, it always satisfies invariants and goals. During the search, our algorithm computes the effect of clinical actions on the input digital twin through a black-box simulator for the VPH model at hand. To do this efficiently, our search algorithm is backtracking-based. Backtracking step occurs when at least one VP of the input digital twin violates treatment invariants or when no further exploration of a subspace can lead to a better solution than the current optimal one. Moreover, to speed up the search space exploration the following heuristics are used: i) *action ordering heuristic* to sort candidate clinical actions at each search step; and ii) *dynamic VP ordering heuristic* to dynamically adjust the order of VPs in the input digital twin to be simulated at each search step according to a sort of *early pruning* such that, on average, the simulation time improves. Finally, we evaluate the effectiveness of our approach on our case study, *i.e.*, the downregulation phase of a clinical protocol for assisted reproduction. To this end, we conduct a multi-arm ISCT involving 21 patients for which we have retrospective clinical data.

For each such patient, who defines a distinct arm of our ISCT, we compute the optimal (lightest) still-effective downregulation treatment. Given that the required computation is extremely intensive, we conduct our ISCT on a High Performance Computing (HPC) infrastructure.

Chapter 5 presents an open-source implementation of FMI 2.0 functionalities needed to save and restore internal state of FMUs. As anticipated, simulation-based approaches for V&V of CPSs as well as ISCT typically require exploring different simulation scenarios (*i.e.*, exogenous input sequences to the CPS model). Many such scenarios have a shared prefix. Accordingly, to avoid simulating many times such shared prefixes, the simulator state at the end of a shared prefix is saved and then restored and used as a start state for the simulation of the next scenario (see, *e.g.*, [140]). To this end, we focus on JModelica, an open-source simulation environment for CPSs based on an open-standard modelling language, namely Modelica. In the chapter, we describe how we have endowed JModelica with our open-source implementation of the FMI 2.0 save-and-restore functions. Finally, we present experimental results evaluating, through 934 benchmark models, correctness and efficiency of our extended JModelica. Our experimental results show that simulation-based V&V is, on average, 22 times faster with our get/set functionality than without it.

Publications arising from the thesis

1. T. Mancini, F. Mari, A. Massini, I. Melatti, I. Salvo, S. Sinisi, E. Tronci, R. Ehrig, S. Röblitz, and B. Leeners. Computing personalised treatments through in silico clinical trials. A case study on downregulation in assisted reproduction. In *25th RCRA International Workshop on Experimental Evaluation of Algorithms for Solving Problems with Combinatorial Explosion (RCRA 2018)*, volume 2271 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2018

2. A. Calabrese, T. Mancini, A. Massini, S. Sinisi, and E. Tronci. Generating T1DM virtual patients for in silico clinical trials via AI-guided statistical model checking. In *26th RCRA International Workshop on Experimental Evaluation of Algorithms for Solving Problems with Combinatorial Explosion (RCRA 2019)*, volume 2538 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2019

3. V. Alimguzhin, T. Mancini, A. Massini, S. Sinisi, and E. Tronci. In silico clinical trials through AI and statistical model checking. In *1st Workshop on Artificial Intelligence and Formal Verification, Logic, Automata, and Synthesis (OVERLAY 2019)*, volume 2509 of *CEUR Workshop Proceedings*, pages 17–22. CEUR-WS.org, 2019

4. S. Sinisi, V. Alimguzhin, T. Mancini, E. Tronci, F. Mari, and B. Leeners. Optimal personalised treatment computation through in silico clinical trials on patient digital twins. *Fundamenta Informaticae*, 174(3–4):283–310, 2020

5. S. Sinisi, V. Alimguzhin, T. Mancini, E. Tronci, F. Mari, and B. Leeners. AI-guided synthesis of personalised pharmacological treatments via in silico clinical trials. In *2nd Workshop on Artificial Intelligence and Formal Verification, Logic, Automata, and Synthesis (OVERLAY 2020)*, 2020

6. S. Sinisi, V. Alimguzhin, T. Mancini, E. Tronci, and B. Leeners. Complete populations of virtual patients for in silico clinical trials. *Bioinformatics*, 2020

7. S. Sinisi, V. Alimguzhin, T. Mancini, and E. Tronci. Reconciling interoperability with efficient Verification and Validation within open source simulation environments. *Simulation Modelling Practice and Theory*, 2021

Chapter 2

Background

2.1 Formal Framework

Throughout the thesis we denote with $\mathbb{R}$, $\mathbb{R}_{0+}$, and $\mathbb{R}_{+}$ the sets of, respectively, all, non-negative, and positive reals, and with $\mathbb{N}$ and $\mathbb{N}_{+}$ the sets of, respectively, non-negative and strictly positive integers. Also, given sets A and B, A^B denotes the set of functions from B to A.

2.1.1 Formalisation of VPH models

We adopt a very general approach to define Virtual Physiological Human (VPH) models and view them as parametric input-output dynamical systems, *i.e.*, parametric functions mapping input time functions (denoting external inputs such as drug administrations) to output time functions (defining the time evolution of observable quantities of interest). This general definition (Definition 2.1) is standard in signals and systems (see, *e.g.*, [200, 124]), especially when, as in the case of physiological models, the system internal state is not accessible, and only selected outputs (*system observables*) can be measured. For physical reasons, we also require that our models are *strictly causal*, *i.e.*, their behaviour up to any time instant depends only on *past* inputs. Also, given the presence of parameters, we can focus on deterministic systems, in that parameters embody any initial condition which the system output might depend on. Finally, our definition captures both *continuous-time* models (as, *e.g.*, those defined by means of Ordinary Differential Equations, ODEs) and *discrete-time* models (as, *e.g.*, those defined by means of difference equations).

Definition 2.1. *A parametric deterministic input-output dynamical system* $\mathcal{S}$ *is defined by a tuple* $(\mathcal{T}, \Lambda, \mathcal{U}, \mathcal{Y}, \mathbf{y})$*, where:*

- $\mathcal{T}$*, the time-set, is either* $\mathbb{R}_{0+}$ *(for continuous-time systems) or* $\mathbb{N}$ *(for discrete-time systems);*

- Λ *is the parameter space;*
- $\mathcal{U}$ *is the input space;*
- $\mathcal{Y}$ *is the non-empty output space;*
- $\mathbf{y}$ *is the observation function of* $\mathcal{S}$*: for any input time-function* $\mathbf{u} \in \mathcal{U}^{\mathcal{T}}$ *and parameter value* $\lambda \in \Lambda$*,* $\mathbf{y}(\mathbf{u}, \lambda) \in \mathcal{Y}^{\mathcal{T}}$ *is a function defining the* time evolution *of the system observables at each time point in* $\mathcal{T}$*, when the system parameters are set to* λ *and the system is fed with input function* $\mathbf{u}$*. For any* $t \in \mathcal{T}$*, we denote with* $\mathbf{y}(t;\ \mathbf{u}, \lambda)$ *the value of* $\mathbf{y}(\mathbf{u}, \lambda)$ *at time point* t*.*

System $\mathcal{S}$ *is* strictly causal *if, for any* $\lambda \in \Lambda$*,* $t \in \mathcal{T}$*, and any pair* $\mathbf{u}_1, \mathbf{u}_2 \in \mathcal{U}^{\mathcal{T}}$ *such that* $\mathbf{u}_1(t') = \mathbf{u}_2(t')$ *for all* $t' < t$*, it holds* $\mathbf{y}(t;\ \mathbf{u}_1, \lambda) = \mathbf{y}(t;\ \mathbf{u}_2, \lambda)$*.*

Strict causality implies that the system initial output, $\mathbf{y}(0;\ \mathbf{u}, \lambda)$, depends on λ, but not on $\mathbf{u}$, *i.e.*, that λ also embodies information about any initial condition of the system (and, in turn, the output at time 0).

2.1.2 Clinical records

In order to evaluate our methods and case study with respect to retrospective clinical data we define the notion of clinical records as follow.

Definition 2.2 (Clinical record)**.** *Let a time series* q *be a set of pairs* $\{\mathtt{time}(q,j), \mathtt{val}(q,j)\}$*, where, for each* j*,* $\mathtt{time}(q,j)$ *and* $\mathtt{val}(q,j)$ *are, respectively, the time instant and the value (observation or input) of the* j*-th available element in* q*. A clinical record* $\mathcal{C}$ *is a pair* (o, u)*, where* o *(observations) consists of a time series* $o(s)$ *for each observable biological quantities* s *and* u *(inputs) consists of a time series* $u(d)$ *of each pharmaceutical compound* d *(drug) administered in* $\mathcal{C}$*.*

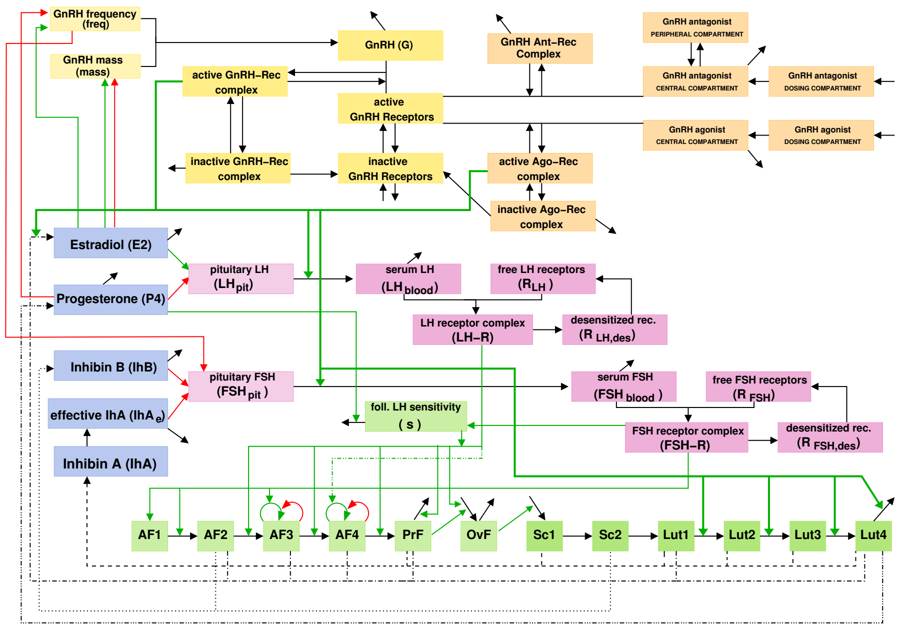


Figure 2.1. Components of the human female menstrual cycle and their relationships, as modelled within GynCycle.

2.2 Case Studies

In this section we present the case studies we use to evaluate our methods.

2.2.1 GynCycle: a VPH model of HPG-axis

GynCycle [184] is a VPH model of the human female Hypothalamic–Pituitary–Gonadal (HPG) axis, focussing on the interactions and feedback mechanisms between components involved in the menstrual cycle like GnRH, FSH, LH, development of follicles and corpus luteum, and the production of E2, P4, Inhibin A and Inhibin B. GynCycle also comprises Pharmacokinetics/Pharmacodynamics (PK/PD) of GnRH analogues and their effects on the menstrual cycle, such as the suppression of gonadotropins. Figure 2.1 shows, as a flowchart, a high-level view of the main components of the human female menstrual cycle as well as of their relationships, as modelled within GynCycle. For each modelled biological species, GynCycle defines a parametric ODE. Below we show the ODEs defining the time evolution of the 4 hormones we regarded as *model observables* in our case-study, *i.e.*, LH, FSH, E2 and P4:

$$
\begin{aligned}
\dot{\mathrm{LH}}_{blood}(t) &= \frac{\mathrm{LH}_{\mathrm{pit}}(t)}{V_{blood}} \cdot \Big(b^{\mathrm{LH}}_{Rel} + k^{\mathrm{LH}}_{\mathrm{G\text{-}R}} \cdot H^{+}(\mathrm{G\text{-}R}(t), T^{\mathrm{LH}}_{\mathrm{G\text{-}R}}; n^{\mathrm{LH}}_{\mathrm{G\text{-}R}})\Big) \\
&\quad - (k^{\mathrm{LH}}_{on} \cdot \mathrm{R}_{\mathrm{LH}}(t) + k^{\mathrm{LH}}_{cl}) \cdot \mathrm{LH}_{blood}(t) \\
\dot{\mathrm{FSH}}_{blood}(t) &= \frac{\mathrm{FSH}_{\mathrm{pit}}(t)}{V_{blood}} \cdot \Big(b^{\mathrm{FSH}}_{Rel} + k^{\mathrm{FSH}}_{\mathrm{G\text{-}R}} \cdot H^{+}(\mathrm{G\text{-}R}(t), T^{\mathrm{FSH}}_{\mathrm{G\text{-}R}}; n^{\mathrm{FSH}}_{\mathrm{G\text{-}R}})\Big) \\
&\quad - (k^{\mathrm{FSH}}_{on} \cdot \mathrm{R}_{\mathrm{FSH}}(t) + k^{\mathrm{FSH}}_{cl}) \cdot \mathrm{FSH}_{blood}(t) \\
\dot{\mathrm{E2}}(t) &= b^{\mathrm{E2}} + k^{\mathrm{E2}}_{\mathrm{AF2}} \cdot \mathrm{AF2}(t) + k^{\mathrm{E2}}_{\mathrm{AF3}} \cdot \mathrm{LH}(t) \cdot \mathrm{AF3}(t) \\
&\quad + k^{\mathrm{E2}}_{\mathrm{AF4}} \cdot \mathrm{AF4}(t) \quad + k^{\mathrm{E2}}_{\mathrm{PrF}} \cdot \mathrm{LH}(t) \cdot \mathrm{PrF} \\
&\quad + k^{\mathrm{E2}}_{\mathrm{Lut1}} \cdot \mathrm{Lut1}(t) + k^{\mathrm{E2}}_{\mathrm{Lut4}} \cdot \mathrm{Lut4}(t) - k^{\mathrm{E2}}_{cl} \cdot \mathrm{E2}(t) \\
\dot{\mathrm{P4}}(t) &= b^{\mathrm{P4}} + k^{\mathrm{P4}}_{\mathrm{Lut4}} \cdot \mathrm{Lut4}(t) - k^{\mathrm{P4}}_{cl} \cdot \mathrm{P4}(t)
\end{aligned}
$$

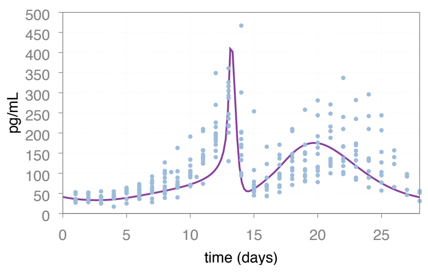
In the differential equations above, LH(t), FSH(t) etc. denote species of interest, while b^{LH}_{Rel}, k^{LH}_{cl}, etc. denote *model parameters*, which typically define patient-specific quantities like decay or reaction rates. Furthermore, as it often happens in complex VPH models, some represented biological mechanisms are not known exactly. This is handled by introducing additional parameters. For example, in order to model stimulatory or inhibitory effects, additional parameters are introduced, together with *Hill functions* of the form:

$$
\begin{aligned}
H^{+}(S(t), T; n) &= \frac{(S(t)/T)^n}{1 + (S(t)/T)^n} \\
H^{-}(S(t), T; n) &= \frac{1}{1 + (S(t)/T)^n}
\end{aligned}
$$

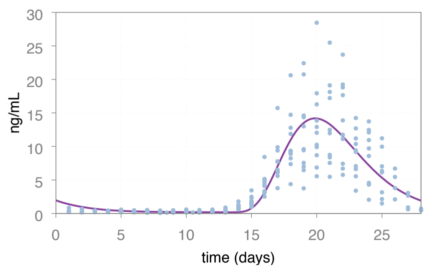
Here, $S(t) \geq 0$ denotes the influencing substance, $T > 0$ is the amount of S that causes 50% of the maximum of H^{+} or H^{-}, and $n \geq 1$ is the Hill coefficient, which determines the rate of switching. The exact values for T and n are not easily observable and hence only a suitable range is known, typically derived from indirect arguments. The whole GynCycle comprises a total of 33 parametric ODEs and 76 model parameters whose values (real numbers) are patient-specific.

The GynCycle model is also equipped with a *reference parameter value*, $\lambda^{(0)} \in \Lambda$, which has been computed in [184] using a Pfizer database comprising 20–25 measures for 4 observed hormones (E2, P4, FSH, and LH) on 12 healthy women, totalling more than 1000 overall measurements. The GynCycle time evolution of E2, P4, LH, and FSH of $\lambda^{(0)}$ is shown in Figure 2.2.

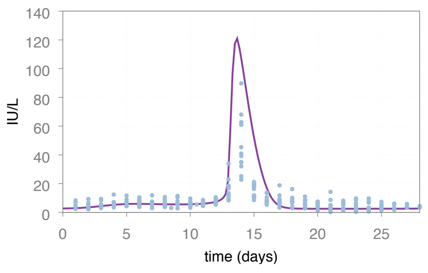
The model supports two pharmaceutical compounds: Triptorelin and Norethisterone, whose effect is the inhibition of the regular menstrual cycle. These drugs



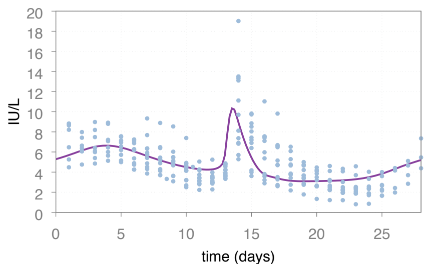
(a) E2



(b) P4



(c) LH



(d) FSH

Figure 2.2. GynCycle evolution under the *reference VP* (curves) averaging data of 12 patients (points) [184].

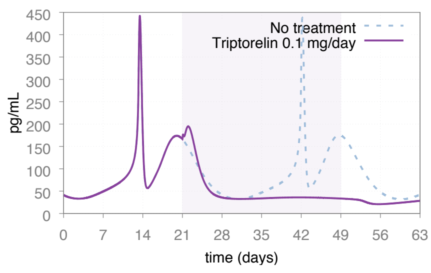
are typically used during pharmaceutical fertility treatments. Figures 2.3 and 2.4 show the expected effects of, respectively, Triptorelin and Norethisterone injections on the observable hormones, namely LH, FSH, E2 and P4.

2.2.1.1 Model Formalisation

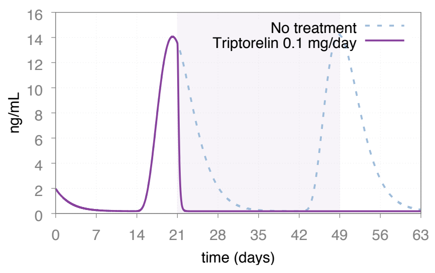
We formalise our GynCycle model as a dynamical system $\mathcal{S} = (\mathcal{T}, \Lambda, \mathcal{U}, \mathcal{Y}, \mathbf{y})$ (Definition 2.1) as follows.

2.2.1.1.1 Time set. Although GynCycle is a continuous-time system, to compute its observation function under a given parameter $\lambda \in \Lambda$ and input function $\mathbf{u}$ we need to resort to *numerical simulation*. This results in both input and observation functions be defined as *bounded-horizon* sequences of samples *evenly spaced in time*. Consequently, for the standpoint of our analysis, we consider the *time set* $\mathcal{T}$ of GynCycle as a bounded interval of $\mathbb{N}_+$.

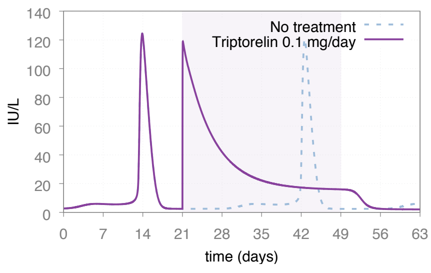
2.2.1.1.2 Parameter space. The model counts 76 real-valued *patient-specific parameters* that define properties of model dynamics such as hormone decay rates,



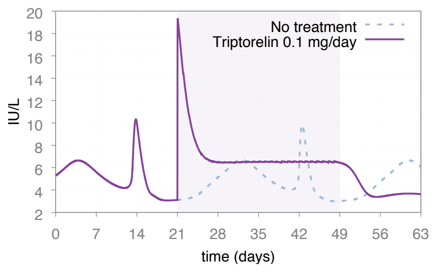
(a) E2



(b) P4



(c) LH



(d) FSH

Figure 2.3. Expected effects Triptorelin on the menstrual cycle. The purple area defines the time window during which the drug is administered.

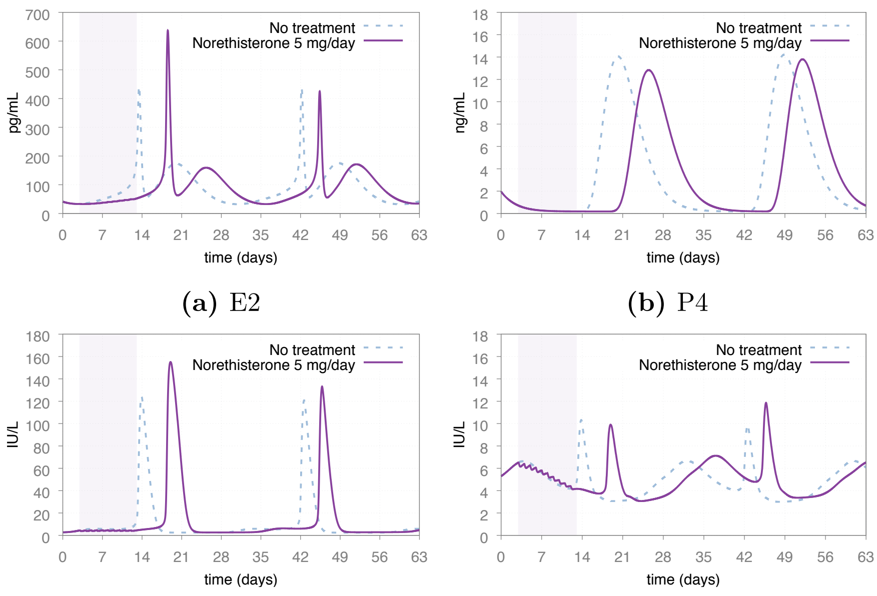
reaction rates, stimulatory and inhibitory effects. Hence, the model parameter space is a closed interval of $\mathbb{R}^{76}$ with known bounds [184].

2.2.1.1.3 Input space. The model input space $\mathcal{U}$ is defined as $\mathbb{R}^2_{0+}$, where each value defines a dose for each of the two supported pharmaceutical compounds. An input function thus defines the sequence of doses administered for each of the two compounds in the model time set.

2.2.1.1.4 Output space. The model output space $\mathcal{Y}$ is $\mathbb{R}^n_{0+}$ where $n \in \mathbb{N}_+$ is the number of (real-valued) model observables. In this thesis, we experiment with $n = 4$ observables, namely: LH, FSH, E2, P4, which are the hormones typically measured in a clinical setting, and for which we have retrospective data (Section 2.2.3).

2.2.2 Assisted reproduction treatments

In this section we, first, give an overview of assisted reproduction protocols and then present our case study, namely, the downregulation phase, currently administered at the Department of Reproductive Endocrinology of University Hospital Zurich (Section 2.2.2.1).



(a) E2 (b) P4

(c) LH (d) FSH

Figure 2.4. Expected effects Norethisterone on the menstrual cycle. The purple area defines the time window during which the drug is administered.

At each menstrual cycle of a human female, among the several follicles initially present in the ovaries, only one, unless in exceptional cases, grows and reaches maturity. A complex competition among follicles takes place, driven by a hormone-based feedback loop, in order to inhibit the maturation of multiple follicles.

An assisted reproduction treatment aims at neutralising (through drugs) such a feedback loop (*downregulation phase*), in order to achieve a controlled and an as-simultaneous-as-possible growth of 5–15 ovarian follicles (*stimulation phase*). When the first three follicles reach a size where a mature oocyte (*i.e.*, an egg cell ready for fertilisation) can be expected (about 18 mm in diameter), ovulation is induced. Then, the oocytes are retrieved, those which are mature are tried to be fertilised *in vitro*, and implanted either immediately within the treatment cycle or later after cryopreservation (*i.e.*, a preservation process by cooling) back into the uterus.

Assisted reproduction treatments are complex and challenging, with low average success rates (around 35% [207]) even in the top clinics, and with many factors that, to date, can be hardly kept under full control. Indeed, as hormonal regulatory systems occur within a complex network of endocrinological, neurological and psychological factors [127, 92, 126], they are difficult to capture within clinical studies, and model-based approaches might be of great aid in taking these many factors under better control.

2.2.2.1 Downregulation phase

Our case study is one of the worldwide classically used downregulation protocols aiming at suppressing the usual hormonal oscillations of the menstrual cycle (as in Figure 2.3) and preparing the patient to the following stimulation.

At the Department of Reproductive Endocrinology of University Hospital Zurich, this protocol currently consists of a sequence of daily administrations of 0.1 mg of Triptorelin (a GnRH analogue). For physiological reasons, downregulation is started within a precise time window of the menstrual cycle, namely between day 21 and day 25. The treatment might have different duration in order to address different patient reactions. In particular, a downregulation treatment is considered *successful* (effective) for a patient in case the blood concentrations of a given set of hormones and other physiological quantities go below certain thresholds within 9 days from the first drug administration, and stay always below such thresholds for the following 21 days. As a consequence, a downregulation treatment might last up to 30 days.

2.2.3 Retrospective clinical records

To evaluate and assess efficacy of the methodology presented in this thesis, we use retrospective clinical data counting 86 anonymised clinical records. Such clinical records have been gathered during the European project PAEON (http://paeon.di.uniroma1.it) and have been kindly made available by Hannover Medical School (a total of 35 clinical records), University Hospital of Lausanne (a total of 39 clinical records) and Pfizer (a total of 12 clinical records). The Pfizer dataset has been originally used in [184] to compute the reference GynCycle VP. In each dataset, for each clinical record we have actual measurements of the blood levels of the 4 model observables (LH, FSH, E2 and P4) on a (roughly) daily basis for an entire menstrual cycle (all clinical records refer to patients subject to no pharmaceutical treatment).

Figure 2.5 shows the distribution of daily measured blood hormone levels on the 86 clinical records using box-and-whisker plots. Data have been aligned with respect to the LH peak, which is typically used to estimate the ovulation day.

Throughout the thesis, we formalise each clinical record in our dataset according to Definition 2.2 where the biological quantities are LH, FSH, E2 and P4, respectively.

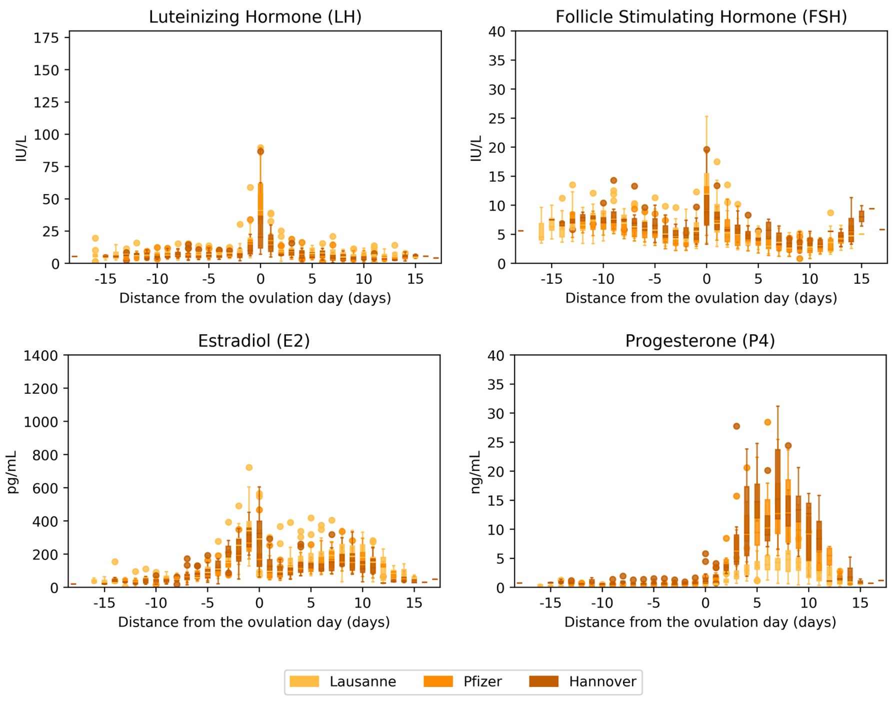


Figure 2.5. Measurement distributions of LH, FSH, E2 and P4 of clinical records within the datasets from Hannover Medical School, University Hospital of Lausanne and Pfizer.

Chapter 3

Computation of complete populations of virtual patients

3.1 Introduction

In this chapter we present methods and software to compute populations of Virtual Patients (VPs) for (possibly non-identifiable) quantitative Virtual Physiological Human (VPH) models. We focus on the typical case of models that, due to their complexity, cannot be analysed symbolically, but need to be numerically simulated (*e.g.*, [99]), and show the effectiveness of our methods on our case study, *i.e.*, a non-identifiable model of the Hypothalamic–Pituitary–Gonadal (HPG) axis (see, Section 2.2.1).

Differently from other approaches, our populations satisfy three important properties: *completeness*, *distinguishability*, and *stratifiedness*.

Completeness means that our populations show *all* model behaviours deemed of interest (*e.g.*, *physiologically meaningful*), even when such a full set of behaviours is *unknown* at model design time (this is typical in large non-identifiable, over-parameterised VPH models, see below). For example, the population we computed in our case study comprises as many as 4 830 264 VPs.

Distinguishability means that no model behaviour (aka *phenotype*) is *over-represented* in our population: any two VPs behave differently (according to a user-defined notion of behavioural distinguishability) in at least one scenario (*e.g.*, input pattern) supported by the model. This avoids waste of computation during an In Silico Clinical Trial (ISCT).

Stratifiedness means that our populations are organised in levels, (strata), each one showing the *entire spectrum* of behaviours under *different indistinguishability criteria*. Although such user-provided criteria can be very general (*e.g.*, focus on different features of model behaviours), a typical definition for them (which we exploit in our case study) is in terms of a hierarchy of classes of behaviours, so that *full granularity control* can be achieved on the number of VPs to be employed in an ISCT, always guaranteeing completeness while avoiding over-representation of behaviours. For example, in our case study we stratified our 4 830 264 VPs into 7 sub-populations, each one comprising a number of VPs ranging from 2 million to just 1. Each sub-population alone is representative of the entire spectrum of model behaviours (of course at different granularity). To design an ISCT proper trade-offs can be sought between the needed behavioural granularity and the budgeted computational effort.

Our *any-time* algorithm, based on *global search* guided by *statistical model checking* (along the lines of [87]), *intelligently explores* the (typically huge) model parameter space, collects those parameter assignments showing a *physiologically meaningful* behaviour (hence representing VPs), and organises them into strata, while guaranteeing a statistically-sound form of *graceful degradation*.

Note that, in many non-identifiable models (like our case-study HPG axis model), most model parameter values *might not* actually represent VPs, as, upon simulation, their associated model evolutions show-up to be physiologically meaningless or, anyway, out of interest. As anticipated in Section 1.2, this is due to, *e.g.*, over-parameterisation, presence of parameters whose values are not measurable through clinical assays (*e.g.*, reaction rates), presence of unknown (hence, not modelled) interdependency constraints among parameters, and use of parameters to define not-well-understood physiological mechanisms. To find parameter assignments yielding physiologically meaningful model behaviours and different phenotypes is thus computationally very hard, and naïve exploration or sampling of the parameter space could be hopeless. The high computational cost of generating a complete population of VPs for non-identifiable models is not surprising, and is indeed a well known problem when using a logic-based modelling approach (*e.g.*, [120]). The corresponding problem in that setting is searching for all attractors, which is a computationally very hard task, for which SAT or model checking–based techniques are typically employed (*e.g.*, [28]).

In our setting (*quantitative* non-identifiable VPH models), in order to automatically recognise physiologically meaningful model evolutions (and thus parameter assignments defining VPs), our approach envisions the *elicitation* and *formalisation* of *background biological and medical knowledge* (possibly *also* coming from available data). Our approach is fully independent of how such knowledge is formalised, as long as we can define a criterion that, given a parameter assignment (a candidate VP), decides whether the resulting model evolution is physiologically meaningful or not.

In our case study, we rely on background knowledge available in terms of known assignments to the model parameters (computed via parameter estimation against clinical data, hence defining *reference* VPs), bounds for model parameters and biological species, and on physiological meaningfulness criteria which ask for (loose) *qualitative similarity* of the model evolution under a candidate VP with respect to that entailed by some *reference* VPs. Such criteria are applicable to a wide class of models, *e.g.*, those defining hormonal signalling networks.

3.2 Methods

Below we define our formal framework and present our methodology to generate complete populations of VPs.

3.2.1 Virtual Patients, phenotypes and populations

As anticipated in Section 1.2, not all assignments to a VPH model parameters yield behaviours of interest. Many might even yield physiologically meaningless behaviours. Conversely, due to, *e.g.*, system over-parametrisation or non-identifiability, multiple parameter assignments may yield (almost) indistinguishable behaviours (*i.e.*, their associated evolutions are very similar on all inputs). Such indistinguishable VPs would increase the computational efforts needed to carry out an ISCT on the entire population, without bringing any advantage in terms of representativeness of the trial.

Forthcoming Definitions 3.1 to 3.3 define the concepts of VP, population of VPs, phenotype, and All-Different Phenotype Population (APP) for a given VPH model. These concepts rest on user-provided Boolean function φ and equivalence relation $\sim$. Boolean function φ defines the conditions to be met by any VPH model parameter λ for its evolutions to be considered as *physiologically meaningful* (hence λ has to be regarded as a VP). Equivalence relation $\sim$ on the set of VPs defines when two VPs shall be considered having indistinguishable behaviour (*i.e.*, showing the same *phenotype*): for any two VPs λ and λ', $\lambda \sim \lambda'$ means that the two VPs show the same phenotype.

Clearly, the definition of both function φ and relation $\sim$ depends on the VPH model at hand, and has to be made starting from expert knowledge. Also, when the model is subject to external inputs (*e.g.*, drug administrations), both φ and $\sim$ might need to be defined on model evolutions under *different input functions*. This allows the expert to define physiological meaningfulness and phenotypes of candidate VPs also in terms of their reactions under different patterns of drug administrations (where such reactions are dictated by the Pharmacokinetics/Pharmacodynamics, PK/PD model equations). In Section 3.2.3 we give a widely-applicable definition for φ and $\sim$ based on *qualitative similarity* of the model evolutions associated to different parameters.

Definition 3.1 (Virtual Patients)**.** *Let $\mathcal{S}$ be a VPH model (as in Definition 2.1) with parameter space Λ.*

Given a function φ of the form $\varphi : \Lambda \to \{0, 1\}$, the population $\hat{\Lambda}$ *of Virtual Patients (VPs) associated to $\mathcal{S}$ with respect to φ is:*

$$\hat{\Lambda} \;=\; \{\lambda \mid \lambda \in \Lambda \wedge \varphi(\lambda) = 1\}\,.$$

Definition 3.2 (Phenotypes)**.** *Let $\mathcal{S}$ be a VPH model (as in Definition 2.1) with parameter space Λ and $\hat{\Lambda}$ be a Virtual Patient (VP) population.*

Given an equivalence relation $\sim$ over $\hat{\Lambda}$, the phenotype space $\hat{\Lambda}/\!\sim$ *of $\hat{\Lambda}$ with respect to $\sim$ is the quotient set of $\hat{\Lambda}$ with respect to $\sim$:*

$$\hat{\Lambda}/\!\sim \;=\; \{[\lambda]_\sim \mid \lambda \in \hat{\Lambda}\}$$

where $[\lambda]_\sim$ is the equivalence class of λ. Any two VPs $\lambda, \lambda' \in \hat{\Lambda}$ such that $\lambda \sim \lambda'$ are said to have the same phenotype $[\lambda]_\sim = [\lambda']_\sim$.

Definition 3.3 (All-Different Phenotype Population)**.** *Let $\mathcal{S}$ be a VPH model (as in Definition 2.1) with parameter space Λ and $\hat{\Lambda}$ be a VP population.*

Given an equivalence relation $\sim$ over $\hat{\Lambda}$, an All-Different Phenotype Population (APP) of VPs *with respect to $\sim$ is any subset $\hat{\Lambda}^\sim$ of $\hat{\Lambda}$ such that no two VPs λ, λ' exist in $\hat{\Lambda}^\sim$ such that $[\lambda]_\sim = [\lambda']_\sim$.*

Definition 3.4 (Complete All-Different Phenotype Population)**.** *Let $\mathcal{S}$ be a VPH model (as in Definition 2.1) with parameter space Λ, $\hat{\Lambda}$ be a VP population and $\sim$ be an equivalence relation over $\hat{\Lambda}$.*

An APP $\hat{\Lambda}^\sim$ is said a Complete All-Different Phenotype Population (CAPP) *if $\hat{\Lambda}^\sim/\!\sim$ equals $\hat{\Lambda}/\!\sim$, i.e., $\hat{\Lambda}^\sim$ contains a representative of* all phenotypes *in the phenotype space of $\hat{\Lambda}$ with respect to $\sim$.*

Note that, when $\sim$ is $\mathbf{1}$ (*i.e.*, the equivalence relation defining a distinct class per VP $\lambda \in \hat{\Lambda}$), we have $\hat{\Lambda}/\mathbf{1} = \hat{\Lambda}$. Hence, the entire population of VPs ($\hat{\Lambda}$) can always be regarded as a CAPP.

3.2.2 Computing complete populations

Given a VPH model $\mathcal{S} = (\mathcal{T}, \Lambda, \mathcal{U}, \mathcal{Y}, \mathbf{y})$ with parameter space Λ, a function φ and an equivalence relation $\sim$ as in Definition 3.4, our goal is to compute a CAPP $\hat{\Lambda}^\sim \subseteq \hat{\Lambda} = \{\lambda \mid \lambda \in \Lambda \wedge \varphi(\lambda) = 1\}$.

In this thesis we focus on cases where the definition of VPH model $\mathcal{S}$, function φ, and the computation of the phenotype $[\lambda]_\sim$ of a VP λ (*i.e.*, the equivalence class of λ according to equivalence relation $\sim$) are too complex for set $\hat{\Lambda}^\sim$ to be computed analytically and/or symbolically in closed form. For such complex scenarios, deciding whether $\varphi(\lambda) = 1$ or not for any given $\lambda \in \Lambda$ (hence, whether λ represents a VP or not) and, in the affirmative case, computing its phenotype $[\lambda]_\sim$ involves a *numerical simulation* of $\mathcal{S}$ and the subsequent analysis of the resulting model evolutions under different inputs. Also, knowing that $\varphi(\lambda) = 1$ for some $\lambda \in \Lambda$ does not allow us to infer (without additional simulations) whether $\varphi(\lambda') = 1$ for other parameters $\lambda' \in \Lambda$, let alone their phenotypes.

In order to cope with such a general setting, we adopt a search-based approach that *explores* the model parameter space Λ looking for parameters $\lambda \in \Lambda$ such that $\varphi(\lambda) = 1$ and belonging to all-different equivalence classes of $\sim$. This calls for VPH models whose parameter space Λ is finite or can be *finitised* by the user, *e.g.*, into a bounded interval of $\mathbb{N}^k$, $k > 0$. Such finitisation can often be performed by exploiting knowledge about, *e.g.*, physiological bounds to the parameter values and *model locality* assumptions (*i.e.*, minor changes to the value of a parameter yield minor changes in the resulting trajectory).

Nevertheless, even when Λ is finite, an exhaustive exploration is practically infeasible unless $|\Lambda|$ is very small. Unfortunately, this is not the case for complex VPH models: for example, the size of the (finitised) parameter space of our case-study model is 1×10^{76}, which makes an exhaustive search clearly out of reach (let alone the fact that computing $\varphi(\lambda)$ for each λ takes seconds of simulation time).

To overcome these obstacles, our search (Section 3.2.2.1) is an *any-time algorithm* relying on Statistical Model Checking (SMC) and hypothesis testing to guarantee proper statistically-sound *graceful degradation*.

3.2.2.1 The algorithm

In the literature, SMC approaches (see [5] for a survey) are used for the formal (black-box) verification of large-scale systems having complex dynamics (such as, *e.g.*, hybrid systems [55]) and a huge state space. SMC uses simulations to compute system behaviours under given inputs and decides through statistical methods (*e.g.*, hypothesis testing) whether the system satisfies a given property or not (see, *e.g.*, [224, 46, 109]). In this setting, our algorithm is an *anytime* procedure which builds on the SMC and hypothesis testing method presented in [87] and extended in [205].

3.2.2.1.1 Core algorithm. Given a VPH model $\mathcal{S}$ having finite (although too large for an exhaustive exploration) parameter space Λ, and given function φ and equivalence relation $\sim$, our algorithm implements a *one-sided error* procedure to compute a CAPP $\hat{\Lambda}^\sim$ for $\mathcal{S}$ with respect to $\sim$. The algorithm randomly samples the parameter space Λ (according to a user-defined *sampling policy*), and iteratively adds to the current $\hat{\Lambda}^\sim$ (initialised to $\emptyset$) those parameters λ that represent VPs (*i.e.*, $\varphi(\lambda) = 1$) *and* show a phenotype different than all those already represented in $\hat{\Lambda}^\sim$.

The algorithm can be interrupted at *any time* and provides a form of *graceful degradation*: after each sample, the algorithm computes an upper bound $\varepsilon \in (0, 1]$ to the probability that further sampling would produce VPs of unseen phenotypes (*error margin*). This fact would prove that the current APP is not indeed a CAPP. When the achieved value for ε reaches a sufficiently-small (target) threshold, the user can decide to stop the algorithm and get the APP computed so far.

The computed value for ε is a function of the number of consecutive failed attempts N that the algorithm is experiencing in discovering VPs of new phenotypes. Clearly, being based on sampling, our algorithm can commit an error in computing the error margin ε (*i.e.*, it could return a value *lower* than a true upper bound). However, by exploiting statistical hypothesis testing methods, given *any* user-requested value $\delta \in (0, 1)$ (*confidence ratio*), our algorithm ensures (see forthcoming Theorem 3.1) that the probability of such an error is at most δ.

3.2.2.1.2 Sampling policy. In order to be effective in discovering VPs of new phenotypes, the employed sampling policy may embody proper *domain expert knowledge* and *structural knowledge* about the VPH model, for example: interdependency constraints among components of the parameter values (very common in over-parameterised models), or sensitivity information of model behaviours with respect to parameter values. Also, the sampling policy can be *refined* and *improved* during the search to embed new knowledge, *e.g.*, about the newly discovered VPs. In Section 3.2.3.4 we will outline a sampling policy for our case-study model (but widely applicable in general), which exploits the above flexibility.

3.2.2.1.3 Parallel computation. Our algorithm takes advantage of large parallel High Performance Computing infrastructures. The parameter space Λ is split upfront into k slices $\Lambda_1, \ldots, \Lambda_k$, and the different computational nodes run, in parallel, independent instances of our core algorithm, where process i ($i \in [1, k]$) draws samples from Λ_i to build population $\hat{\Lambda}_i^{\sim}$. When $\hat{\Lambda}_i^{\sim}$ is computed for all slices, a final population $\hat{\Lambda}^{\sim}$ is produced by taking the union of the phenotype spaces of all $\hat{\Lambda}_i^{\sim}$ and by choosing one representative VP from each equivalence class. To take load balancing into account, the number of slices (k) can be much higher (*e.g.*, one order of magnitude) than the number of available nodes. The k processes can then be dynamically suspended and brought back to execution by an orchestrator to keep the values of ε balanced among the different slices. This approach to parallelism and load balancing is very effective (see, *e.g.*, [146]) and avoids overhead due to inter-process communication (as that experienced in, *e.g.*, [150]).

3.2.2.1.4 Simultaneous computation of multiple stratified APPs. Our algorithm can work with multiple equivalence relations $\sim_1, \ldots, \sim_L$, defining different behavioural indistinguishability (*i.e.*, same phenotype) criteria, *e.g.*, at different levels of abstraction. When it makes sense to use the same policy to sample the VPH model parameter space Λ for all the $\sim_l$ ($l \in [1, L]$), then the L CAPPs can be computed simultaneously using the *same sequence* of random samples. In Section 3.2.3 we will exploit this possibility to compute a hierarchy of stratified CAPPs for our case-study VPH model.

3.2.2.1.5 Complete algorithm and main result. Let $\Lambda_1, \ldots, \Lambda_k$ be a partitioning of the finite (or finitised) parameter space Λ of our VPH model $\mathcal{S}$ (Definition 2.1) into $k > 0$ slices. Our overall algorithm runs in parallel k instances of Algorithm 1, where instance $i \in [1, k]$ runs on slice Λ_i of Λ and computes $L > 0$ APPs, one for each given equivalence relation $\sim_l$ ($l \in [1, L]$) on the population of VPs entailed by the given function φ (Definitions 3.1 and 3.3). During computation, each parallel branch (Algorithm 1) outputs (see Line 11) a stream of tuples of the form $(\sim_l, \hat{\Lambda}_i^{\sim_l}, \varepsilon_l)$ (one after each sample and for each equivalence relation $\sim_l$) stating that (see Theorem 3.1), with statistical confidence $(1 - \delta)$, the probability that further sampling within Λ_i will disprove that $\hat{\Lambda}_i^{\sim_l}$ is a CAPP of Λ_i with respect to $\sim_l$ is $< \varepsilon_l$.

Algorithm 1 includes (see Line 12) a periodic revision of the sampling policy in order to exploit the new acquired knowledge (of course at the price of resetting all counters N_l, $l \in [1, L]$).

```
1  function slice_APPs($\mathcal{S}$, $\Lambda_i$, $\varphi$, $\sim_1, \ldots, \sim_L$, $\delta$)
2      $\hat{\Lambda}_i^{\sim_1}, \ldots, \hat{\Lambda}_i^{\sim_L} \leftarrow \emptyset$;
3      $N_1, \ldots, N_L \leftarrow 0$;
4      while not interrupted do
5          $\lambda \leftarrow$ new sample from $\Lambda_i$ according to sampling policy;
6          foreach $l \in [1, L]$ do
7              if $\varphi(\lambda) = 1 \wedge [\lambda]_{\sim_l} \notin \hat{\Lambda}_i^{\sim_l}/\sim_l$ then
8                  add $\lambda$ to $\hat{\Lambda}_i^{\sim_l}$; $N_l \leftarrow 0$; $\varepsilon_l \leftarrow 1$
9              else
10                 $N_l$++; $\varepsilon_l \leftarrow 1 - \delta^{1/N_l}$
11             output $(\sim_l, \hat{\Lambda}_i^{\sim_l}, \varepsilon_l)$;
12         if sampling policy to be revised then
13             revise policy; $N_1, \ldots, N_L \leftarrow 0$
```

Algorithm 1: The i-th parallel branch ($i \in [1, k]$) of our any-time algorithm to compute multiple (*e.g.*, stratified) APPs.

Theorem 3.1. *Let $\mathcal{S}$ be a VPH model as in Definition 2.1 having finite parameter space Λ, φ and $\hat{\Lambda}$ be as in Definition 3.1 (with $\hat{\Lambda}$ unknown), $\sim_1, \ldots, \sim_L$ be $L > 0$ equivalence relations on $\hat{\Lambda}$, and $\delta \in (0, 1)$. Also, let $\Lambda_1, \ldots, \Lambda_k$ be a splitting of Λ into $k > 0$ slices.*

For all $i \in [1, k]$, $l \in [1, L]$, whenever the i-th branch of Algorithm 1 (working on Λ_i), outputs $(\sim_l, \hat{\Lambda}_i^{\sim_l}, \varepsilon_l)$, then, with probability $\geq (1 - \delta)$, value ε_l is an upper bound to the probability that further sampling will disprove that $\hat{\Lambda}_i^{\sim_l}$ is a CAPP of Λ_i with respect to $\sim_l$.

Proof. Assume that the branch of Algorithm 1 working on slice Λ_i ($i \in [1, k]$) outputs tuple $(\sim_l, \hat{\Lambda}_i^{\sim_l}, \varepsilon_l)$ with $l \in [1, L]$ and $\varepsilon_l < 1$ (in case $\varepsilon_l = 1$ the thesis trivially follows).

From Algorithm 1, ε_l is $1 - \delta^{1/N_l}$, where N_l is the number of the last consecutively drawn random samples that did not result in a VP with a phenotype not represented in the current APP $\hat{\Lambda}_i^{\sim_l}$.

Let $H_0(\hat{\Lambda}_i^{\sim_l}; \varepsilon_l)$ be the following *null hypothesis*: "the probability to disprove, by further sampling, that $\hat{\Lambda}_i^{\sim_l}$ is a CAPP is $\geq \varepsilon_l$".

To prove the thesis, it suffices to show that the probability that the algorithm rejects $H_0(\hat{\Lambda}_i^{\sim_l}; \varepsilon_l)$ when it actually holds (*type-I error*) is at most δ.

Indeed, if $H_0(\hat{\Lambda}_i^{\sim_l}; \varepsilon_l)$ holds, the probability to sample a VP of a phenotype not represented in $\hat{\Lambda}_i^{\sim_l}$ would be (by the hypothesis itself) $\geq \varepsilon_l$.

Hence, the probability that the experienced N_l failures to draw such a VP by means of N_l independent and identically distributed (iid) samples actually occur is $\leq (1 - \varepsilon_l)^{N_l}$.

Since $\varepsilon_l = 1 - \delta^{1/N_l}$ (Algorithm 1), we have $(1 - \varepsilon_l)^{N_l} = (\delta^{1/N_l})^{N_l} = \delta$. The thesis follows.

□

3.2.3 Computing complete stratified populations for the GynCycle VPH model

In this section we show how we instantiated the general methodology described in Section 3.2 to the complex state-of-the-art VPH model of the HPG axis (namely GynCycle, introduced in Section 2.2.1) in order to compute a *stratified* set of CAPPs.

Section 3.2.3.1 describes how we discretised the GynCycle parameter space. Section 3.2.3.2 describes how we defined physiological meaningfulness criteria to recognise VPs (*i.e.*, function φ); Section 3.2.3.3 describes how we defined *stratified* phenotypes (*i.e.*, equivalence relations $\sim_i$, $i \in [1, L]$, for a given number of strata $L \in \mathbb{N}_+$). Finally, Section 3.2.3.4 outlines our policy to sample (a slice of) the parameter space and its revision mechanism aimed at embedding in the sampling policy the new discovered knowledge.

We argue that the approach we used in GynCycle is based on general concepts which are applicable to a wide set of VPH models, *e.g.*, those defining hormonal signalling networks (see, *e.g.*, Type-I Diabetes Mellitus, T1DM, VPH models [110, 93, 53]).

In order to obtain robust results, we computed physiological admissibility metrics and phenotypes of model evolutions across 120 days (*i.e.*, as many as roughly 4 menstrual cycles), after ignoring the first 3 cycles (to get rid of any *transient model behaviours*, with this value being established by preliminary experiments). The *time quantum* between samples was set to 14.4 minutes (*i.e.*, 100 samples per day) to account for the physiological time scales of the modelled signalling pathways. Hence, input and observation functions are encoded as sequences of $h = 12\,000$ samples, one every 14.4 minutes.

3.2.3.1 Parameter space discretisation

To enable our SMC-based approach to computation of CAPPs, we need a finitisation of our parameter space. By preliminary experiments we assessed that a change of parameter values of $< 10\%$ yields very small changes in the resulting model trajectories (model locality). Hence, we discretised the interval for each parameter into 10 values and produced a finitised parameter space Λ of size 1×10^{76}. Although finite, this size is still too large for an exhaustive exploration to be performed. However, thanks to our informed sampling policy (Section 3.2.3.4), we were able to compute large APPs proved complete with a high statistical confidence (95%) and a small error margin (as low as 5×10^{-5}).

3.2.3.2 Physiological meaningfulness

As anticipated in Section 2.2.1, in [184, 61] GynCycle has been fitted against a database (courtesy of Pfizer) comprising 20–25 measures for 4 observed hormones (E2, P4, FSH and LH) on 12 healthy women, totalling more than 1000 measurements. This activity produced a parameter assignment $\lambda^{(0)} \in \Lambda$ which entails an observation function that *averages* the behaviours of those 12 patients (Figure 2.2).

In hormonal signalling pathways like those in GynCycle, all healthy humans show the *same qualitative behaviour* of such hormones. Hence, $\lambda^{(0)}$ defines a VP that we can (and do) regard as a *reference VP*. We thus defined function φ (which encodes the physiological meaningfulness criteria that must be satisfied by a parameter assignment λ for it to be considered a VP, see Definition 3.1) asking for *qualitative similarity* between the model observation function under λ and under $\lambda^{(0)}$. Namely, we proceed as outlined in the following sections.

3.2.3.2.1 Representative portfolio of input functions. In order to derive VPs whose behaviour is meaningful also when drugs are administered, we defined a *representative portfolio* **U** of 5 different input functions.

Beyond the no-drug input (under which the GynCycle observation function must show a behaviour similar to that in Figure 2.2, hence representing a healthy natural menstrual cycle), we considered the following standard treatment strategies, consisting of daily administrations of different doses for each of the two pharmaceutical compounds supported by the model: Triptorelin and Norethisterone, whose effect is the inhibition of the regular menstrual cycle. These strategies are common in fertility treatments.

Our portfolio **U** of input functions is defined as follows:

1. No drug;
2. A standard dose of Triptorelin (0.1 mg) for 28 days starting from cycle day 31;
3. 50% of a standard dose of Triptorelin (0.05 mg) for 28 days starting from cycle day 31;
4. A standard dose of Norethisterone (10 mg) for 10 days starting from cycle day 25;
5. 50% of a standard dose of Norethisterone (5 mg) for 10 days starting from day 25.

3.2.3.2.2 Physiological meaningfulness as qualitative similarity. Our function φ returns 1 on $\lambda \in \Lambda$ (thus declaring λ to be a VP), if any only if the model observation functions under λ, when subject to *each* of the input functions in **U**, have values always within certain *physiological bounds*, and can be *time-scaled* and/or *time-shifted* (up to a certain limit) so to satisfy certain *qualitative similarity metrics*, when compared to the observation functions entailed by the reference VP $\lambda^{(0)}$ under the *same* input. Time shifting and scaling allow us to deal with, respectively, time-alignment issues and different menstrual cycle durations.

The qualitative similarity metrics we exploited are standard (discrete-time) *signal processing metrics* (see, *e.g.*, [213]): the *normalised zero-lag cross-correlation* and the *normalised energy difference*, which we require to be, respectively, above and below certain thresholds.

Intuitively, a high-enough normalised zero-lag cross-correlation between two functions shows that they share peaks and valleys (this metrics takes its maximum value 1 when the two functions differ only by a multiplicative factor), while a low-enough normalised energy difference shows that the two functions have a similar energy.

For our case study, in our forthcoming experiments we set our thresholds to 70% and 80% for the normalised zero-lag cross-correlation and the normalised energy difference, respectively. We also set limits for time-scaling and time-shifting to, respectively, $\pm 10\%$ and 35 days. Such values (defined after preliminary experiments) are generous enough to allow us to accept model evolutions quite different from the evolutions entailed by the reference VP, but still appearing physiologically meaningful to a visual inspection.

In the following sections we formally define our metrics and describe technical details on how $\varphi(\lambda)$ is actually computed, for any given $\lambda \in \Lambda$.

Formal definition We formally define such metrics as follows.

Let $\mathcal{S} = (\mathcal{T}, \Lambda, \mathcal{U}, \mathcal{Y}, \mathbf{y})$ (Definition 2.1) be a VPH model, whose output space $\mathcal{Y}$ is $\mathbb{R}^n_{0+}$ where $n \in \mathbb{N}_+$ is the number of (real-valued) observables.

Given a time function $\mathbf{f} : \mathcal{T} \to \mathcal{W}$ (with $\mathcal{W}$ being any non-empty set of values) and $\alpha \in \mathbb{R}_+$ and $\tau \in \mathbb{R}_{0+}$, we denote with $\mathbf{f}^{\alpha,\tau} : \mathcal{T} \to \mathcal{W}$ the time function defined, for all time points $t \in \mathcal{T}$, as:

$$\mathbf{f}^{\alpha,\tau}(t) = \mathbf{f}(\alpha t + \tau).$$

Thus, $\mathbf{f}^{\alpha,\tau}$ is function $\mathbf{f}$ time-shifted by τ and time-stretched by factor α.

Let also $\lambda^{(0)} \in \Lambda$ be a distinguished VP of $\mathcal{S}$ (the *reference VP* as for GynCycle) and $\mathbf{U} \subset \mathcal{U}^{\mathcal{T}}$ be our portfolio of representative input functions for $\mathcal{S}$ (for GynCycle, the portfolio defined in Section 3.2.3.2.1).

Given a parameter assignment $\lambda \in \Lambda$, values for $\alpha \in \mathbb{R}_+$ (time stretch) and $\tau \in \mathbb{R}_{0+}$ (time shift), a model observable i and an input function $\mathbf{u} \in \mathbf{U}$, we define the following two qualitative similarity metrics between the $\mathcal{S}$ evolution of observable i under parameter λ (time-shifted by τ and time-stretched by α) and reference parameter $\lambda^{(0)}$, when subject to the same input function $\mathbf{u}$:

Normalised zero-lag cross-correlation

$$\rho_{\mathbf{u},\lambda^{(0)},\lambda,i}(\alpha,\tau) = \frac{\sum_{t\in\mathcal{T}} \mathbf{y}_i^{\alpha,\tau}(t;\ \mathbf{u},\lambda) \times \mathbf{y}_i(t;\ \mathbf{u},\lambda^{(0)})}{||\mathbf{y}_i^{\alpha,\tau}(\mathbf{u},\lambda)|| \times ||\mathbf{y}_i(\mathbf{u},\lambda^{(0)})||} \tag{3.1}$$

Normalised Energy Difference (NED)

$$E_{\mathbf{u},\lambda^{(0)},\lambda,i}(\alpha,\tau) = \frac{\left| ||\mathbf{y}_i^{\alpha,\tau}(\mathbf{u},\lambda)||^2 - \left|\left|\mathbf{y}_i(\mathbf{u},\lambda^{(0)})\right|\right|^2 \right|}{\left|\left|\mathbf{y}_i(\mathbf{u},\lambda^{(0)})\right|\right|^2} \tag{3.2}$$

As an example, Figure 3.1 shows how a model evolution (the blue curve) can be time-shifted and -scaled (the dashed blue curve) in order to be aligned to a reference model evolution (the green curve). As the picture shows, after being aligned the two curves exhibit a high correlation. Also, the figure highlights the difference of energies of the two curves, which, after normalisation, is used in our qualitative similarity metrics to understand how close is the two model evolutions are in terms of their energy.

Definition 3.5 (Similarity-based physiological admissibility function)**.** *Let* $\mathcal{S} = (\mathcal{T}, \Lambda, \mathcal{U}, \mathcal{Y}, \mathbf{y})$ *(Definition 2.1) be a VPH model, whose output space* $\mathcal{Y}$ *is* $\mathbb{R}^n_{0+}$ *where* $n \in \mathbb{N}_+$ *is the number of (real-valued) observables.*

Let also $\theta_\rho, \theta_E \in \mathbb{R}_+$ *be two* thresholds*,* $\mathbb{A}$*,* $\mathbb{T}$ *and* $\mathbb{B}_1, \ldots \mathbb{B}_n$ *be bounded ranges of* $\mathbb{R}_{0+}$ *(*time-stretch*,* time-shift *and* observable bounds *ranges) such that* $1 \in \mathbb{A}$ *and* $0 \in \mathbb{T}$*.*

We define our similarity-based physiological admissibility function $\varphi : \Lambda \to \{0, 1\}$ *as follows. For any* $\lambda \in \Lambda$*,* $\varphi(\lambda) = 1$ *if there exists* $\alpha \in \mathbb{A}$*,* $\tau \in \mathbb{T}$ *such that, for each model observable* $i \in [1, n]$ *and* $\mathbf{u} \in \mathbf{U}$*,* all *conditions below hold:*

- $\rho_{\mathbf{u},\lambda^{(0)},\lambda,i}(\alpha, \tau) \geq \theta_\rho$
- $E_{\mathbf{u},\lambda^{(0)},\lambda,i}(\alpha, \tau) \leq \theta_E$
- $[\min_{t\in\mathcal{T}}(\mathbf{y}_i^{\alpha,\tau}(t;\ \mathbf{u}, \lambda)), \max_{t\in\mathcal{T}}(\mathbf{y}_i^{\alpha,\tau}(t;\ \mathbf{u}, \lambda))] \subseteq \mathbb{B}_i$.

We have $\varphi(\lambda) = 0$ *otherwise.*

In other words, Definition 3.5 declares a parameter value λ as physiologically meaningful (*i.e.*, a VP) if the entailed model evolution can be time-scaled and/or -shifted so that all observables are within the given physiological bounds and the values of all similarity metrics are within the given thresholds.

Computation Computing $\varphi(\lambda)$ for any given λ is quite heavy. In particular, GynCycle must be numerically simulated under each candidate parameter λ and each input function $\mathbf{u} \in \mathbf{U}$, in order to retrieve the observation function $\mathbf{y}(\mathbf{u}, \lambda)$. Also, time-scaling and time-shifting issues must be evaluated before computing our similarity metrics between $\mathbf{y}(\mathbf{u}, \lambda)$ and $\mathbf{y}(\mathbf{u}, \lambda^{(0)})$. Hence, we need to find suitable values for α and τ, which, however, are *real* values. To cope with this problem, we deployed the following (greedy) approach, starting from the model evolutions computed by the simulator under parameter λ and $\lambda^{(0)}$ and all input functions in $\mathbf{U}$: we considered only the (α, τ) pairs for which $\mathbf{y}_i^{\alpha,\tau}(t;\ \mathbf{u}, \lambda)$ and $\mathbf{y}_i(t;\ \mathbf{u}, \lambda^{(0)})$ share, in the time domain, at least a *local maximum* or a *local minimum* for at least one $\mathbf{u} \in \mathbf{U}$.

Since we defined the time set of GynCycle as a bounded interval of $\mathbb{N}$ (see, Section 2.2.1), we can efficiently enumerate all possible values for (α, τ) (which represent all possible peak-alignments) by defining and computing all solutions of a Constraint

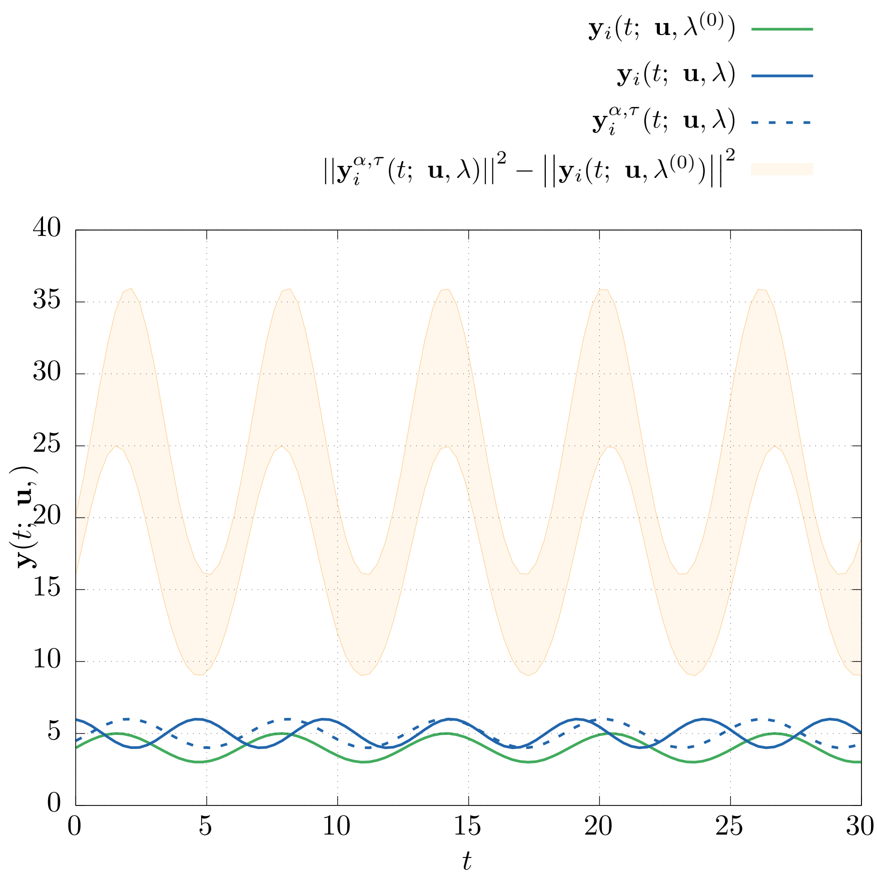


Figure 3.1. An example of two model evolutions representing $\mathbf{y}_i(t;\ \mathbf{u}, \lambda^{(0)})$ (green curve) and $\mathbf{y}_i(t;\ \mathbf{u}, \lambda)$ (blue curve), respectively. The dashed curve is the time-scaled and -shifted version of $\mathbf{y}_i(t;\ \mathbf{u}, \lambda)$. The yellow area denotes the difference of model evolution energy.

Satisfaction Problem (CSP) (by means of an off-the-shelf solver). Namely, for each candidate assignment to α and τ (a solution of our CSP), we compute our qualitative similarity metrics, and return $\varphi(\lambda) = 1$ as soon as we find one assignment for which the qualitative similarity metrics satisfy the given thresholds (and 0 if we do not find any).

3.2.3.3 Stratified phenotypes

Our definition of behavioural indistinguishability (*i.e.*, same-phenotype equivalence relation) of different VPs follows an approach consistent to the one we used to decide physiological meaningfulness. However, in this case, similarity is *quantitatively* evaluated between the observation functions of each pair of VPs (*i.e.*, parameters that, by satisfying function φ in Section 3.2.3.2, already satisfy the qualitative similarity metrics thresholds against the *reference VP* $\lambda^{(0)}$).

To compare two observation functions available in the form of discrete sequences of real-valued samples evenly spaced in time (as is our case), we compare the coefficients of their Discrete Fourier Transforms (DFTs) (see, *e.g.*, [213] and citations therein, and, *e.g.*, [163] for an application to healthcare). We thus define the equivalence relation $\sim_\psi$ (parametric in $\psi \in \mathbb{R}_+$, *quantisation factor*) among the VPs as stated by Definition 3.6.

Definition 3.6 (Equivalence relation $\sim_\psi$)**.** *Let $\mathcal{S} = (\mathcal{T}, \Lambda, \mathcal{U}, \mathcal{Y}, \mathbf{y})$ be a dynamical system with $\mathcal{T} = \mathbb{N}_+$, parameter space Λ, and $\mathcal{Y} = \mathbb{R}^n_{0+}$, where $n \in \mathbb{N}_+$ is the number of model observables. Let also $\lambda^{(0)} \in \Lambda$ be a distinguished parameter assignment, $\psi \in \mathbb{R}_+$ and $\mathbf{U} \subseteq \mathcal{U}^\mathcal{T}$ be a set of input functions for $\mathcal{S}$.*

Given $\lambda^{(1)}, \lambda^{(2)} \in \Lambda$, let $Y^{(k)}_{c,i,\mathbf{u}}$ be the c-th coefficient ($c \in [0, h-1]$) of the DFT of the time evolution of the i-th model observable ($i \in [1, n]$) when the model is fed with input $\mathbf{u} \in \mathbf{U}$ under parameter $\lambda^{(k)}$ ($k \in [0, 2]$, which refers to the distinguished $\lambda^{(0)}$ when $k = 0$).

The quantum *of $Y^{(k)}_{c,i,\mathbf{u}}$, $Q_\psi(Y^{(k)}_{c,i,\mathbf{u}})$, is:* $\left\lfloor \frac{\left|Y^{(k)}_{c,i,\mathbf{u}}\right|^2}{\psi \left|\left|\mathbf{y}_i(\mathbf{u}, \lambda^{(0)})\right|\right|^2} \right\rceil$.

We stipulate that $\lambda^{(1)} \sim_\psi \lambda^{(2)}$ if, for each $\mathbf{u} \in \mathbf{U}$, $c \in [0, h-1]$, and $i \in [1, n]$, it holds $Q_\psi(Y^{(1)}_{c,i,\mathbf{u}}) = Q_\psi(Y^{(2)}_{c,i,\mathbf{u}})$.

Equivalence relation $\sim_\psi$ (Definition 3.6) is defined in terms of the energy of the observation function of each model observable $i \in [1, n]$ under the distinguished parameter assignment $\lambda^{(0)}$ $\left(\left|\left|\mathbf{y}_i(\mathbf{u}, \lambda^{(0)})\right|\right|^2\right)$, which acts as a *normalising factor*. This is important, because the different model observables may assume values in very different ranges. In our experiments (Section 3.3) $\lambda^{(0)}$ is the GynCycle *reference VP*.

The following Remark 3.1 shows that our definition of $\sim_\psi$ (Definition 3.6) immediately gives us a theoretical upper bound to the Normalised Energy Difference (NED) shown by the observation functions of any two VPs, $\lambda^{(1)}$ and $\lambda^{(2)}$, belonging to the same equivalence class of $\sim_\psi$, $\lambda^{(1)} \sim_\psi \lambda^{(2)}$, for any model observable $i \in [1, n]$ and input function $\mathbf{u} \in \mathbf{U}$.

Remark 3.1. *Let $\mathcal{S} = (\mathcal{T}, \Lambda, \mathcal{U}, \mathcal{Y}, \mathbf{y})$, $\lambda^{(0)}$, n, ψ, $\mathbf{U}$, and $\sim_\psi$ be as in Definition 3.6. For each $i \in [1, n]$ $\mathbf{u} \in \mathbf{U}$, and $\lambda \in \Lambda$, let $\mathbf{y}_i(\mathbf{u}, \lambda)$ be the (real-valued) observation function of the i-th model observable under parameter λ when $\mathcal{S}$ is fed with input function $\mathbf{u}$.*

For every $\lambda^{(1)}, \lambda^{(2)} \in \Lambda$ such that $\lambda^{(1)} \sim_\psi \lambda^{(2)}$ we have that, for each $i \in [1, n]$ and $\mathbf{u} \in \mathbf{U}$, the Normalised Energy Difference (NED) between $\mathbf{y}_i(\mathbf{u}, \lambda^{(1)})$ and $\mathbf{y}_i(\mathbf{u}, \lambda^{(2)})$ is upper-bounded by ψ, that is:

$$\frac{\left| \left|\left|\mathbf{y}_i(\mathbf{u}, \lambda^{(1)})\right|\right|^2 - \left|\left|\mathbf{y}_i(\mathbf{u}, \lambda^{(2)})\right|\right|^2 \right|}{\left|\left|\mathbf{y}_i(\mathbf{u}, \lambda^{(0)})\right|\right|^2} \leq \psi.$$

Proof. Time evolution $\mathbf{y}_i(\mathbf{u}, \lambda^{(k)})$ ($k \in [0, 2]$) is a vector of real-valued samples $\mathbf{y}_i(\mathbf{u}, \lambda^{(k)}) = (y^{(k)}_{t,i,\mathbf{u}})_{t=0}^{h-1} \in \mathbb{R}^h_{0+}$.

Let $Y^{(k)}_{i,\mathbf{u}} = (Y^{(k)}_{c,i,\mathbf{u}})_{c=0}^{h-1} \in \mathbb{C}^h$ be the vector of coefficients of the DFT of $\mathbf{y}_i(\mathbf{u}, \lambda^{(k)})$, each one being a complex number. Namely: $Y^{(k)}_{c,i,\mathbf{u}} = \sum_{t=0}^{h-1} y^{(k)}_{t,i,\mathbf{u}} \exp(-\frac{2\pi \jmath t c}{h})$, where $\jmath = \sqrt{-1}$.

We recall that, given vector $\mathbf{X} = (X_c)_{c=0}^{h-1} \in \mathbb{C}^h$ (or, as a special case, $\mathbf{X} \in \mathbb{R}^h_{0+}$), the squared L2-norm of $\mathbf{X}$ is $||\mathbf{X}||^2 = \sum_{c=0}^{h-1} |X_c|^2$.

From Parserval's theorem we know (see, *e.g.*, [213]) that

$$\left|\left|\mathbf{y}_i(\mathbf{u}, \lambda^{(k)})\right|\right|^2 = \frac{1}{h} \left|\left|Y^{(k)}_{i,\mathbf{u}}\right|\right|^2 .$$

Hence, $\left| \left|\left|\mathbf{y}_i(\mathbf{u}, \lambda^{(1)})\right|\right|^2 - \left|\left|\mathbf{y}_i(\mathbf{u}, \lambda^{(2)})\right|\right|^2 \right|$ rewrites as: $\frac{1}{h} \sum_{c=0}^{h-1} \left| \left|Y^{(1)}_{c,i,\mathbf{u}}\right|^2 - \left|Y^{(2)}_{c,i,\mathbf{u}}\right|^2 \right|$.

Since $\lambda^{(1)} \sim_\psi \lambda^{(2)}$, we have, for each $c \in [0, h-1]$, $Q_\psi(Y^{(1)}_{c,i,\mathbf{u}}) = Q_\psi(Y^{(2)}_{c,i,\mathbf{u}})$, which implies (Definition 3.6) that

$$\left| \left|Y^{(1)}_{c,i,\mathbf{u}}\right|^2 - \left|Y^{(2)}_{c,i,\mathbf{u}}\right|^2 \right| \leq \psi \left|\left|\mathbf{y}_i(\mathbf{u}, \lambda^{(0)})\right|\right|^2$$

for all $c \in [0, h-1]$.

Thus:

$$\left| \left|Y^{(1)}_{i,\mathbf{u}}\right|^2 - \left|Y^{(2)}_{i,\mathbf{u}}\right|^2 \right| \leq h\psi \left|\left|\mathbf{y}_i(\mathbf{u}, \lambda^{(0)})\right|\right|^2$$

and the thesis follows.

□

In the formula, the energy difference is normalised with respect to the energy of the same observable under the *reference VP* $\lambda^{(0)}$, when subject to the same input.

Thus, by considering L increasing values for ψ: $\psi_1 < \cdots < \psi_L$ ($L \in \mathbb{N}_+$), we define L equivalence relations $\sim_{\psi_1}, \ldots, \sim_{\psi_L}$ that group VPs in larger behavioural indistinguishability classes as their associated quantisation factor increases (*stratified phenotypes*). In our experiments (Section 3.3), we choose $L = 7$ and an increasing set of 7 values for ψ (see Table 3.1), where ψ_L is such to place all generated VPs into a *single* equivalence class. Indeed, value ψ is a *very loose* upper bound for the NED between VPs belonging to the same equivalence class. This is because it does not take into account the fact that all our VPs are known to satisfy the physiological meaningfulness criteria of Section 3.2.3.2.2 (qualitative similarity with respect to to the time evolution of $\lambda^{(0)}$). In particular, since such criteria depend on optimal time-shifts and time-stretches sought *for each single* VP, our bound to the NED cannot exploit such knowledge and needs to stick to the worst-case. To this end, in our experimental analysis, we also compute, by means of auxiliary hypothesis testing–based SMC tasks (along the same lines of Algorithm 1, with error margin 1% and confidence ratio 5%), the *actual* maximum NED between VPs belonging to the same equivalence class of each stratum (see Table 3.1).

3.2.3.4 Sampling policy and parallel computation

As many VPH models, GynCycle is organised in several components, one for each of the modelled hormones. Changing the values of the elements of the parameter vector occurring in a few components typically changes the overall model dynamics only partially.

This key observation is at the hearth of our sampling policy. In order to draw, with high probability, a parameter assignment that proves to be a VP, we exploit the knowledge acquired so far in terms of the parameter assignments that already proved to be VPs. Namely, given the set $\hat{\Lambda}_{\text{current}}$ of VPs discovered so far (*population of known VPs*), our sampling policy draws a random parameter λ by changing uniformly at random the elements occurring in $p \in \mathbb{N}_+$ model components (chosen uniformly at random) from a parameter $\hat{\lambda}$ chosen uniformly at random from $\hat{\Lambda}_{\text{current}}$ (if $\hat{\Lambda}_{\text{current}}$ is empty, then $\hat{\lambda} = \lambda^{(0)}$). The value of p is drawn from a Zipf's distribution (which is a bounded discrete distribution based on a power law, *i.e.*, $p \sim ap^{-b}$, where a is a normalisation factor), in order to draw with high probability small values of p. In our experiments we set b to 3 so that the expected value for p is about 1.11.

The sampling policy is periodically revised by updating $\hat{\Lambda}_{\text{current}}$ with the new discovered VPs. However, in order to avoid too frequent policy updates (which would resort in an immediate reset of the consecutive failure counters, see Section 3.2.2.1), set $\hat{\Lambda}_{\text{current}}$ is updated only every a given number N of samples. In our experiments we chose N such that experiencing N consecutive failures to find a new VP (regardless of its phenotype) would allow us to conclude, with statistical confidence $(1 - \delta) = 95\%$, that the probability that additional VPs will be found by further sampling is less than $\varepsilon = 1 - \delta^{1/N} = 5 \times 10^{-5} = 0.005\%$. This results in $N = 59\,914$.

For the above sampling policy to work on top of a slicing of the parameter space Λ to be processed in parallel, it is enough to ensure that $\lambda^{(0)}$ belongs to all slices. This was done by defining our (initially continuous) parameter space finitisation as a grid having $\lambda^{(0)}$ as one of its vertices, and by defining the k slices by bisecting Λ on values $\lambda^{(0)}_{i_1}, \ldots, \lambda^{(0)}_{i_r}$ for any subset of coordinates $i_1, \ldots, i_r$ within $[1, 76]$, thus defining $k = 2^r$ slices $\Lambda_1, \ldots, \Lambda_k$ all containing $\lambda^{(0)}$. In our experiments we chose $r = 7$ random coordinates, hence $k = 128$.

3.3 Experimental Results

Here we present our results on GynCycle. In Section 3.3.1 we show the APPs we computed, in Section 3.3.2 we analyse the behaviour of our sampling policy, and in Section 3.3.3 we perform a qualitative and quantitative evaluation of the representativeness of our populations with respect to retrospective clinical data (86 medical cases courtesy of Hannover Medical School, University Hospital of Lausanne, and Pfizer).

3.3.1 Computed All-Different Phenotype Populations

We ran our SMC-based algorithm on a parallel High Performance Computing (HPC) infrastructure (the Marconi cluster at Cineca, Italy) with the settings defined above, in order to compute the stratified APPs as defined in Section 3.2.3.3. The confidence ratio δ was set to 0.05.

The computation was stopped after around 60 days. In total, our algorithm sampled 414 245 648 parameters (simulating GynCycle for 7 months on each of them and for each of the input functions in the representative portfolio described in Section 3.2.3.2.1). Overall, 4 830 264 parameters were declared to define VPs.

Table 3.1 lists the sizes of the 7 computed APPs. The bottom line refers to the entire population of VPs, $\hat{\Lambda}^{\mathbf{1}}$ (which is an APP with respect to equivalence relation $\mathbf{1}$).

We decided to terminate our (any-time) computation when we achieved $\varepsilon = 5 \times 10^{-5}$ for *all slices* on the *top three* strata. This means that, by Theorem 3.1, with statistical confidence $1 - \delta = 95\%$, the probability that further sampling (in *any* single slice) would disprove that such top three APPs are indeed CAPPs is below the error margin (5×10^{-5}).

As for the remaining strata, the table reports the minimum, maximum and average error margins across the $k = 128$ parallel SMC processes (one per slice) at the time of termination of our any-time computation. Since the exploration of each slice is an independent process, the values of the k error margins for each stratum can be quite different, as the value for ε for a given slice depends on the time when the last VP belonging to *that* slice was generated. Also, when we terminated our experiment, a new VP (of a phenotype *known* to the top three strata) was *just* generated. Hence, the max ε for the population $\hat{\Lambda}^{\mathbf{1}}$ consisting of *all* VPs (bottom line of Table 3.1) is 1.

Figure 3.2 shows the time evolutions for the GynCycle observables under the VPs belonging to the computed APPs for all strata except the extreme two. It can be seen that, despite the number of VPs greatly reduces at higher levels of our stratification, all APPs retain full *representativeness* of the entire spectrum of possible behaviours (this is also a consequence of the small values for the NED in all strata).

A final note is in order. Although 60 days could appear an unusually-long time for a computation (especially if compared to the time typically needed by classical model fitting tasks), this is a *one-time activity* for the input VPH model, and can be sped-up almost arbitrarily by using more computational nodes (*e.g.*, if using 1280 nodes –which is perfectly feasible in today's infrastructure-as-a-service platforms– with groups of 10 nodes jointly exploring each of our 128 slices, would have required just 6 days). Indeed, once a population of VPs for a given model has been computed, it can be used to carry-out *multiple* ISCT (*i.e.*, for different treatment strategies or medical devices). Each ISCT can be carried-out on the *most appropriate stratum* of VPs, depending, *e.g.*, on the chosen trade-off between budgeted computational effort and required completeness of the trial. Also, more sophisticated approaches can be exploited, *e.g.*, *iterative deepening* within the stratification of phenotypes (guided by simulation results) searching for a VP showing a failure of the candidate treatment or medical device (a *counter-example*, see, *e.g.*, [140]).

3.3.2 Sampling policy behaviour

Our *informed* sampling policy was able, on average, to find an admissible VP every 86 attempts (average success rate: 1.17%). This is to be compared to a *uniform (non-informed) sampling policy*, which was *unable* to discover a single VP after *50 million attempts* (within our 128 slices).

Figure 3.3 shows the behaviour of the error margin $\varepsilon = 1 - \delta^{1/N}$ values achieved by our informed sampling policy during the computation of our APPs. The figure shows the case of $\hat{\Lambda}^{\sim 4050}$, *i.e.*, the APP associated to the smallest value of ψ (see Table 3.1) for which we reached a value of $\varepsilon = 5 \times 10^{-5}$ for all slices. The plot shows the values for ε reached by each of the 128 parallel computations (light

id	ψ	APP size	error margin (ε)			max NED
			min	avg	max	
7	16 200	1	5×10^{-5}	5×10^{-5}	5×10^{-5}	163.23%
6	8100	104	5×10^{-5}	5×10^{-5}	5×10^{-5}	144.36%
5	4050	3862	5×10^{-5}	5×10^{-5}	5×10^{-5}	106.74%
4	2700	43 941	5×10^{-5}	6.75×10^{-3}	4.51×10^{-1}	84.70%
3	1800	251 239	5.09×10^{-4}	2.36×10^{-2}	1	59.09%
2	900	2 136 710	3.25×10^{-3}	8.33×10^{-2}	1	48.07%
1	–	4 830 264	9.87×10^{-3}	1.81×10^{-1}	1	–

Table 3.1. Stratified GynCycle APPs for different behavioural indistiguishability criteria ($\sim_\psi$). Statistical confidence: 95%.

curves) when discovering each of its VPs (x axis), thus disproving that the current APP was indeed a CAPP. Values for x have been normalised into percentages of the total number of the VPs discovered by each parallel computation. The dark curve shows the average ε among such computations.

We note that the average ε lies for most of the time at values *one order of magnitude higher* than the value we chose to terminate our experiments ($\varepsilon = 5 \times 10^{-5}$, see Table 3.1). This shows that our informed sampling policy was *always* effective to extract (with probability much higher than 5×10^{-5}) new VPs when (we know that) they actually exist.

Finally, Figure 3.4 shows the location of VPs within the GynCycle parameter space. The figure shows one line per VP, which connects the chosen values for the 76 parameter vector elements. Interestingly, for some of them, only a few values of their domains actually occur in VPs. Such constraints were *unknown* at the time of model design.

3.3.3 Validation against clinical data

Here we assess the representativeness of our entire population of VPs (*i.e.*, $\hat{\Lambda}^{\mathbf{1}}$) with respect to retrospective clinical data on 86 clinical records kindly made available to us by Hannover Medical School (35 cases), University Hospital of Lausanne (39 cases) and Pfizer (12 cases, which were originally used in [184, 61] to compute the reference GynCycle VP). As anticipated in Section 2.2.3, in each dataset, for each clinical record we have actual measurements of the blood levels of the 4 model observables (LH, FSH, E2, and P4) on a (roughly) daily basis for an entire menstrual cycle (all medical cases refer to patients subject to no pharmaceutical treatment). Below we perform both a qualitative and a quantitative assessment of the representativeness of our VP population against such datasets.

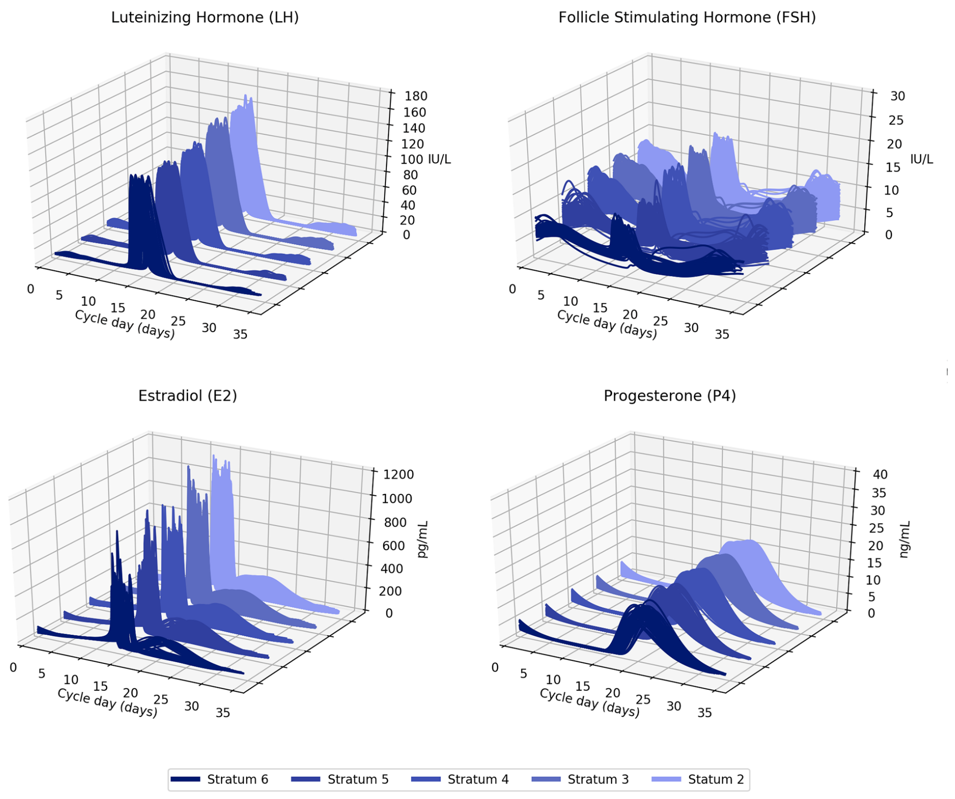


Figure 3.2. Time evolutions for the GynCycle observables under the VPs belonging to the computed stratified APPs

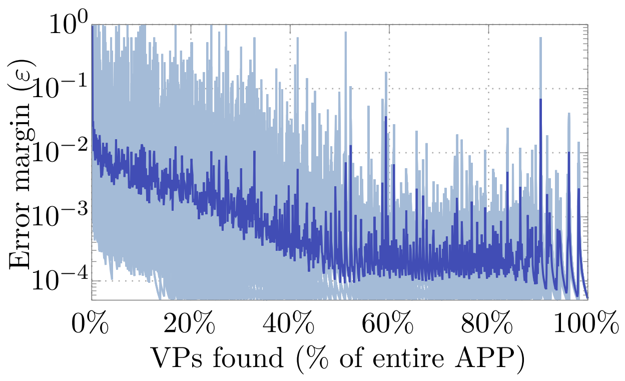


Figure 3.3. Average error margin (ε) reached during parallel computations (stratum 5)

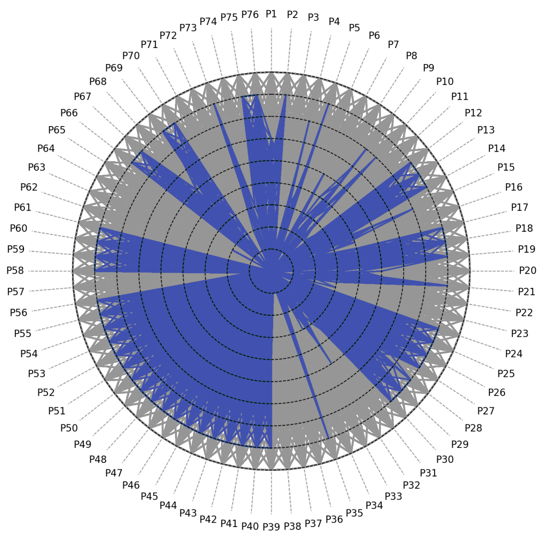


Figure 3.4. Parameter space exploration

3.3.3.1 Qualitative evaluation

Figure 3.5 shows daily measured blood hormone levels on the 86 clinical records (box-and-whisker plots) on top of the set of time evolutions of the 4 model observables of our entire population of VPs (blue curves). Curves as well as data have been aligned with respect to the LH peak, which is typically used to estimate the ovulation day. This allows to take into account the different transient periods of model trajectories stemming from the different VPs in our population.

Figure 3.5 shows that our VP population is highly representative of the available clinical measurements, and that the qualitative time evolutions of our VPs faithfully reflect those of the available data.

3.3.3.2 Quantitative evaluation

We formalise each clinical record according to Definition 2.2 introduced in Section 2.1.2. The Average Normalised Mean Absolute Error $\texttt{aNMAE}(C, \lambda)$ between a clinical record C and VP λ is the average (among observables s) of the average (among clinical measurements j of s) of the absolute error between the model output (under parameter λ and input function $\mathbf{u}_C$ defining the same sequence of drug administrations of C) and the j-th measurement of s in C, normalised with respect to the maximum measured value $\texttt{maxval}(s)$ of s in C. Namely:

$$\texttt{avg}_s\texttt{avg}_j \frac{|\mathbf{y}(\texttt{time}(s, j);\ \mathbf{u}_C, \lambda) - \texttt{val}(s, j)|}{\texttt{maxval}(s)}.$$

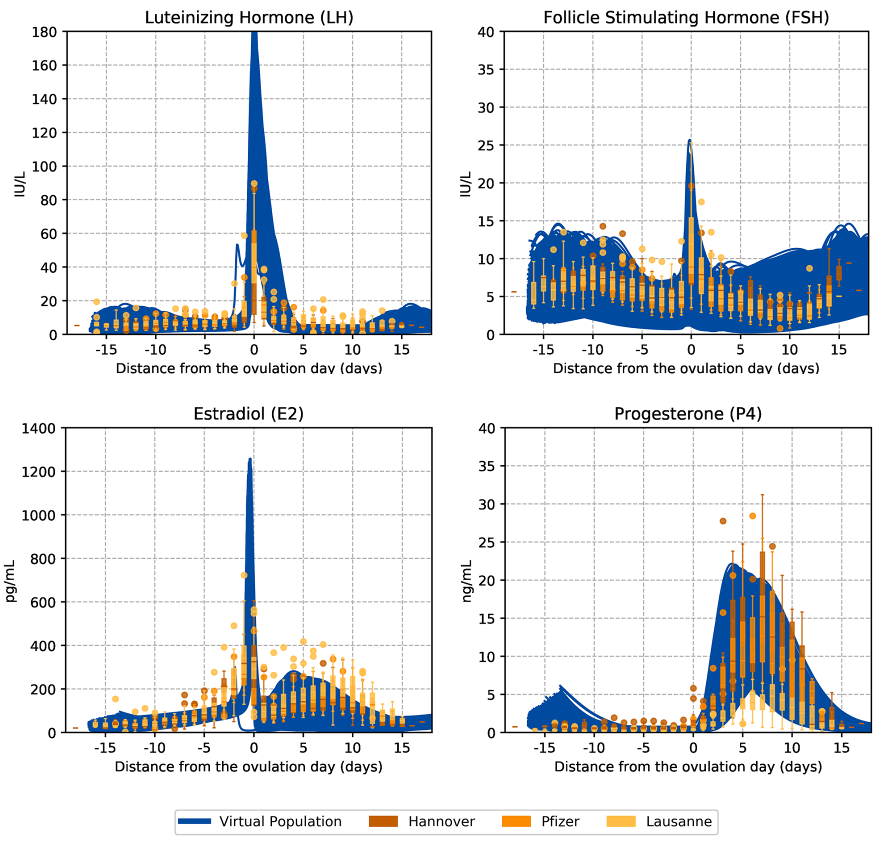


Figure 3.5. Qualitative validation of our GynCycle population against clinical data.

We recall that $\mathbf{u}_C$ always defines absence of treatment (consistently with our datasets) and that we aligned model evolutions and data at the LH peak day.

We say that a clinical record C is *covered* by our VP population $\hat{\Lambda}$ *within* aNMAE e if there exists λ_C in $\hat{\Lambda}$ such that $\texttt{aNMAE}(C, \lambda_C) \leq e$. Figure 3.6 shows the coverage of our datasets as a function of the aNMAE. The results show that most clinical records are covered by our population within *small* aNMAE values. Namely, the totality of the Pfizer, Hannover, and Lausanne medical cases are covered within aNMAE 15%, 20%, and 35%, respectively. As for the latter dataset, 90% of the cases are actually covered within an aNMAE of just 20%.

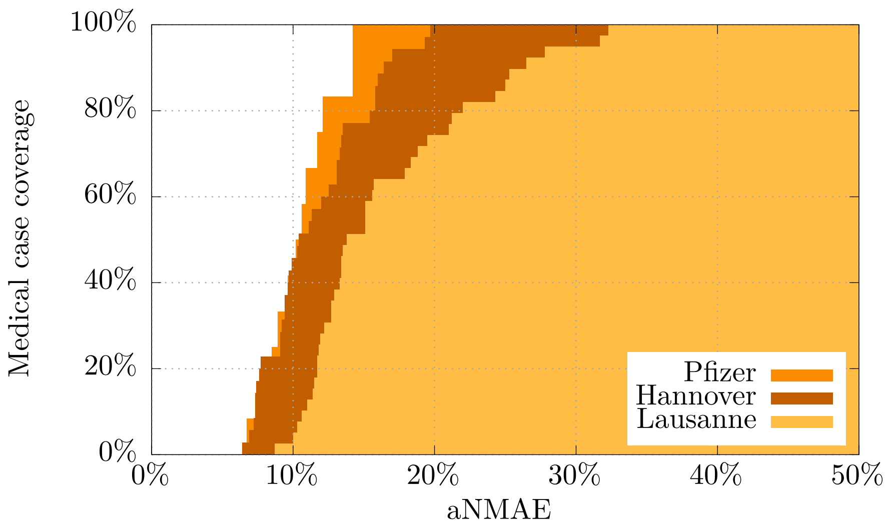


Figure 3.6. Quantitative validation of our GynCycle population against clinical data.

3.4 Related Work

In the literature different approaches to generate VPs have been devised. We distinguish those based on *optimisation* techniques and those based on *statistical* techniques. Both global and local *optimisation*-based approaches, relying on Artificial Intelligence (AI) as well as on numerical methods, have been widely studied. Global optimisation–based approaches are computationally heavy, however convergence is guaranteed to a global optimum (see, *e.g.*, [222, 167]). On the other hand, local optimisation approaches are in general computationally lighter than global ones, but convergence is only guaranteed towards a local optimum (*e.g.*, [114, 117, 157]). The reader is referred to, *e.g.*, [173] for an in-depth comparison among the two approaches. Typically, both global as well as local optimisation approaches rely on model identifiability, *i.e.*, different values for the model parameters lead to different model behaviours (see, *e.g.*, [134]). In such a case, different model identification techniques can be used. Examples are in [66, 175, 81, 206, 2, 201]. When clinical data are scarce, *e.g.*, when clinical assays are costly, risky and invasive, identification approaches can be applied by averaging data coming from different patients. However, the resulting parameter value yields an *inter-patient* model behaviour (see, *e.g.*, [184]).

Since the goal of the above mentioned approaches is to generate a model parameter value that fits available experimental data, a huge amount of data per patient is needed in order to generate a virtual population that is representative of all human phenotypes. As a result, generating a complete set of VPs using model identification techniques would require considering a large amount of patients (ideally at least one for each relevant phenotype) along with a huge amount of clinical data for each of such patients. Unfortunately, this is exactly one of the obstacles ISCT aims at overcoming. Thus, while model identification techniques are essential to validate models against clinical data and to develop *patient-specific* models, they can be hardly used to generate a complete set of VPs. Moreover, VP models are often globally or partially unidentifiable, *i.e.*, wide ranges of parameter values lead to very similar model behaviours (see, *e.g.*, [42]). However, being able to generate VPs (*i.e.*, parameter values) which yield clearly distinguishable model behaviours is of crucial importance for ISCT. *Statistical* approaches are also widely used. In such approaches a parameter probability distribution, rather than a single value, is inferred (see, *e.g.*, [95, 118, 194]). They are typically used for physiology-based PK/PD models (see, *e.g.*, [189]), *i.e.*, quantitative VP models where parameter values are measurable physiological quantities (such as blood flow, organ volumes, etc). Unfortunately, most VP models have hard-to-measure parameters such as stoichiometric constants and reaction rates. In fact, probability distribution functions for them are typically unknown. Limited knowledge about VP model parameters calls for methods that handle models whose parameters are partially unknown. Since model-checking techniques enable exploration of all possible behaviours of a model, it is not surprising that such approaches have been investigated in order to generate VPs for both qualitative as well as quantitative VP models. In particular, for qualitative models (*e.g.*, Boolean models) the problem of finding model parameter values yielding a biological meaningful behaviour is reduced to the problem of finding stable states of the model, *i.e.*, attractors. In such a setting, symbolic model checking is widely used (see, *e.g.*, [158, 88, 3, 103]). There, *physiological meaningfulness* can be defined using a *temporal logic*, such as CTL or LTL. Some examples are in [40, 22, 20, 85, 14] or in [91], where a probabilistic model checking approach is used. Also, in the case of quantitative models, Satisfiability Modulo Theories (SMT)-based approaches are investigated for the computation of parameter values of Ordinary Differential Equations (ODEs) (see, *e.g.*, [135, 132] and citations therein). Unfortunately, even using model approximation techniques like those in [9, 152], the above mentioned model checking approaches cannot be used to generate VPs from complex (non linear) ODE-based models, *i.e.*, those sought to support ISCT. Hence, the complexity of the VPH models relevant for *in silico* clinical practice makes such models out of reach for approaches like those above mentioned and appoints numerical integration as the only viable means to compute (black-box) the model evolutions. Indeed, in such a setting only simulation-based approaches are viable.

3.5 Conclusions

In this chapter we presented methods and software to compute a *representative and stratified population of VPs* for a given quantitative model of the human physiology (plus drugs PK/PD). Our approach is especially designed for complex (*e.g.*, non-linear ODE–based) parametric VPH models that cannot be analysed symbolically or integrated in closed form, but must be numerically simulated. To this end, our algorithm runs a global search on the space of model parameter values, guided by statistical model checking and hypothesis testing to sample those parameter values to be simulated, exploiting suitable biological and medical knowledge elicited from experts to recognise physiologically meaningful behaviours and different phenotypes, and structural knowledge of the model to intelligently drive the search via an informed sampling policy. Our algorithm can be stopped at *any-time*, since it continuously provides an upper bound (correct with a user-defined confidence level) to the probability that further computation will discover new phenotypes.

Results of this chapter have been presented in [36, 8, 196]. A previous work has been also presented in [205]. Software developed in this chapter is available at the following url: https://bitbucket.org/mclab/vipgenerator.

Chapter 4

Computation of optimal personalised treatments

4.1 Introduction

In this chapter we present methods and software that, by means of extensive computer simulation–based experimental campaigns (In Silico Clinical Trials, ISCT) guided by intelligent search, optimise a pharmacological treatment for an individual patient. We show the effectiveness of our approach on a case study involving a real pharmacological treatment, namely the *downregulation* phase of a complex clinical protocol for assisted reproduction in humans.

To do this, in order to conduct an In Silico Clinical Trial (ISCT), we exploit the state-of-the-art Virtual Physiological Human (VPH) model of the Hypothalamic–Pituitary–Gonadal (HPG) axis presented in Section 2.2.1, *i.e.*, the GynCycle model, (whose accuracy has been experimentally assessed in [184]). Namely, starting from clinical data collected from a real patient, we first compute her *digital twin*, which defines a *digital* representation of *that* patient physiology. Then, we compute *in silico*, by means of intelligent search on such a digital twin, the lightest (in terms of overall amount of administered drug) treatment still effective for that patient.

Our search algorithm extends standard artificial intelligence techniques (such as, *e.g.*, classical planning [218], CSP [185] and SAT [23]) in order to *intelligently* explore the space of possible time-series of drug administrations (treatments) by driving a simulator of the VPH model at hand. To improve the performance of our search algorithm, we also define suitable ordering heuristics (which keep our approach complete) and we evaluate their marginal performance gain. Moreover, our implementation can be easily adapted to several VPH models and to a wide class of clinical treatments as it drives a Modelica black-box simulator.

In order to assess the technical viability of our approach, we exploit the Gyn-

Cycle VPH model to conduct a multi-arm ISCT involving 21 patients (for which we have retrospective clinical measurements). For each such patient, who defines a distinct arm of our ISCT, we compute the optimal (lightest) still-effective downregulation treatment. Given that the required computation is extremely intensive, we conduct our ISCT on a large High Performance Computing (HPC) infrastructure.

The possibility to *optimise in silico* a complex treatment for a given human patient *before* its actual administration shows the potential of artificial intelligence search methods for model-based (*in silico*) medicine.

4.2 Methods

Below we define our formal framework and present our methodology to compute optimal personalised treatments.

4.2.1 Formal framework

In this section we give the necessary background knowledge for this chapter.

Throughout this chapter, VPH model are defined as in Definition 2.1. The input space of the VPH model is defined by the *administrations of n_u different pharmaceutical drugs* (clinical actions). In particular, for each $i \in [n_u]$, the input function $\mathbf{u}_i$ associates to each time instant $t \in \mathcal{T}$ the administered dose of the i-th drug, to be chosen within a set A_i of possible doses (which always includes dose zero).

The GynCycle VPH model (see, Section 2.2.1) and the downregulation treatment (see, Section 2.2.2) are used as a case study.

A population of Virtual Patients (VPs) is also used as a *starting point* for our ISCT. We denote such a population as $\mathcal{P}$ through this chapter.

4.2.1.1 Digital twin of a human patient

To compute a personalised treatment for a human patient, we first need to compute a *digital representation* for her. This will be done by using clinical data (in the form of a clinical record $\mathcal{C}$, Definition 2.2) available from that patient in order to select, from our representative population of VPs, the subset of VPs that are *compatible* with (*i.e.*, fit) such data.

To do so, we proceed as follows. Let $\mathcal{C}$ be a clinical record and λ be a VP of our population. Also, let $\eta(\mathcal{C}, \lambda) \in \mathbb{R}_{0+}$ be a function (*i.e.*, a Key Performance Indicator, KPI) computing the degree of similarity between the clinical measurements of $\mathcal{C}$ and the time evolutions of the VP λ with respect to all biological quantities measured in $\mathcal{C}$. The more $\eta(\mathcal{C}, \lambda)$ is near to zero the more the behaviour of VP λ is similar to the clinical measurements of $\mathcal{C}$. By using such a metric, Definition 4.1 defines the concept of *patient digital twin* as the set of *all* VPs λ in our population $\mathcal{P}$ having a value for $\eta(\mathcal{C}, \lambda)$ up to a given threshold.

Definition 4.1. *Given a clinical record $\mathcal{C}$, a population of VPs $\mathcal{P}$, a threshold $\delta \in \mathbb{R}_{0+}$ for a KPI η, the* digital twin *P of $\mathcal{C}$ is the following set:*

$$P(\mathcal{C}) = \{\lambda \in \mathcal{P} \mid \eta(\mathcal{C}, \lambda) \leq \delta\}\,.$$

Intuitively, the digital twin of a human patient is a *digital* representation of the patient physiology in the form of *all* VPs entailing model behaviours that fit the patient clinical measurements (up to a KPI η and a threshold δ).

4.2.2 Computing optimal personalised treatments

In this section, we define the conditions of a successful treatment through invariants and goals (Section 4.2.2.1), describe how to perform digital twin simulations (Section 4.2.2.2), and describe the algorithm used at the core of our optimal personalised treatment computation (Section 4.2.2.3).

4.2.2.1 Modelling treatment invariants and goals

We define conditions that must be *always* and *eventually* satisfied by a successful treatment by means of *invariants* and *goals*, respectively. Invariant and goal treatment conditions are modelled as continuous-time *monitors* embedded within the VPH model, along the lines of [140, 145]. Monitors observe the state of the system and check whether the properties of interest are satisfied.

In particular, given a model trajectory $\mathbf{y}_i(\mathbf{u}, \lambda)$ under a given input $\mathbf{u}$, our monitor output is Undet as long as $\mathbf{y}_i(\mathbf{u}, \lambda)$ satisfies the invariants (in other words, the monitor decision is "undetermined" as long as there is *hope* to extend the current treatment into a successful treatment) and goes to and *stays* at value Fail as soon as invariants are *violated*. When a Undet model trajectory satisfies the goal conditions, the monitor output turns to Success, informing the caller that the input $\mathbf{u}$ defines a *successful* treatment (*i.e.*, an *effective* treatment which *always* satisfies invariants).

The use of continuous-time monitors embedded in the VPH model gives us a flexible way to model both bounded safety and bounded liveness properties, and is a standard approach in the simulation-based verification of cyber-physical systems (see, *e.g.*, [142, 144]).

In our case study, the properties of interest are the conditions of successful downregulation treatments (Section 2.2.2). In particular, our *invariant* requires that the day of the first drug administration is between day 21 and day 25 of the menstrual cycle. Moreover, the value of all the biological quantities under

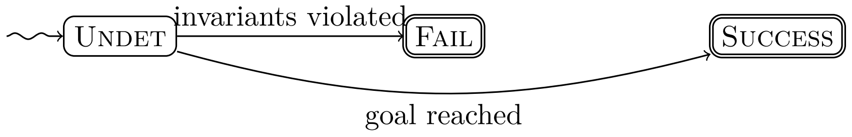


Figure 4.1. Treatment monitor.

observation must go and stay below their thresholds from the 9-th day after the first drug administration. Our *goal condition* instead requires that values for those quantities stay below their thresholds for 21 consecutive days.

As a consequence, our monitor output is Undet in the initial state, and eventually turns either to Fail or to Success (see Figure 4.1). The model output turns to value Fail as soon as, from the 9-th day after the first drug administration, the value of *any* of the physiological quantities under observation exceeds its threshold. Conversely, model output turns from value Undet to value Success as soon as all the thresholds are satisfied for 21 consecutive days.

4.2.2.2 Digital Twin simulation

As anticipated in Section 2.1.1, the complexity of the (highly non-linear) differential equations typically occurring in actual VPH models hinders the possibility for their symbolic analysis. Numerical *simulation* is often the only means to compute the time evolution of the model quantities of interests (mainly blood hormone concentrations in our case study GynCycle model) upon a sequence of clinical actions (administration of one or more drugs with their associated doses). Also, as VPH model equations are often stiff, simulation can be an *expensive* process from a computational point of view (see, *e.g.*, [7, 113]).

Our algorithm for optimal personalised treatment computation regards the input VPH model as an *executable* black-box model. In order to drive the VPH model simulator, our algorithm assumes that it accepts the following basic commands (which are available or can be readily implemented within most modern simulators, see, *e.g.*, [143, 146] and Chapter 5): `store`, `load`, `free`, `run`, `get`. Command `store`(l) stores in memory the current state of the simulator and labels with l such a state. Command `load`(l) loads into the simulator the stored state labelled with l. Command `free`(l) removes from the memory the state labelled with l. Command `run`(a, t) (with $a \in A_1 \times \cdots \times A_{n_u}$ and $t \in \mathcal{T}$) executes clinical action a (thus administering given –possibly zero– doses for all available drugs) and then advances the simulation of time t. Command `get`(x) (with x denoting one of the model variables) returns the value of model variable x in the current state.

Since the input of our algorithm is a digital twin $P(\mathcal{C})$ computed from a patient clinical record $\mathcal{C}$ and consisting of a set of $n_p = |P(\mathcal{C})| \in \mathbb{N}_+$ VPs (see Section 4.2.1.1), during the search we drive the simulation of n_p *independent copies* of our VPH model (each one connected to a copy of the monitor checking for treatment invariants and goals, as explained in Section 4.2.2.1). The different model copies are instantiated with different values for the model parameter vector defined in $P(\mathcal{C})$, hence represent the different VPs in the patient digital twin.

By means of commands `run`(a, t) ($a \in A_1 \times \cdots \times A_{n_u}$), we simultaneously apply the same clinical action to *all* VPs in $P(\mathcal{C})$ and synchronously advance all models by time t. Furthermore, by means of `store` and `load` commands, we can efficiently expand a tree of sequences of clinical actions, as typically done by backtracking-based state exploration algorithms. In particular, upon a backtrack, we only need to load back a previously stored state of our models, thus avoiding to simulate the entire input sequence starting from the initial state.

4.2.2.3 Intelligent backtracking–driven simulation

In this section we describe our intelligent search that drives a digital twin simulator, by defining our search tree (Section 4.2.2.3.1) and criteria for the evaluation and expansion of search tree nodes (Section 4.2.2.3.2). We also introduce two ordering heuristics aiming at efficiently pruning the search tree (Section 4.2.2.3.4) and speeding up the digital twin simulation time (Section 4.2.2.3.5), while retaining completeness of the overall algorithm.

4.2.2.3.1 Search tree definition. Given a patient clinical record $\mathcal{C}$, its digital twin $P(\mathcal{C})$, and possible alternative doses for each treatment drug at hand (*i.e.*, sets $A_1, \ldots, A_{n_u}$, each one including dose zero), our backtracking-based algorithm performs a search in the space of possible treatments (*i.e.*, time sequences of clinical actions in $A_1 \times \cdots \times A_{n_u}$). Under the following assumptions: (i) the sets of allowed doses $A_1, \ldots, A_{n_u}$ for the n_u drugs are *finite*; (ii) drug administrations occur at time instants multiple of a given *time-quantum* $\tau \in \mathbb{R}_+$; and (iii) sought treatments have a bounded duration (τh, with $h \in \mathbb{N}_+$ being the treatment *horizon*), we can limit our search to a *finite tree* of maximum depth h, where each tree node at depth $0 \leq l \leq h$ defines a *discrete sequence* of l clinical actions (elements of $A_1, \ldots, A_{n_u}$), each one occurring in the first l time points multiple of τ.

As described in Section 4.2.2.2, our search algorithm drives a simulator containing multiple copies of the VPH model (one per VP in the digital twin given as input), each one connected to a copy of the monitor which checks for treatment invariants and goals.

Accordingly, each node of our search tree keeps also track of the digital twin simulator state reached by injecting the clinical action sequence of that node. This is done via a `store` command (see Section 4.2.2.2). A transition between a parent node and a child node consists of extending the clinical action sequence of the parent node with a new action $a \in A_1 \times \cdots \times A_{n_u}$. In doing so, we also advance the digital twin simulator state of the parent node via a `run` command (see Section 4.2.2.2). In Section 4.2.2.3.2, we describe more in details how a search tree node is evaluated and expanded.

Before the search starts, for each VP λ in the input digital twin, we prepare its associated copy of the VPH+monitor model within the digital twin simulator, by setting the parameter vector to λ. We also store the initial simulator state via a `store` command (see Section 4.2.2.2).

Clearly, the assumptions above together with the entailed search tree structure hold for a very wide class of treatments. For example, in our downregulation treatment case study, there is only one possible drug, namely Triptorelin, hence $n_u = 1$. Also, as such treatments allow at most one Triptorelin administration per day, we can safely set $\tau = 1$ day. Finally, successful treatments cannot last more than $h = 25 + 9 + 21 = 55$ days, starting from day 1 of the patient menstrual cycle (*i.e.*, the latest cycle day, 25, when the treatment might start, plus the latest day, 9, within which the safety conditions must be met, plus the number of consecutive days, 21, in which the safety conditions must be always satisfied in order to declare success).

Moreover, in our case study, the initial simulator state represents day 1 of the menstrual cycle of all VPs in the input digital twin.

4.2.2.3.2 Search tree node evaluation and expansion.

At each node of the search tree, our algorithm needs to extend the current sequence of clinical actions (a treatment *prefix*, which is the empty sequence in the initial state). To do so, the algorithm:

1. Stores the current states of the VP models by means of a `store` command.

2. For each clinical action $a \in A_1 \times \ldots \times A_{n_u}$ (according to the heuristic ordering discussed in Section 4.2.2.3.4), injects a into the simulator and advances the simulation (simultaneously for all VPs defined within the simulator) by time τ (this is done by issuing a command `run`(a, τ)).

3. If the monitor output for one VP returns FAIL, the algorithm infers that there is no hope to extend the partial treatment to a complete treatment successful for all the VPs in the digital twin and issues a backtrack, by loading back (via a `load` command) the simulator state of all VPs associated to the parent node in the search tree.

4. Otherwise, if the monitor output for all VPs is SUCCESS, the algorithm infers that the current treatment is successful for all VPs.

5. Otherwise, the algorithm continues by expanding the new search tree node and extending the current treatment prefix.

Whenever no further actions can be executed in the current node of the search tree, the current simulator state is freed from memory by a `free` command before triggering a backtrack.

In order to reduce the required simulation time when expanding each node, the multiple model copies are not actually packaged into a single (much heavier to be simulated) model, but are kept separate and simulated *sequentially.* This allows our algorithm to perform two performance optimisations: a sort of *early pruning* of the current treatment prefix and a *dynamic revision of the order* with which the VPs in the input digital twin will be actually simulated in the sequel (Section 4.2.2.3.5).

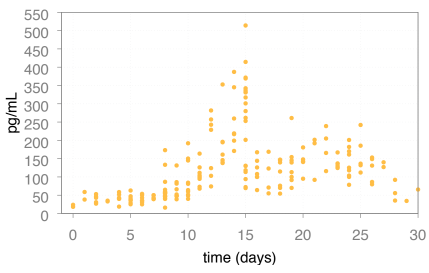
4.2.2.3.3 Optimality constraint. Since we seek the *optimal* personalised treatment for the input digital twin, our algorithm does not stop at the first found successful treatment. Indeed, along the lines of [139], it keeps track of the *lightest* successful treatment found so far, *i.e.*, the one envisioning the administration of the *minimum* overall drug amount $D_{\min} \in \mathbb{R}_{0+}$. Initially, $D_{\min}$ is set to $+\infty$.

At each node of the search tree, any action extending the current partial treatment with an administration of a drug dose which would make the overall administered drug amount reaching or exceeding $D_{\min}$ is regarded as not applicable.

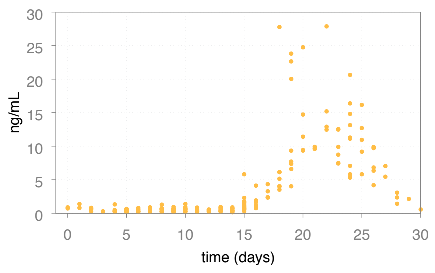
The optimality constraint (whose threshold is updated each time a new optimal treatment is found), the presence of a bounded horizon h and the fact that our algorithm does not stop at the first successful treatment clearly guarantee that a global optimum is always returned (if any successful treatment exists).

4.2.2.3.4 Action ordering heuristic. In a black-box setting as ours, no inference can be made on the effects of each candidate action without performing a simulator run to actually advance the model and then querying the model monitor output. Hence, approaches to compute, through inference, a dynamic preference order among the candidate actions to be tried during search (as those exploited in, *e.g.*, classical planning, planning for white-box hybrid systems –see Section 4.4– or (Q)CSP, SAT or local search solvers, see *e.g.*, [185, 23, 82, 86, 138, 32, 33]) cannot be applied.

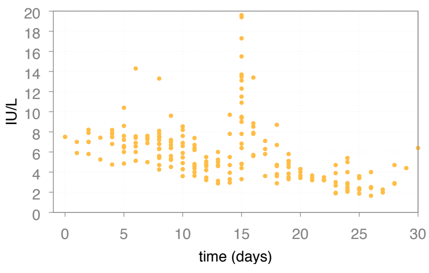
The only information available to the algorithm without running the simulator is value $D_{\min}$ and the overall cost of the *current* partial treatment (overall amount of drug administered). Hence, given that our algorithm searches for a successful treatment of minimum cost and that any action has a non-negative contribution to the overall treatment cost, not surprisingly our algorithm tries the clinical actions on each node in the search tree in ascending order of their associated dose (*i.e.*, cost), hence performing a *greedy*, optimistic choice.



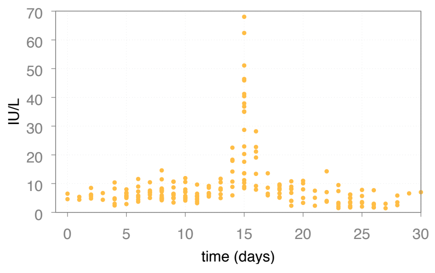
(a) E2



(b) P4



(c) FSH



(d) LH

Figure 4.2. Measurements (dots) of, respectively, E2, P4, FSH and LH of the 21 clinical records within the dataset from Pfizer Inc. and Hannover Medical School (Germany).

4.2.2.3.5 Dynamic VP simulation ordering. When, during the evaluation of search tree node, the monitor output after the simulation of a VP in the digital twin returns FAIL, our algorithm, by performing a sort of *early pruning*, does not simulate nor evaluate the remaining VPs, but it backtracks and tries the next available action from the parent node, if any. Upon backtracking, the algorithm changes the verification ordering of the VPs in the digital twin in such a way that those VPs not satisfying treatment invariants or not yet simulated in the previous search step are simulated *first*. The rationale is that those VPs that already satisfied treatment invariants in previous step, will more likely satisfy treatment invariants in the current step, where a slightly different drug dose is chosen.

4.3 Experimental Results

In this section we outline how we set up and conducted our ISCT in order to assess the technical viability of the proposed approach.

4.3.1 Retrospective clinical records

Our ISCT has been conducted against 21 clinical records contained in the following two datasets of our retrospective clinical data: Pfizer Inc. and Hannover Medical School (Germany) (see Section 2.2.3). Such 21 clinical records count around 800 measurements for the hormones that are monitored during assisted reproduction treatments and that play the most important role during the human menstrual cycle: LH, FSH, E2, and P4. Figure 4.2 shows measurements of these hormones in our overall dataset.

Each of the 21 clinical records defines a *distinct arm* of our multi-arm ISCT.

4.3.2 Computing digital twins: exclusion criteria and VP selection

From the GynCycle model, a population of VPs has been generated using the approach presented in [205, 150] with an error threshold equal to 0.01% and a statistical confidence bound equal to 95%. Such a population, $\mathcal{P}$, counts about 10^7 VPs. Also, it is representative of (almost) all patient behaviours allowed by the model (in the sense of [205, 150]). Note that, although generating such a population required weeks of HPC computation (see [150]) this is an off-line and one-time task for a VPH model, as also shown in Section 3.3.1 for our novel VP-generation methodology.

In Section 3.3.3.2, we adopted the Average Normalised Mean Absolute Error (aNMAE) because our goal was to assess coverage of our VP population on retrospective clinical data. Indeed, aNMAE is less sensitive to outliers as it aggregates absolute errors before normalising them over the maximum value of s. This normalisation is needed to compare values for the MAE computed for the different species. Conversely, in the setting of this chapter the prediction power of VPs (within a digital twin) is very important as our final goal is to compute personalised treatments. Thus, we need a measure of prediction accuracy which is more sensitive to errors and outliers. To this end we use Mean Absolute Percentage Error (MAPE), which normalises each absolute error over the actual observation value, namely, $\texttt{val}(s, j)$.

In particular, in order to compute digital twins, we use as KPI $\eta(\mathcal{C}, \lambda)$ the average (over all biological quantities s) MAPE between the clinical measurements of $o(s)$ in $\mathcal{C}$ and time evolutions of s in VP λ. Formally:

$$\eta(\mathcal{C}, \lambda) \;=\; \texttt{avg}_s \texttt{avg}_j \frac{|\mathbf{y}(\texttt{time}(s, j);\; \mathbf{u}_C, \lambda) - \texttt{val}(s, j)|}{\zeta(\texttt{val}(s, j))}$$

where $\zeta(\texttt{val}(s, j))$ is $\texttt{val}(s, j)$ if $\texttt{val}(s, j) \neq 0$ and a small positive constant otherwise (to avoid division by zero).

In order to carry out our multi-arm ISCT, we computed a digital twin for each of the 21 clinical records in our retrospective clinical data. In particular, we set the threshold value δ (Definition 4.1) to 35% and, for each clinical record $\mathcal{C}$, we included in the digital twin of $\mathcal{C}$, $P(\mathcal{C})$, all $\lambda \in \mathcal{P}$ such that $\eta(\mathcal{C}, \lambda) \leq 35\%$. Hence, for each clinical record, we selected *all* VPs having an average MAPE at most 35%.

Since the reference downregulation treatment of Section 2.2.2 does not succeed on all possible phenotypes (this is not surprising since, as described in Section 2.2.2, fertility treatments are known to have a low rate of success), we removed from $P(\mathcal{C})$ those VPs for which the treatment using the maximum allowed dose (*i.e.*, 0.1 mg on each day for up to 30 days, see Section 2.2.2) fails, for any $\mathcal{C}$. Such an *exclusion criterion* directly stems from domain knowledge on downregulation treatments (namely: for these protocols, if a treatment envisioning one full drug dose per day fails for a patient, a lighter treatment will fail as well). Our 21 digital twins contain, on average, 843 424 VPs each.

Unfortunately, in the worst-case scenario, each step of our backtracking search requires to simulate all VPs within the input digital twin in order to check the monitor output (*i.e.*, treatment invariants and goals, see Section 4.2.2.3.2). Thus, since simulating one VP requires seconds of numerical integration, the total computation time of our algorithm is highly affected by the size of the input digital twin, and may easily become impractical.

To overcome such an obstacle, we note that within a digital twin of a given clinical record we can (and typically do) have VPs exhibiting very similar model behaviours both with and without drug administrations. Simulating such very similar VPs would then simply be a waste of computation as long as our ISCT is concerned. In order to reduce the size of a digital twin, we compute a subset $\hat{P}(\mathcal{C})$ of $P(\mathcal{C})$, for each given clinical record $\mathcal{C}$, such that $\hat{P}(\mathcal{C})$ does not contain such *redundant* VPs.

To this end, we define the *distance* among the time evolutions of any model observable $\mathbf{y}_i(\mathbf{u}, \lambda)$ of two VPs normalised against the time evolution of $\mathbf{y}_i(\mathbf{u}, \lambda)$ of a third (reference) VP (Definition 4.2) when subject to the same input.

Definition 4.2. *Let λ, λ' and $\lambda^\star$ be three VPs of a VPH model having n model observables.*

For every model observable $\mathbf{y}_i(\mathbf{u}, \lambda)$ ($i \in [n]$) the distance *between λ and λ' with respect to $\lambda^\star$ when subject to the same input function $\mathbf{u}$ on i is:*

$$d_{\mathbf{u},\lambda^\star,i}(\lambda, \lambda') = \frac{||\mathbf{y}_i(\mathbf{u}, \lambda) - \mathbf{y}_i(\mathbf{u}, \lambda')||}{||\mathbf{y}_i(\mathbf{u}, \lambda^\star)||}$$

where $|| \ ||$ is the L2-norm.

Intuitively, Definition 4.2 defines the *Euclidean distance* between the model observable time evolutions of two given VPs (λ and λ') *normalised* with respect to the L_2-norm of that of a *reference* VP, $\lambda^\star$.

Given this notion of distance, we build $\hat{P}(\mathcal{C})$ from the digital twin $P(\mathcal{C})$ of a clinical record $\mathcal{C}$ as follows:

- We select, from $P(\mathcal{C})$, the reference VP $\lambda^\star$ as one that *minimises* $\eta(\mathcal{C}, \lambda^\star)$. In other words, $\lambda^\star$ is one of the VPs of $\mathcal{P}$ that *best matches* the clinical measurements of $\mathcal{C}$.
- We *remove* from $P(\mathcal{C})$ those VPs whose time evolutions of *all* biological quantities have a distance below a certain threshold from other VPs in $\hat{P}(\mathcal{C})$ with respect to $\lambda^\star$ (Definition 4.3).

Such a distance threshold obviously depends on the VPH model and treatment subject of the ISCT.

Definition 4.3. *Let $\mathcal{S} = (\mathcal{T}, \Lambda, \mathcal{U}, \mathcal{Y}, \mathbf{y})$ (Definition 2.1) be a VPH model, whose output space $\mathcal{Y}$ is $\mathbb{R}^n_{0+}$ where $n \in \mathbb{N}_+$ is the number of (real-valued) observables and a set of input functions $\mathbf{U} \subset \mathcal{U}^{\mathcal{T}}$.*

Let $\mathcal{C}$ be a patient clinical record, $P(\mathcal{C})$ be its digital twin, $\lambda^\star$ be any VP such that $\eta(\mathcal{C}, \lambda^\star)$ is minimal, and d be the distance function of Definition 4.2.

Then, given a threshold θ, we call compact digital twin $\hat{P}(\mathcal{C})$ any *subset of $P(\mathcal{C})$ such that the following condition holds:*

$$\forall \lambda \in P(\mathcal{C}) \,.\, \lambda \notin \hat{P}(\mathcal{C}) \rightarrow \exists \lambda' \in \hat{P}(\mathcal{C}) \,.\, \forall i \in [n] \; \forall \mathbf{u} \in \mathbf{U} \; d_{\mathbf{u},\lambda^\star,i}(\lambda, \lambda') \leq \theta/2.$$

As a result of the application of the above approach and of our exclusion criteria, we obtained compact digital twins with an average size of 11. This has been achieved by tuning the distance threshold θ in such a way that: (i) VPs having redundant and undistinguishable behaviours have been filtered out; (ii) the computed compact digital twins were representative enough for all VPs in corresponding patient digital twins; and (iii) their sizes were small enough to be computationally affordable. Also, we consider a portfolio of input functions, $\mathbf{U} \subset \mathcal{U}^{\mathcal{T}}$, containing a no-drug input and the reference downregulation treatment (Section 2.2.2). Figure 4.3 shows the distribution of digital twin sizes.

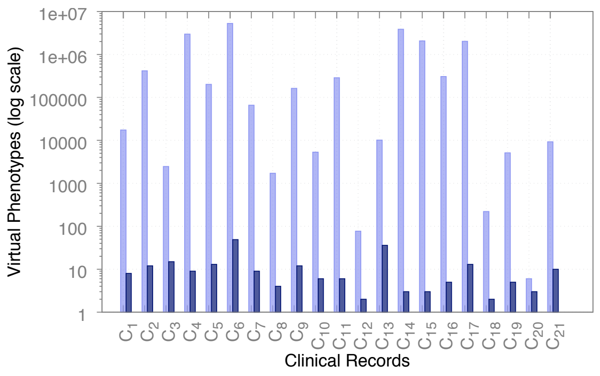


Figure 4.3. Bar chart showing digital twin sizes (in terms of number of VPs) before (light) and after (dark) the removal of redundant VPs and the application of our exclusion criteria for each clinical record.

4.3.3 Multi-arm In Silico Clinical Trial run

We ran our 21-arm ISCT using a large HPC infrastructure (the Marconi cluster) kindly provided by the Cineca consortium.

For each patient clinical record $\mathcal{C}$ in our retrospective clinical data (see Section 2.1.2), we, first, computed a compact digital twin $\hat{P}(\mathcal{C})$ (see Section 4.3.2), and then we ran our algorithm of Section 4.2.2 on an independent node of the cluster searching for an optimal (*i.e.*, lightest) and robust (with respect to *all* VPs of $\hat{P}(\mathcal{C})$) downregulation treatment for *that* specific digital twin. Thus, the whole 21-arm ISCT has been conducted in an *embarrassing parallel fashion.*

Each VP in a compact digital twin has been encoded in a Modelica (www.modelica.org) model (encompassing the GynCycle VPH model taking clinical actions as input, a parameter vector assignment, and a monitor to check for treatment invariants and goals) and has been compiled as an executable object (*i.e.*, a Functional Mock-up Unit, FMU) using the extended version of JModelica v2.1 presented in the forthcoming Chapter 5. Our search algorithm (which drives the FMU of each VP) has been implemented in Python.

In the following sections, we first evaluate the marginal impact of our heuristic ordering of actions (Section 4.3.3.1) and of our dynamic simulation ordering of VPs within a digital twin (Section 4.3.3.2). Then, we present computational results (Section 4.3.3.3) and outcomes (Section 4.3.3.4) of our multi-arm ISCT.

4.3.3.1 Evaluation of the action ordering heuristic

The goal of this section is to assess the marginal impact of the heuristic presented in Section 4.2.2.3.4, which sorts actions by their ascending associated cost (administered drug dose), in the overall efficiency of the optimal personalised treatment search algorithm.

To this end, we compare it against 10 runs of an action selection strategy that evaluates the possible actions in *random* order when expanding each node of the search tree, and compare the performance of our heuristics against such reference strategy.

In particular, we defined a limit of 10^4 on the number of search tree nodes to expand, and analysed how fast the cost (overall amount of drug dose) associated to the personalised treatments found *decreases* during the search space exploration, on all digital twins involved in our multi-arm ISCT.

In Figure 4.4, we present 3 representative executions where our heuristic finds an optimal solution at different points in time (with respect to the number of expanded search tree nodes). For convenience of presentation, the initial value of D_{min} in Figure 4.4 is set to 3.0 (drug quantity employed by the reference treatment, *i.e.*, 0.1 mg on each day for up to 30 days) instead of $+\infty$ (as stated in Section 4.2.2.3.3). This is in agreement with our exclusion criteria of Section 4.3.2 as the reference treatment succeed on all VPs within our digital twins.

We note that, by choosing actions randomly, the algorithm might show its first solution improvements a bit earlier (Figures 4.4a and 4.4c). However, in all cases, our heuristics finds the optimal solution (*i.e.*, the lightest treatment) within a number of expanded nodes which is *orders of magnitude* lower than the number of expanded nodes required by the random action ordering strategy. As an example, Figure 4.4c shows that, after 10^4 expanded nodes, 8 out of 10 runs with the random action ordering still failed to find a solution better than the reference treatment (*i.e.*, 3.0 mg).

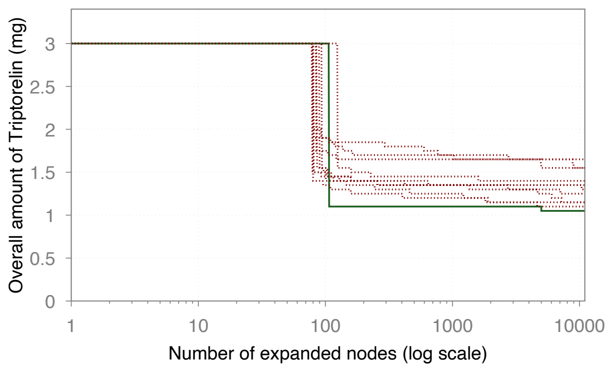
4.3.3.2 Evaluation of the dynamic ordering of VPs within a digital twin

In order to evaluate the marginal impact of our dynamic VP simulation order within the input digital twin (Section 4.2.2.3.5) at each search tree node, we compared the execution of our algorithm against a reference obtained by averaging 10 different runs where the initial order of the VPs is randomised and fixed at the beginning of the search.

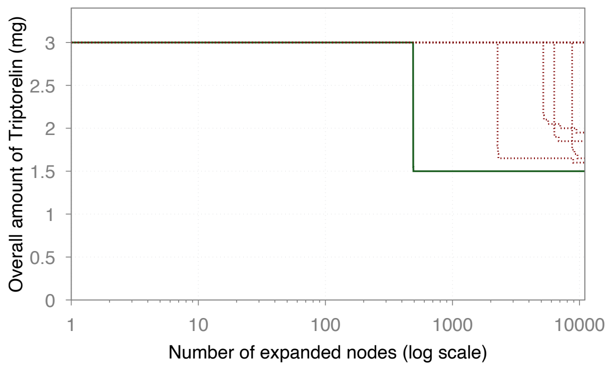
In particular, we ran our algorithm for each digital twin involved in our multi-arm ISCT, and compared the saving in terms of the number of simulations performed within each given number of search tree nodes.

Figure 4.5 shows our result on 3 digital twins having different sizes: (i) a small size — 3 VPs (Figure 4.5a); (ii) an average size — 13 VPs (Figure 4.5b); and (iii) a large size — 36 VPs (Figure 4.5c), which are representative of the spectrum of behaviours among our digital twins.

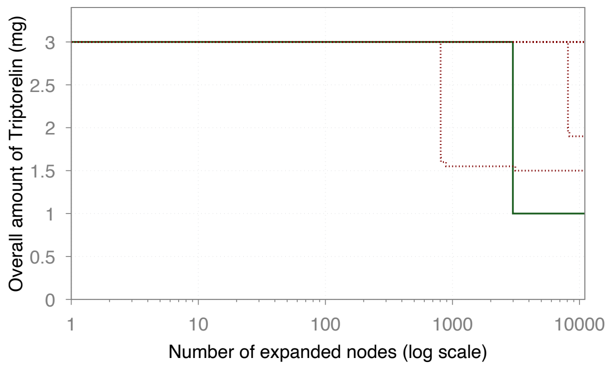
We note that the our dynamic VP simulation order is always beneficial. The highest saving in the number of simulations always occurs at the early stages of the search, where the most pruning activity is performed. In deeper stages of the search, the savings stabilise at values of the order of 10%–20%. Since, in our setting, the time needed to simulate a VP is essentially constant (as VPs differ only in model parameter values and not in the system of Ordinary Differential Equations, ODEs), such savings translate in comparable reductions of the overall computation time.



(a)

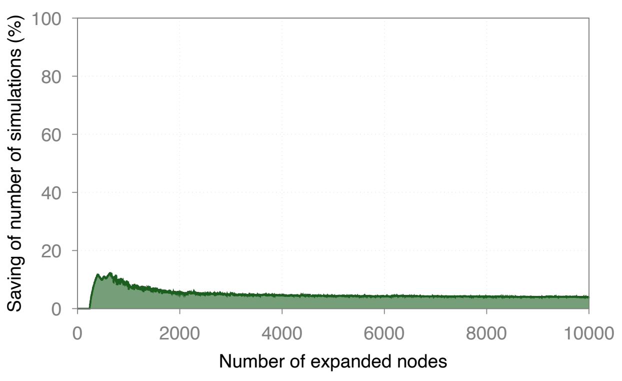


(b)

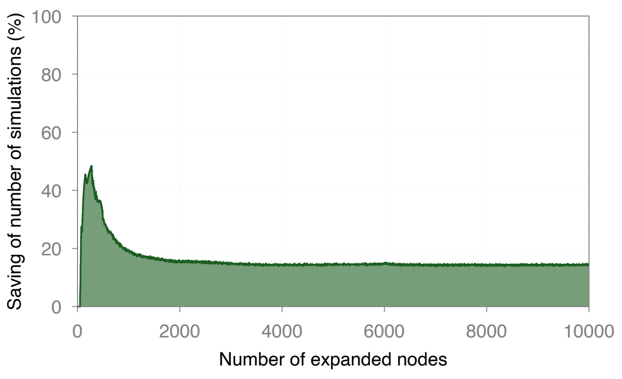


(c)

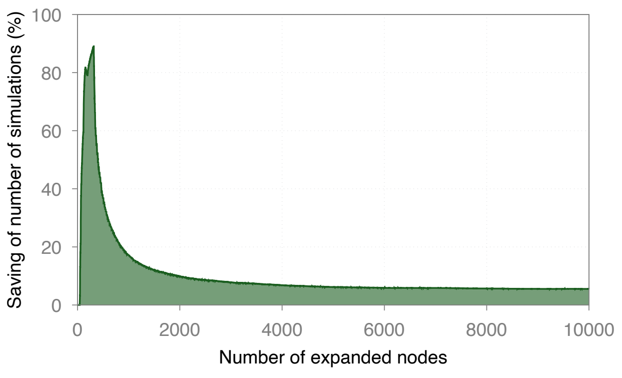
Figure 4.4. Comparison of solution paths traversed by our algorithm when using our ordering heuristic (green line) against the references (red dashed lines) among different digital twins of our multi-arm ISCT.



(a)



(b)



(c)

Figure 4.5. Marginal impact of our dynamic VP simulation ordering strategy among the digital twins of our multi-arm ISCT.

4.3.3.3 Computational results

We ran our multi-arm ISCT on the Marconi HPC infrastructure kindly provided by Cineca, Italy. Due to CPU-hours budget limit, we ran each arm of our ISCT in parallel for 15 days. Each arm searches the optimal treatment for one of the 21 patients in our dataset. Indeed, our algorithm can be regarded as an *any-time* algorithm in that, at any moment during search, it keeps track of the lightest successful treatment found so far.

Figure 4.6a gives, for each ISCT arm, the share of simulation time within the whole computation time. As we already argued in Section 4.3.2, the size of the input digital twin highly impacts the speed (in terms of expanded search tree nodes per seconds) of our algorithm. In fact, in each search node (in the worst case), our algorithm needs to numerically simulate all VPs within the input digital twin. As shown in Figure 4.6a, the time spent in carrying out simulations is, on average, the 81.2% of the total computation time. Note that, as already described in Section 4.3.3, each VP model (encoding the GynCycle VPH model, a parameter vector assignment and the treatment monitor) is compiled into an FMU. This compilation is needed only once as during the search each FMU is used through its interface (i.e., the Functional Mock-up Interface, FMI) (see Section 4.2.2.3.2 where high-level simulation Application Programming Interfaces (APIs) are described and Chapter 5 for a deeper description of the FMI standard). This strategy already saves computation time as it avoids compiling the VP Modelica models for each input sequence that has to be injected during the search.

This implies that any technique aimed at reducing the average number of simulations per search node (like the reduction of the size of the considered digital twin defined in Section 4.3.2, the action order heuristic evaluated in Section 4.3.3.1, and the dynamic VP simulation ordering evaluated in Section 4.3.3.2) is expected to have a *substantial impact* on the actual performance of the algorithm.

Figure 4.6b shows, for each of the 21 parallel search processes (one per clinical record), the number of search nodes expanded within our time limit of 15 days (orange bars, left Y axis), and compares it to the size of the digital twin considered for that clinical record (blue bars, right Y axis).

The figure show a *clear negative correlation* (correlation coefficient equal to -60%) between number of expanded nodes within our time limit and the digital twin size for all clinical records. For example, we note that when the digital twin size is small, as for, *e.g.*, $\mathcal{C}_4$, the number of expanded nodes is large, *i.e.*, near 10^6. Instead, when the digital twin size is large, as for, *e.g.*, $\mathcal{C}_{17}$, the number of expanded nodes is quite low, *i.e.*, less than 3×10^4.

4.3.3.4 ISCT outcomes

Figure 4.7 shows the distribution of the total amount of drug employed by the optimal personalised treatments computed by our algorithm. Note that, although we ran our multi-arm ISCT with a limited time budget and did not find provably *lightest* treatments, the treatments we actually computed use, on average, less than *half* (40.82%) of the drug quantity employed by the reference treatment of Section 2.2.2 (*i.e.*, 0.1 mg on each day for up to 30 days, totalling up to 3 mg and never less than 2.8 mg, since the required thresholds are never reached before day 7). As Figure 4.7 shows, the computed treatments save, on average, 59.18% of the drug administered in the reference treatment (standard deviation: 13%).

In Figure 4.8, we show our computed treatments for the 21 patient clinical records during our multi-arm ISCT. We note that, the majority of the computed drug administration sequences is close to the reference downregulation treatment in terms of administration frequency and treatment duration, *i.e.*, around 28 days. Besides this, there are also few computed treatments having a very light overall amount of drug dose. This of course depends on the ability of the digital twins to capture patient peculiarities and in general on the ability of the VPH model at hand to represent the human physiology of interest. However, it is important to read carefully such promising results as an *in vivo* evaluation is needed to assess the accuracy of the GynCycle model predictions of *non-standard drug dosing patterns* (*i.e.*, our personalised treatments).

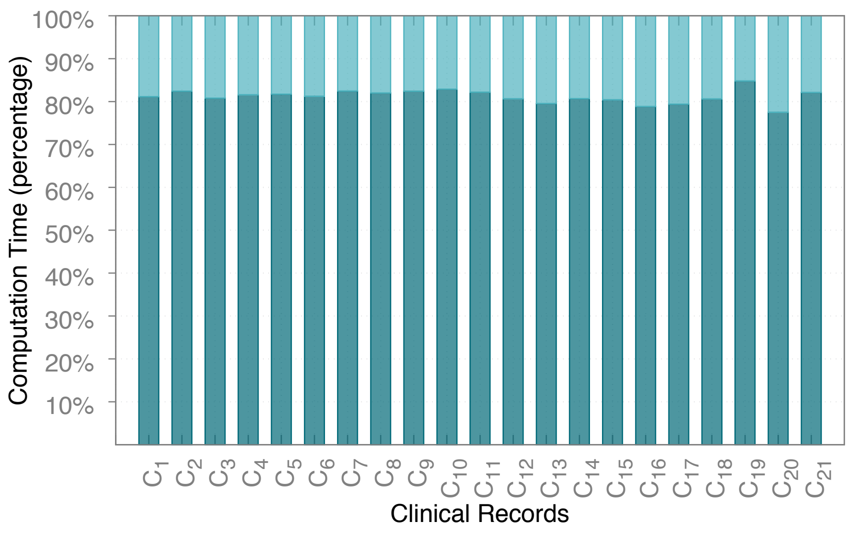
4.4 Related Work

Individualised treatments have the potential value to reduce costs and improve outcomes of standard clinical treatments. In recent years data-driven techniques have been investigated thanks to the availability of big data [178]. For example, the knowledge-base approach in [77] has been used to optimise treatment plans for lung cancer. Unfortunately, in presence of scarce clinical data for the patient at hand, the above approaches cannot be applied. For example, in our case study hormones blood concentrations are not measured every day, since those measurements are costly and invasive. Model-based approaches, exploiting PharmacoKinetics (PK), as *e.g.*, [220], are used instead to build populations of virtual phenotypes. Such populations are used to optimise and individualise drug doses [105, 212]. PK-based models, however, do not define how administered drugs can affect a VP (namely, PharmacoDynamics, PD), *i.e.*, possible side-effects due to drug administrations are not taken into account. In our model-based setting, we have to face with complex VPH models, *e.g.*, HumMod [94], Physiomodel [154], and GynCycle [184] defined through highly non-linear differential equations modelling the underlying biological mechanisms (*e.g.*, inhibitory and stimulatory effects). As outlined in Section 2.1.1, such VPH models are hybrid systems that can be defined by systems of ODEs (see, *e.g.*, [120, 21, 216]) whose inputs are discrete event sequences (see, *e.g.*, [140, 144]). To find an optimal treatment means to find an optimal plan in hybrid domains, where the behaviour of the given system is described by both discrete and continuous quantities. In the literature, there are many techniques to model planning problems in hybrid domains, *e.g.*, PDDL+ [78, 211]. Most of PDDL+ planners can deal only with linear dynamics (*e.g.*, [48]). [35] proposed a Satisfiability Modulo Theories (SMT)-based approach for solving PDDL+ problems with non-linear dynamics. Model checking techniques are also used to find plans. Examples in this direction are [31, 30], which exploits symbolic model checking, UPMurphi [59, 57], which given as input a PDDL+ problem specification computes a universal plan, CGMurphi [56], an explicit model checker used to compute optimal controllers, and [9, 152, 156, 187], which define methodologies to compute controllers for non-linear hybrid systems. However, as outlined in Section 4.2.2.2, the typical complexity of the differential equations of VPH models relevant for clinical practice makes such models out of reach for symbolic approaches like those mentioned above, and appoints numerical integration as the only viable means to compute (black-box) the model evolutions under a given input function. In particular, even considering that clinical actions have constant and equal duration (as, *e.g.*, in [128]), no reasoning or inference can be made on action effects in a black-box setting as ours, because the only way to interact with the models is through numerical simulation. The automated synthesis of rational decisions and plans in black-box environments is common in several other application domains of high industrial relevance, like smart grids (see, *e.g.*, [147, 149, 90]), games (see, *e.g.*, [130]) and real-time manoeuvring of Unmanned Aerial Vehicles (see, *e.g.*, [180]). The works closest to ours are those in [130, 79] and citations thereof, where a simulator is used to discover the effect of actions. In such works, the simulator is defined as a factored state model where actions are (black-box) procedures and states are represented in terms of variables. Then an algorithm, namely, Iterated Width (`IW`), is typically employed. It consists of iteratively calling a breadth-first procedure, namely, `IW`(i), with $i = 1, 2, 3, \ldots$, until the problem is solved or i becomes greater than the number of state variables (see, *e.g.*, [129]). During each call to `IW`(i), states are pruned accordingly to how novel they are. A state is novel if and only if values for at most i state variables have not seen before. Authors point out that `IW` is efficient for the majority of classical planning problems where it is enough that $i = 1, 2$, *i.e.*, actions change at most 2 state variables. However, in our setting, our simulator state consists of the union of all state vectors, *i.e.*, $\mathbf{x}(t)$, of each VP+monitor within the input digital twin. Moreover, the most of variables within such state vectors consist of real-valued biological quantities. Also, we remind that the dynamics of such model are defined by means of ODEs. Hence, when we apply a clinical action (and simulate our digital twin) the vast majority of state variables change their value. In this setting, applying `IW` means nearly always to call only `IW`(n), with n the overall number of variable of our simulator state. This behaviour makes the pruning on which `IW` relies on quite ineffective as `IW` degenerates to prune truly duplicate states, which are also extremely rare as our variables are real valued. Note that, in order to increase pruning of duplicate states, one could think of simplifying the dynamics of the VPH model at hand (*e.g.*, by discretising state variable domains). However, this is not a straightforward approach as it poses several questions about accuracy, credibility, and trust of the entailed model predictions, which have to be again experimentally assessed.

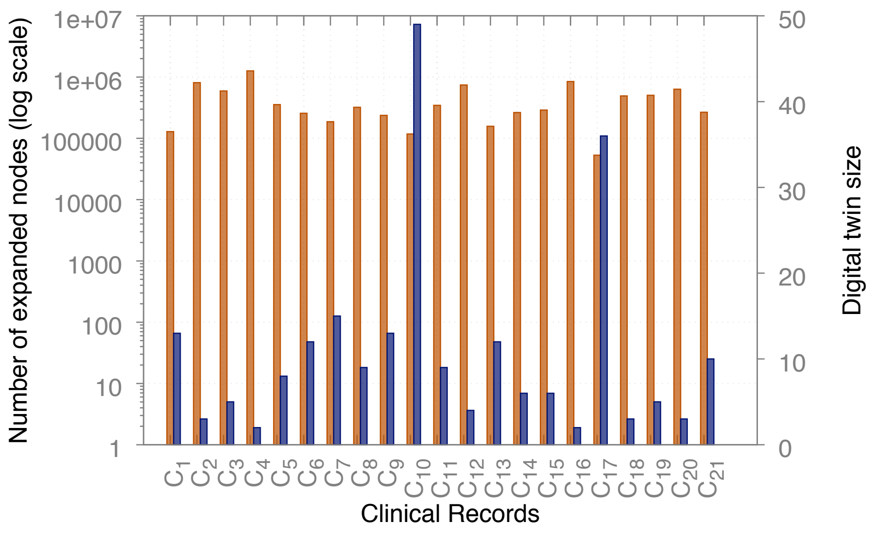
4.5 Conclusions

In this chapter, we presented methods and software based on intelligent search aimed at synthesising optimal personalised treatments by means of ISCT, exploiting quantitative models of the physiology and drugs Pharmacokinetics/Pharmacodynamics (PK/PD) of interest, and clinical measurements on human patients from which we define their digital twins. We applied our approach on a case study involving a complex state-of-the-art model of the human female HPG axis, in order to compute, for any given patient, a *personalised treatment* for the downregulation phase of an assisted reproduction protocol, which is effective on the patient at hand, but minimises the overall amount of drug used (hence, indirectly, the associated cost as well as likelihood and severity of adverse effects). The possibility to *optimise in silico*, in a few weeks of computation on a HPC infrastructure, a complex treatment for a *given* human patient *before* its actual administration shows the potential of artificial intelligence for model-based (*in silico*) personalised medicine. This however calls for *trusted* (*i.e.*, qualified) VPH models, which is currently one of the major obstacles for the uptake of ISCT in clinical practice. Indeed, the results of our ISCT (conducted using a state-of-the-art *validated* VPH model as GynCycle) are extremely promising, but must be taken with care. In particular, an *in vivo* evaluation of the actual effectiveness of the personalised treatments generated by our algorithm is needed in order to assess the accuracy of the GynCycle model in predicting the patient reactions to *non-standard drug dosing patterns*, as those computed by our algorithm.

Results in this chapter have been presented in [141, 198, 197]. Software developed in this chapter is available at the following url: https://bitbucket.org/mclab/treatmentdesign.



(a) Time share of digital twin simulations (dark bars) wrt. total computation time (dark and light bars)



(b) Expanded search tree nodes (orange bars, left Y axis) vs. digital twin sizes (blue bars, right Y axis)

Figure 4.6. Computational results of our ISCT.

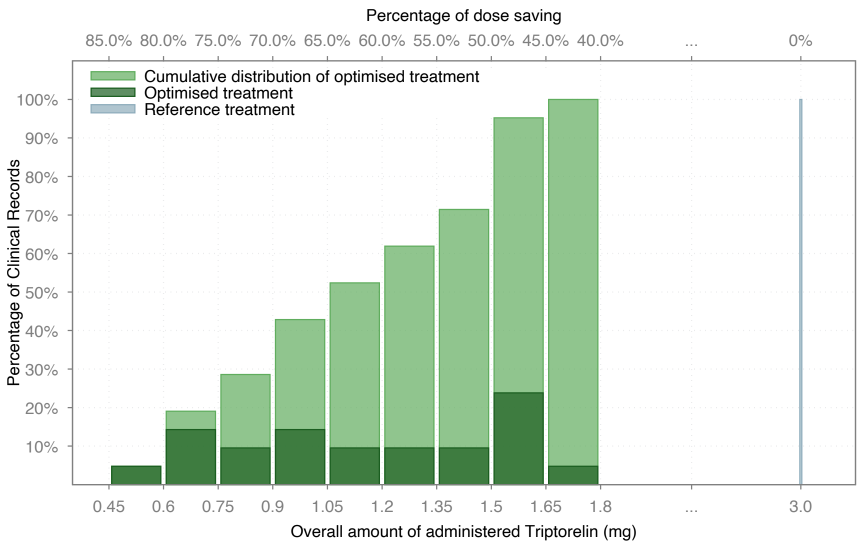


Figure 4.7. Multi-arm ISCT outcomes.

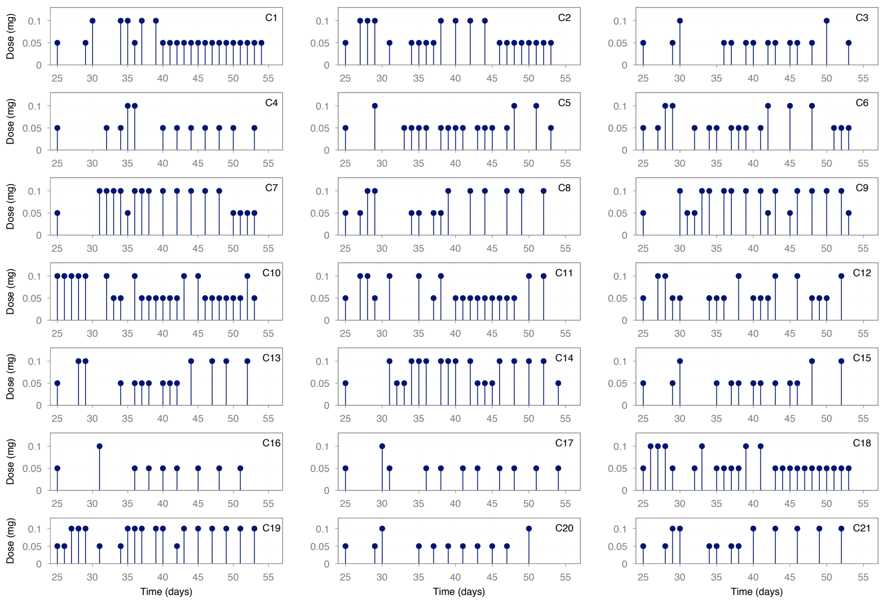


Figure 4.8. Personalised treatments computed through our multi-arm ISCT in 15 days.

Chapter 5

Reconciling interoperability and simulation efficiency

5.1 Introduction

Cyber-Physical Systems (CPSs) integrate physical (*e.g.*, mechanical, electrical, etc.) and software (*e.g.*, Software Digital Radios, SDRs, control software, etc.) subsystems. CPSs are widely used in several fields like aerospace, smart grid, manufacturing, automotive, robotics, and health-care (see, *e.g.*, [133, 123, 179, 125, 65, 90, 205, 36, 89]). As they couple the discrete and continuous dynamics of software and physical subsystems, respectively, CPSs are typically defined by means of hybrid systems (see, *e.g.*, [11]). Due to such a complex nature of CPS models, simulation-based approaches are *typically* used to support design and Verification and Validation (V&V) activities. V&V aims at checking whether the CPS model behaviour satisfies given specifications, *e.g.*, safety properties (see, *e.g.*, [172, 6]). In the literature, there are *many* examples where V&V activity is performed by means of numerical simulations (see, *e.g.*, [45, 29, 215, 164, 67, 225, 1] and [112] for a survey).

Many simulation-based software tools are available to support CPS design such as, *e.g.*, AUTOSAR, Automation Studio, AVL Cruise, CATIA, ControlBuild, Simulink, dSpace, EnergyPlus, IBM Rational Rhapsody, ICOS, IGNITE, Dymola, JModelica, MapleSim, Ptolemy II and Virtual Engine. On one side, the increasing availability of those software tools enables designers to choose the tool chain that best suits their needs. On the other side, such an availability poses huge interoperability (model exchange) and integration (co-simulation) challenges between CPSs modelled using different languages and/or tools.

Furthermore, since physical and/or software subsystems are *usually* designed by different companies (*e.g.*, Original Equipment Manufacturers, OEMs), it is also crucial to preserve Intellectual Property (IP) (see, *e.g.*, [68, 217, 153]). As a result, even carrying out simulations of those models may pose problems.

To overcome such interoperability problems, a standardised format, namely, Functional Mock-up Interface (FMI), has been proposed in 2010 as an open standard. Models adhering to FMI are called Functional Mock-up Units (FMUs). The current version, *i.e.*, FMI 2.0 [27], enables both Model Exchange (ME) and Co-Simulation (CS). ME refers to, *e.g.*, the usage of FMUs within different simulation environments, while CS refers to, *e.g.*, the distributed simulation of heterogeneous systems coupling together several FMUs. Recently, another standard has been defined, *i.e.*, System Structure and Parametrization (SSP), to describe relationships among systems of interconnected FMUs and their parametrisation in order to be used in different simulation environments [160].

Typically, simulation-based V&V of CPSs requires exploring different simulation scenarios, *i.e.*, sequences of controllable and/or uncontrollable exogenous inputs. In this setting, to avoid simulating many times a common prefix of different scenarios, the simulator state is saved in order to be restored later as a start state. This improves efficiency of simulation-based V&V approaches by simulating only once the same prefix of different simulation scenarios (see, *e.g.*, [145] and citations thereof).

Not only V&V approaches benefit from the usage of these Application Programming Interfaces (APIs), but also, *e.g.*, approaches for the automated synthesis of plans and control strategies such as [151, 17, 4, 60, 188] where simulation-based Model Predictive Control (MPC) methods are used. In particular, in [188], a MPC approach is employed for the design of an insulin pump exploiting a Virtual Physiological Human (VPH) model of glucose metabolism. Also, simulation-based approaches for In Silico Clinical Trials (ISCT) take advantage of the above optimisation. Indeed, in a ISCT setting, different sequences of drug administrations (*i.e.*, exogenous inputs) are tested on different Virtual Patient (VP) models by means of simulations. In this respect, our methodology presented in Chapter 4 relies on these simulator capabilities (see Section 4.2.2.3.2) to efficiently computes personalised pharmacological treatments.

To this end, modern simulators (such as, *e.g.*, Simulink [155]) offer their own API allowing to save and restore the internal simulator state on demand. As well as the other simulators, also the FMI 2.0 API specifies a way of saving and restoring internal FMU states. This feature is crucial to increase efficiency of simulation-based V&V.

Unfortunately, the implementation of such a feature is not mandatory according to FMI 2.0 specifications. Hence, even if the FMI standard is currently adopted by several modelling environments (see [27] for a full list) only a few commercial software implement save-and-restore feature within their generated FMUs.

Among them we note Dymola, a state-of-the-art modelling and simulation environment [54] that is based on Modelica [80], an open-standard language for modelling dynamical systems.

Modelica is an object-oriented, equation-based language for model-based development. Moreover, it allows the definition of complex dynamical systems as a network of smaller subsystems. Currently, Modelica is widely used by many companies and it is adopted by several simulation environments: commercial (*e.g.*, SimulationX [69] and SystemModeler [221]) as well as open source (*e.g.*, Open Modelica [174] and JModelica [161]). However, none of the currently available open-source Modelica environments implement the FMI 2.0 optional feature for saving and restoring FMU states. As a result, the interoperability enabled by the FMI standard cannot be fully exploited for V&V when using those open-source environments.

This motivates developing such a feature for Modelica-based and open-source CPS simulation environments.

In the literature the save-and-restore feature has been widely exploited for formal verification both for finite state systems (see, *e.g.*, [97, 98, 62, 58]) as well as, in a simulation-based framework, for CPSs (see, *e.g.*, [140, 144, 145]). We refer the readers to those references and citations therein for general considerations and more details about algorithms that exploit such a feature.

In this chapter, we provide methods and tools to implement FMI 2.0 functionalities that save and restore the internal FMU state for ME FMUs and we focus on JModelica modelling and simulation environment.

Furthermore, we present experimental results to evaluate the correctness of our proposed implementation. Finally, we also conduct an analysis of performance focusing on V&V approaches that drive given FMUs by means of simulations. To do so, we analyse 934 FMUs generated from benchmarks models taken from widely-used repositories and we show that, using our tool, a V&V activity in the style of [140] is, on average, 22 times faster than without it.

We remark that such a proposed implementation closes an important gap between commercial and open-source Modelica environments. Indeed, it enables the application of the above-mentioned simulation-based V&V approaches to CPSs modelled within open-source Modelica environments. Furthermore, it also fosters the development of new approaches.

Moreover, the aim of the work presented in this chapter is also to encourage developers of open-source simulation environments to implement modern functionalities of current FMI specifications; and to push towards future FMI specifications that include a mandatory implementation of such functionalities. For these reasons, our proposed implementation is free and *publicly available* at the following repository: https://bitbucket.org/mclab/jmodelica.org, as a fork of the JModelica 2.1 open-source distribution, which is developed by Modelon[1] and currently available

[1]https://www.modelon.com

upon request.

5.2 Methods

In this section, first, we introduce FMI 2.0 for ME with a particular emphasis on JModelica FMU implementation. Second, we describe the steps needed to enable saving and restoring internal FMU states. Finally, we present our approach to evaluate the correctness of our implementation and to measure performance when adopting such implementation within simulation-based V&V methods.

5.2.1 FMI 2.0 for Model Exchange

FMI ME 2.0 [26] specifies an interface to the model of a dynamical system defined by differential, algebraic and/or difference equations. An executable implementing FMI is called FMU. Any simulation environment supporting FMI can simulate an FMU produced by any modelling environment. In particular, an FMU is a ZIP archive consisting of the following parts. First, C code and/or binaries implementing functions defined by the FMI API. Second, an Extensible Markup Language (XML) document with model metadata including, *e.g.*, model identifier, names of variables and capability flags. The latter allows omitting implementation of non-mandatory functionalities. For example, the `canGetAndSetFMUstate` and `canSerializeFMUstate` capability flags indicate whether FMU supports functionality to save/restore and serialize/deserialize its internal state or not.

5.2.2 Enable saving and restoring of JModelica FMU internal state

JModelica generates an FMU starting from a model written in Modelica. To do so, Modelica equations are translated into C code and compiled to produce an FMU. The output FMU consists also of a library (the same for all models) implementing the functions of the FMI standard. In JModelica, such a library is called *RuntimeLibrary*, and contains also relevant data structures responsible for the internal FMU state.

The internal FMU state is a snapshot containing all the information needed to simulate the FMU starting from the moment when the state was retrieved.

Figure 5.1 depicts the conceptual UML class diagram of such FMU internals focusing on the components of the internal FMU state. The main class modelling an FMU instance is `jmi_t`. The internal state of a JModelica FMU is composed of the following components.

Values of model variables. `jmi_t` contains several arrays of real values (*e.g.*, `z` and `z_last`) to store the information related to the actual values of model variables.

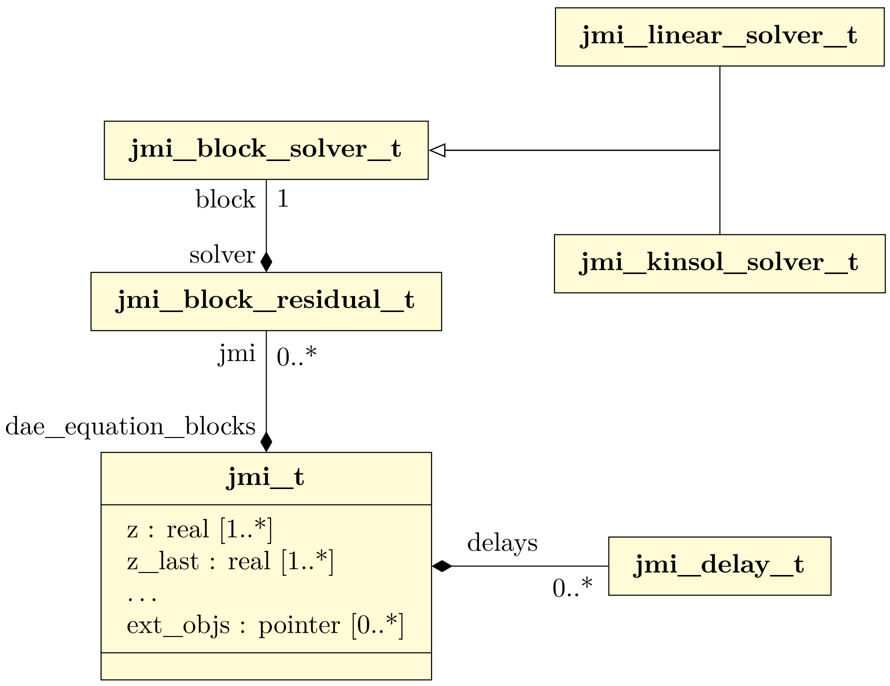


Figure 5.1. Conceptual UML class diagram of JModelica FMU implementation.

Delay buffers. Modelica equations can contain occurrences of `delay(expr,t,tmax)` (*delay operator*) that return `expr(time-t)`, *i.e.*, the value of the expression `expr` `t` time units in the past. JModelica implements such an operator by maintaining a buffer (`jmi_delay_t`) containing past values of expression `expr` from current time back to `time-t`. the optional argument `tmax` guarantees an upper bound for the buffer size when `t` is not constant during simulation. The buffer size increases until the current time is greater than `t` (`tmax` when `t` is variable). After that, the size remains constant.

Internal state of algebraic solvers. Algebraic equations are split into blocks (`jmi_block_residual_t`) that can be solved independently. Each such block is equipped with an instance of a linear (`jmi_linear_solver_t`) or a non-linear (`jmi_kinsol_solver_t`) solver. These solvers are stateful, so their internal state also makes part of the FMU state.

Internal state of external objects. Modelica allows to call external (*e.g.*, defined in C) functions (array `ext_objs` of function pointers), that can have their own state. Note that there is no way to retrieve the state of such external objects, since they are completely opaque. For example, Modelica Standard Library (MSL) has the CombiTimeTable block that computes its

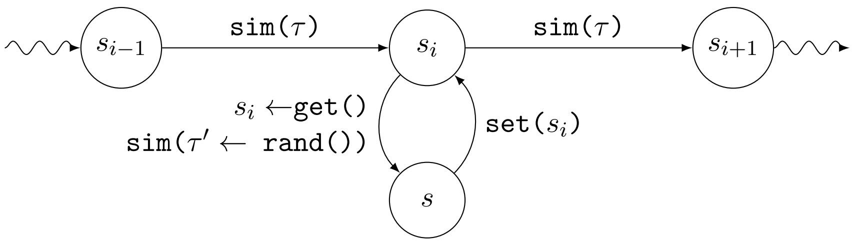


Figure 5.2. A glimpse of our strategy to evaluate correctness.

output signal by interpolation in a table. It is implemented using C functions that read a table from a file into an in-memory data structure and allow to query its values. From the JModelica FMU *RuntimeLibrary* perspective, such data structure is just an opaque pointer and there is no way to access it.

Note that, since the above data structures are static, FMU states have a constant size (in bytes).

We extend JModelica FMU *RuntimeLibrary* by suitably implementing the following FMI functions:

- `fmi2GetFMUstate()`. This method retrieves the current state of a given FMU as an in-memory data structure, which is opaque to the user. We refer to this method in the text as `get()`.
- `fmi2SetFMUstate()`. This method replaces the current state of an FMU with a given FMU state previously retrieved using `fmi2GetFMUstate()`. We refer to this method in the text as `set()`.
- `fmi2FreeFMUstate()`. This method frees up the memory occupied by a given FMU state retrieved using `fmi2GetFMUstate()`.
- `fmi2SerializeFMUstate()`. This method serializes the FMU state retrieved using `fmi2GetFMUstate()` into a byte array that can be stored in a file or sent over the network.
- `fmi2DeserializeFMUstate()`. This method deserializes a given byte array into an FMU state that can be then passed to `fmi2SetFMUstate()`.
- `fmi2SerializedFMUstateSize()`. This method returns the size of a byte array that is enough to serialize a state of the current FMU.

5.2.3 Correctness evaluation strategy

To validate our methods to get and set complete FMU states, we have to check that a call to `set()` correctly replaces the current FMU state with an FMU state obtained by a previous call to `get()`. To do so, we need to verify that further simulations are correctly computed accordingly to the actual FMU state.

In particular, we define the following two ways of performing a simulation of an input FMU.

Given an FMU and a value $\tau \in \mathbb{R}_{0+}$, let s be the current state of the input FMU, we denote with $s' = \texttt{sim(}\tau\texttt{)}$ a function that simulates the input FMU advancing its current state s for τ time units and reaching the state s'.

Given an FMU and values $\tau \in \mathbb{R}_{0+}$, $\tau' \in \mathbb{R}_{0+}$, let s be the current state of the input FMU, we denote with $s' = \texttt{sim}^{\star}\texttt{(}\tau, \tau'\texttt{)}$ a function that executes the following instructions:

1. $s \leftarrow$`get()`;
2. `sim(`τ'`)`;
3. `set(`s`)`;
4. $s' \leftarrow$ `sim(`τ`)`.

For the sake of readability and without loss of generality, we omit the call to `fmi2FreeFMUstate()` in the above instructions as its correctness does not affect our validation process.

Intuitively, a call to $\texttt{sim}^{\star}\texttt{(}\tau, \tau'\texttt{)}$, for any $\tau' \in \mathbb{R}_{0+}$, is semantically equivalent to a call to `sim(`τ`)`, since they both advance the current FMU state for τ units of time. This leads to the following remarks.

Remark 5.1. *Given an FMU and a value $\tau \in \mathbb{R}_{0+}$ let s be the current state of the input FMU. Then for all $\tau' \in \mathbb{R}_{0+}$, `sim(`$\tau$`)` and $\texttt{sim}^{\star}\texttt{(}\tau, \tau'\texttt{)}$ reach exactly the same state, formally $\texttt{sim(}\tau\texttt{)} = \texttt{sim}^{\star}\texttt{(}\tau, \tau'\texttt{)}$*

Remark 5.2. *Given an FMU and a value $\tau \in \mathbb{R}_{0+}$, let s be the initial state of the input FMU, let $\mathbf{s}_\tau = s_0, s_1, s_2, \ldots, s_n$ and $\mathbf{s}^{\star}_\tau = s^*_0, s^*_1, s^*_2, \ldots, s^*_n$ be the sequences of FMU states reachable from s by consecutive executions of `sim(`τ`)` and $\texttt{sim}^{\star}\texttt{(}\tau, \tau'\texttt{)}$, respectively, for all $\tau' \in \mathbb{R}_{0+}$. If both `get()` and `set()` are implemented correctly then, for all $i \in \{0, \ldots, n\}$, $s_i = s^*_i$.*

To experimentally verify the above statement, we need to check that our implementation works for all values of τ'. This is of course impossible. To overcome such an obstacle we employ a statistical approach based on hypothesis testing where τ' takes values in a bounded interval $\mathbb{B} \subset \mathbb{R}_{0+}$ (see, *e.g.*, [87, 148, 150]).

Given a value for $\varepsilon \in (0, 1)$, we define a *null hypothesis* H_0 which states that the probability of sampling $\tau' \in \mathbb{B}$ such that the state reached by executing `sim(`τ`)` is different from the state reached by executing $\texttt{sim}^\star(\tau, \tau')$ is greater than ε. Formally H_0 is defined as: $H_0 : \Pr\{\tau' \in \mathbb{B} \,.\, \texttt{sim}(\tau) \neq \texttt{sim}^\star(\tau, \tau')\} \geq \varepsilon$.

Then we apply statistical hypothesis testing [165] and try to reject H_0 through a given number of trials N.

At each trial we randomly sample a value of $\tau' \in \mathbb{B}$ according to a uniform distribution and we check whether the state reached by $\texttt{sim}(\tau)$ is different from the state reached by $\texttt{sim}^\star(\tau, \tau')$ or not.

If within N trials we find a value τ' such that $\texttt{sim}(\tau) \neq \texttt{sim}^\star(\tau, \tau')$, then τ' is a counterexample showing that our implementation is not correct. Formally, we prove H_0 when it holds.

On the other hand, rejecting H_0 after N trials, even if it holds, introduces a *Type-I Error*.

In particular, given a value $\delta \in (0, 1)$, the probability that we make an error by rejecting H_0 when it holds is bounded by δ.

This is shortly stated by saying that H_0 is rejected with statistical confidence $1-\delta$. Finally, we conclude that the probability to sample τ' such that the state reached by executing $\texttt{sim}(\tau)$ is different from the state reached by executing $\texttt{sim}^\star(\tau, \tau')$ is less than ε, formally $\Pr\{\tau' \in \mathbb{B} \,.\, \texttt{sim}(\tau) \neq \texttt{sim}^\star(\tau, \tau')\} < \varepsilon$.

By exploiting results of [87], given $\delta \in (0, 1)$ and $\varepsilon \in (0, 1)$, the number of trials is computed as $N = \lceil \log(\delta) / \log(1 - \varepsilon) \rceil$.

```
1 function evaluateCorrectness():
    Input: FMU
    Input: $\tau$, simulation duration
    Input: $\varepsilon \in (0, 1)$
    Input: $\delta \in (0, 1)$
    Input: $\mathbb{B} \subset \mathbb{R}_{0+}$
    Output: either (True, −, −) or a counterexample (False, $\tau'$, $i$)
2   N ← $\left\lceil \frac{\log(\delta)}{\log(1-\varepsilon)} \right\rceil$;
    /* $H_0 = \Pr\{\tau' \in \mathbb{B} \,.\, sim(\tau) \neq sim^\star(\tau, \tau')\} \geq \varepsilon$ */
3   for $i \in [1, N]$ do
4     $\tau'$ ← rand($\mathbb{B}$);
5     $s_i$ ← sim($\tau$); $s$ ← sim*($\tau, \tau'$);
6     if $s_i \neq s$ then
7       return (False, $\tau'$, $i$) /* $H_0$ is proved */
8   return (True, −, −) /* $H_0$ is rejected */
```

Algorithm 2: Hypothesis testing approach to evaluate correctness of `set()` and `get()`.

Algorithm 2 describes our *Hypothesis Testing* approach and Figure 5.2 sketches our correctness evaluation strategy.

Note that at Line 6 of Algorithm 2, if, for a given τ', `sim(`τ`)` $\neq$ `sim`*`(`τ, τ'`)` is true, we return such a τ' as a counterexample meaning that by simulating the input FMU i times we reach a state proving that `get()` and `set()` are not working correctly.

The above considerations prove the following theorem.

Theorem 5.1. *Given an FMU having* `get()` *and* `set()` *implemented, values* $\varepsilon, \delta \in (0,1)$*, a bounded interval* $\mathbb{B} \subset \mathbb{R}_{0+}$ *and a value for* $\tau \in \mathbb{R}_{0+}$*, Algorithm 2 is such that:*

1. *it terminates in* N *steps, where* $N = \left\lceil \frac{\log(\delta)}{\log(1-\varepsilon)} \right\rceil$*;*

2. *when it returns* `True`*, with confidence* $1-\delta$*:* $\Pr\{\tau' \in \mathbb{B} \,.\, \mathit{sim}(\tau) \neq \mathit{sim}^\star(\tau, \tau')\} < \varepsilon$*;*

3. *when it returns* `False`*, we have a counterexample, i.e., a value for* $\tau' \in \mathbb{B}$*, proving that* `get()` *and* `set()` *are not correctly implemented.*

In Section 5.3.3 we apply Algorithm 2 on different FMUs in order to validate our implementation of `get()` and `set()`.

5.2.4 Performance evaluation strategy

To evaluate performance of our implementation, we focus on simulation-based V&V approaches. As anticipated in Section 5.1, simulation-based V&V of CPSs requires to explore all simulation scenarios. A simulation scenario is a sequence of exogenous inputs to be injected to the given CPS model, *i.e.*, FMU under verification. In this setting, the given FMU (representing the CPS under verification) is driven in the space of all possible simulation scenarios by means of a visit (see, *e.g.*, [143, 142, 146] and citations thereof). The space of all simulation scenarios is defined as a tree where each edge is labelled with an exogenous input and each node represents the FMU state reached by injecting the sequence of inputs associated to the path leading to that node.

As an example we can consider an FMU defining a simple hybrid system (see, *e.g.*, [11]) having only one state variable. In particular, let x be a signal (i.e., a real-valued function of time) and, given a positive real number T (time step), let u be T-piecewise constant signal, i.e., a signal changing its value only at time instants of the form kT ($k = 0, 1, 2, 3, \ldots$). In the following we assume that u takes values in the set $\{-1, 1\}$ (i.e., for all t, $u(t) \in \{-1, 1\}$). When writing equations, as usual, we will write x for $x(t)$ and $\dot{x}$ for $\dot{x}(t)$.

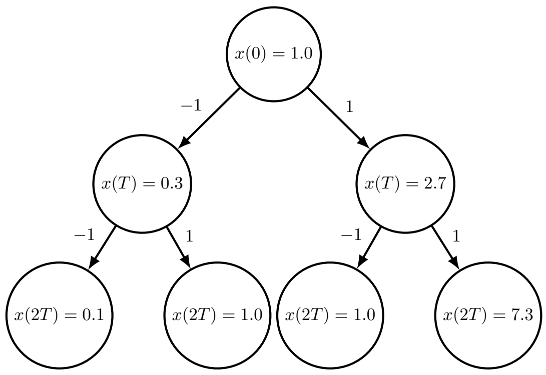


Figure 5.3. Example of a tree of simulation scenarios. Nodes denote FMU states whereas edges denote injected input values.

Given signals x and u as above, we define the behaviour of our simple example of hybrid system by means of the following differential equation:

$$\dot{x} = \begin{cases} -x & \text{if } u \text{ is } -1 \\ x & \text{if } u \text{ is } 1. \end{cases}$$

Let $x(0) = 1$ be the initial state of this FMU and $2T$ the simulation horizon. Figure 5.3 depicts all possible simulation scenarios starting from such an initial state and illustrates all state traversed by the FMU when the input function $u(t)$ can change value (i.e., at each kT with $k = 0, 1, 2$). Note that, as described above, states reached by different sequence of inputs (tree paths) are different for us.

Typically, the goal of a simulation-based V&V activity is to search for an input sequence driving the system to an *undesirable* (error) state (e.g., to verify a safety property stating that *nothing bad* ever happens) or to a *desirable* state (e.g., to verify a liveness property stating that *something good* sooner or later will happen). Checking a liveness property typically requires looking at the future evolution of the system. As a result, in general, liveness properties cannot be casted as safety properties (see, e.g., [137]). However, in a bounded horizon setting, such as the one in our simulation-based setting, liveness properties can be casted as safety properties (see, e.g., [24, 192, 119]) stating that a given state (desirable or undesirable) is reachable, with some suitable input, within the given time horizon. Accordingly, w.l.o.g., we can cast simulation-based V&V as the problem of finding an input sequence driving the system to a given state within the given time horizon.

If such a state is not found, the time horizon is increased until an *undesirable* or *desirable* state is found, or some upper bound is reached, namely, the provided simulation horizon h (see, *e.g.*, [75, 181]). This is equivalent to search for the shortest sequence of exogenous inputs that leads to a given state (see, *e.g.*, [41] and [62, 58, 43] in a finite state context) in the space of all simulation scenarios of length h. The simulation horizon h is typically set to a value large enough to guarantee that, with high confidence, the *undesirable* or *desirable state* (if any) is reachable with an input sequence of length at most h. To keep h small a Breadth-First Search (BFS) is used (*e.g.*, as in bounded model checking [25]). Furthermore, as anticipated in Section 5.1, such simulation-based V&V approaches are similar to several planning and optimisation approaches such as, *e.g.*, [130, 211, 180, 79, 199], where simulation scenarios and exogenous inputs correspond, respectively, to plans and actions to be taken in order to drive the system into a goal state.

In this setting, in order to decrease the number of simulations and, in turn, the computation time of the V&V activity (or the optimisation task), it is important to avoid simulating many times prefixes common to different simulation scenarios. To do so, it is crucial that the given FMU has the save-and-restore state functionality implemented as FMU states can be saved in memory in order to be restored later as initial states of the simulation.

Hence, such an exploration visit in the tree of simulation scenarios is a demanding application to evaluate our proposed implementation. In particular, we distinguish two kind of approaches. The former uses FMUs that are not equipped with the save-and-restore feature of FMI 2.0. We refer to this approach in the text as *without-save-restore visit*. The latter is a *save-restore visit* that uses our generated FMUs implementing those FMI features to save and restore FMU states.

It is worth noting that during a *save-restore visit*, since the state space to explore can be huge, to keep in memory the whole state space can be infeasible. To this end, several solutions have been devised in the literature (see, *e.g.*, [140, 144, 145]). The general idea is to dynamically choose the best states to store and those to forget (compatibly with memory constraints). Also, whenever no further inputs can be injected from the current node of the tree, the corresponding simulator state can be removed from memory. We refer the readers to those references for more details and algorithms that efficiently satisfy memory constraints during the visit. Accordingly, here we focus on evaluating performance of our implementation of the save-and-restore feature in terms of simulation time.

The remainder of this section is organised as follows. First, we define the space of simulation scenarios of a given CPS model (*i.e.*, FMU under verification) as a tree (Section 5.2.4.1). Second, we show that the time needed to explore such a tree (for both a *without-save-restore visit* and a *save-restore visit*) strictly depends on the time needed to drive the input FMU through each node of the tree (Sections 5.2.4.2 and 5.2.4.3). Last, we quantify the speed-up of a *save-restore visit* (Section 5.2.4.4).

5.2.4.1 Tree definition

In our setting, we perform a visit of a balanced finite tree of depth $h > 1$, where each tree node at depth $0 < i \leq h$ has a constant branching factor $b > 1$. The tree root corresponds to the initial state of the given FMU, while nodes correspond to FMU states that can be reached through simulations from the initial state. The edges of the tree correspond to possible actions that can be taken during the visit, *i.e.*, different values for the FMU exogenous inputs. For example, in the tree of Figure 5.3, the branching factor, b, and the maximum depth, h, are both 2.

Furthermore, to simplify calculations, during the visit we assume that the simulation of the input FMU is advanced by a fixed quantity of $\tau \in \mathbb{R}_+$ time units each time a new node is being visited (in our example τ corresponds to T, *i.e.*, 1 second). Also, to simplify our analysis, we assume that the execution time of a simulation depends on the current FMU state and on the simulation duration, *i.e.*, τ, and not on values of exogenous inputs. This might not be strictly true for each simulation, but it is a very reasonable assumption on average for typical CPS models.

5.2.4.2 Cost of a *without-save-restore visit*

A *without-save-restore visit* aims at driving FMUs without get-and-set state capabilities. Hence, when we have to explore a node in the tree, we need to simulate the given FMU from its initial state up to the state denoted by such a node. This means that a node at depth $i > 0$ can be reached after a simulation of $i\tau$ time units from the initial state of the given FMU. We denote such operation with $\texttt{sim}(i\tau)$. For example, in Figure 5.3, the state $x(2) = 0.1$ can be reached by means of a $\texttt{sim}(2\tau)$ operation, where $\tau = 1$ second, and by injecting $u(0) = -1$ and $u(1) = -1$. As already stated in Section 5.2.3, `sim()` can be defined according to FMI 2.0 specifications.

We define our cost function as $\mathcal{C}(i) = \mathcal{T}(\texttt{sim}(i\tau))$, which measures the execution time, $\mathcal{T}$, to reach a node at depth $i > 0$. Finally, we define the total execution time of a *without-save-restore visit* as the sum of the costs of each single node in the tree:

$$\sum_{i=1}^{h} \mathcal{C}(i)\, b^i \tag{5.1}$$

5.2.4.3 Cost of a *save-restore visit*

A *save-restore visit* aims at driving FMUs by exploiting their get and set capabilities in order to simulate only once each prefix common to different paths of the given tree. To do so, each time a node at depth $i > 0$ has to be visited, we perform the following steps:

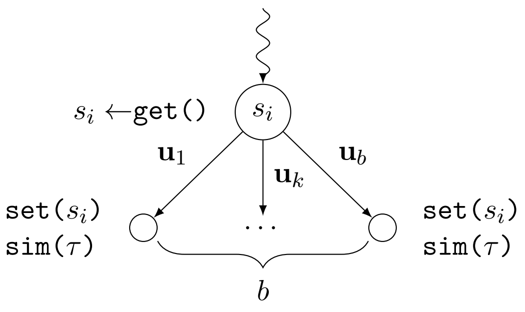


Figure 5.4. Usage of `set()` and `get()` to explore nodes during a *save-restore visit*

1. `set()`, to restore (load) the state corresponding to the node at depth $i-1$ as a start state of the given FMU;

2. `sim(`τ`)`, to advance the current FMU state to the state reached after a simulation of τ time units also by injecting the value for exogenous inputs corresponding to the traversed edge.

3. `get()`, to store (save), *e.g.*, on disk, the reached FMU state for further steps.

For the sake of readability, we omit the time needed to perform a `fmi2FreeFMUstate()` because it is negligible. Figure 5.4 outlines the above steps.

As in the previous section, we define a cost function, *i.e.*, $\mathcal{C}^\star(i)$, which denotes the execution time spent to visit a node at depth $i > 0$ during a *save-restore visit*, as follows.

$$\mathcal{C}^\star(i) = \mathcal{T}(\frac{1}{b}\,\texttt{get()} + \texttt{set()} + \texttt{sim}(\tau)).$$

Note that the `get()` is performed at depth $i-1$ (see Figure 5.4) as the corresponding retrieved FMU state will be then used as a start state for all the children of that node. However, to simplify the formulation we count such `get()` time in the child nodes. Hence, each child node (at depth i) contributes $\frac{1}{b}$`get()` to the total `get()` time, *i.e.*, the total `get()` time is amortised by the branching factor.

Accordingly, along the lines of Equation (5.1), the total execution time needed by a *save-restore visit* to explore the whole tree space is equal to the sum of the costs of each node:

$$\sum_{i=1}^{h} \mathcal{C}^\star(i)\; b^i \tag{5.2}$$

5.2.4.4 Speed-up of a *save-restore visit*

We define the speed-up, $S(h,b)$, as the ratio between the cost of a *without-save-restore visit*, Equation (5.1), and the cost of a *save-restore visit*, Equation (5.2), on a perfectly balanced tree having branching factor b and depth h:

$$S(h,b) = \frac{\sum_{i=1}^{h} \mathcal{C}(i)\ b^i}{\sum_{i=1}^{h} \mathcal{C}^\star(i)\ b^i} \tag{5.3}$$

When the above ratio is greater than 1 it means that a *save-restore visit* is faster than a *without-save-restore visit*.

Furthermore, we note that the above formula, *i.e.*, Equation (5.3), can be simplified under the following assumptions.

First, the time spent for simulating an FMU for $i\tau$ time units, *i.e.*, $\mathcal{T}(\texttt{sim}(i\tau))$, is approximately equal to $i \times \overline{\mathcal{T}}(\texttt{sim}(\tau))$, that is i times the average execution time of a simulation of length τ.

Second, we can use average execution times of `get()` and `set()`, *i.e.*, $\overline{\mathcal{T}}(\texttt{get()})$ and $\overline{\mathcal{T}}(\texttt{set()})$, respectively. As Section 5.2.2 describes, this is motivated by the fact that states of a given FMU have a constant size (in bytes).

In Section 5.3.4, we experimentally show that such assumptions are reasonable.

We rewrite $\mathcal{C}(i)$ as $\mathcal{C}(i) = i \times \overline{\mathcal{T}}(\texttt{sim}(\tau))$ and $\mathcal{C}^\star(i)$ as $\mathcal{C}^\star(i) = \frac{1}{b}\ \overline{\mathcal{T}}(\texttt{get()}) + \overline{\mathcal{T}}(\texttt{set()}) + \overline{\mathcal{T}}(\texttt{sim}(\tau))$.

Hence, the speed-up of Equation (5.3), $S(h,b)$, can be written as follows:

$$S(h,b) = \frac{\overline{\mathcal{T}}(\texttt{sim}(\tau)) \sum_{i=1}^{h} ib^i}{\left(\frac{1}{b}\ \overline{\mathcal{T}}(\texttt{get()}) + \overline{\mathcal{T}}(\texttt{set()}) + \overline{\mathcal{T}}(\texttt{sim}(\tau))\right) \sum_{i=1}^{h} b^i}.$$

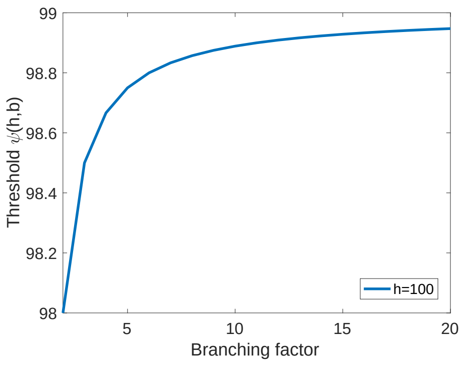
This leads to the following proposition.

Proposition 5.1. *Given* $\mathcal{C}(i) = i \times \overline{\mathcal{T}}(\texttt{sim}(\tau))$ *and* $\mathcal{C}^\star(i) = \frac{1}{b}\ \overline{\mathcal{T}}(\texttt{get()}) + \overline{\mathcal{T}}(\texttt{set()}) + \overline{\mathcal{T}}(\texttt{sim}(\tau))$*, the speed-up of a* save-restore visit *is greater than 1, i.e.,* $S(h,b) > 1$*, when the following equation is satisfied:*

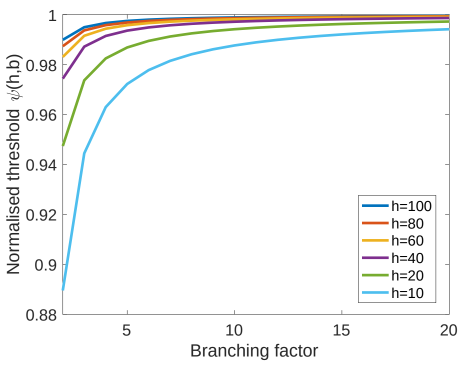
$$\frac{\frac{1}{b}\ \overline{\mathcal{T}}(\texttt{get()}) + \overline{\mathcal{T}}(\texttt{set()})}{\overline{\mathcal{T}}(\texttt{sim}(\tau))} < \psi(h,b) \tag{5.4}$$

where $\psi(h,b)$ *is a threshold defined as:*

$$\psi(h,b) = \frac{\sum_{i=1}^{h} ib^i}{\sum_{i=1}^{h} b^i} - 1 = \frac{hb^{h+1} - (h+1)b^h + 1}{(b-1)(b^h-1)} - 1. \tag{5.5}$$



(a)



(b)

Figure 5.5. Threshold $\psi(h, b)$ computed for $h = 100$ (a) and for different values of h (b). In the right figure (b), our threshold is also normalised by $h - 1$ for presentation purposes.

Remark 5.3. *When b is very large, formally $b \to \infty$, and $h > 1$, the speed-up threshold $\psi(h, b)$ approaches to $h-1$, formally $\psi(h, b) \to h-1$. Thus, Proposition 5.1 can be rephrased saying that $S(h, b) > 1$ when the following condition is satisfied.*

$$R(h, \tau) = \frac{\overline{\mathcal{T}}(\mathtt{set()})}{(h-1)\overline{\mathcal{T}}(\mathtt{sim}(\tau))} < 1. \tag{5.6}$$

Intuitively, Remark 5.3 says that when the branching factor of a given tree is sufficiently large and the ratio $R(h, \tau)$ is less than 1, we expect that the use of get/set state functionality speeds up a *without-save-restore visit*. Note that, since the `get()` is executed once for each node of the tree and the retrieved state is then used as many times as the value of the branching factor b of that node (*i.e.*, for each child node), the inefficiency of a `get()` is amortised among those child nodes. Hence, even if for some model the average time of a `get()` is high because the operation is inefficient, the term $\frac{1}{b}$`get()` of Equation (5.4) becomes negligible. In Section 5.3.4.1 we show that our assumptions that motivate this remark are reasonable for a large set of benchmark models.

In Figure 5.5a we show that, starting from small values of b, the threshold $\psi(h, b)$ rapidly reaches values towards the limit $h - 1$. This is true also for different values of h as Figure 5.5b shows.

Hence, under our assumptions, given an FMU and a tree having a low branching factor, namely 4-5, if the average execution time needed to perform a `set()` operation is lower than the time to perform a simulation of a scenario of length $h - 1$, then the speed-up of a *save-restore visit* with respect to a *without-save-restore visit* is greater than 1.

In Section 5.3.4, we present experimental results regarding the speed up computation on different FMUs by taking into account trees of different depths and branching factors.

5.3 Experimental Results

In this section, we present experimental results to evaluate the correctness and the effectiveness of our implementation against state-of-the-art case studies. After briefly describing such case studies, we then evaluate correctness and we finally show how our implementation enhances performance of simulation-based V&V approaches.

5.3.1 Case studies

In Section 5.3.4 we see that, depending on the FMU at hand, different components of the FMU architecture are involved during saving and restoring an FMU state.

To assess correctness and to evaluate performance on all such components we evaluate our implementation by using FMUs generated from the following widely established model libraries.

MSL: the official Modelica library collecting models that are developed and reviewed by the Modelica Association (version 3.2.2, https://modelica.org). It provides models and components from different engineering domains such as mechanical, electrical, magnetic, fluid, thermal and control systems. The library contains both models defining standardised interfaces or building blocks and models that are directly usable. For our purpose, we focus on the latter class of models, which comprises 385 models having 150 model variables on average.

Scalable Test Suite (STS): a Modelica library of benchmark models useful for assessing performance of large scale systems (version 1.11.4, see [37]). The library contains 16 modules covering electrical, mechanical, power and thermal domains. Such modules are scalable in terms of their size (*i.e.*, number of variables). The STS provides also 207 ready-to-run models validated by the authors and having an average of 2913 model variables.

BioModels Database (BMD): a well-known repository of mathematical models of biological systems taken from the scientific literature [121]. A subset of these models, consisting of manually curated models, is widely-used as a benchmark for SBML simulators. From such a subset, we used models already translated

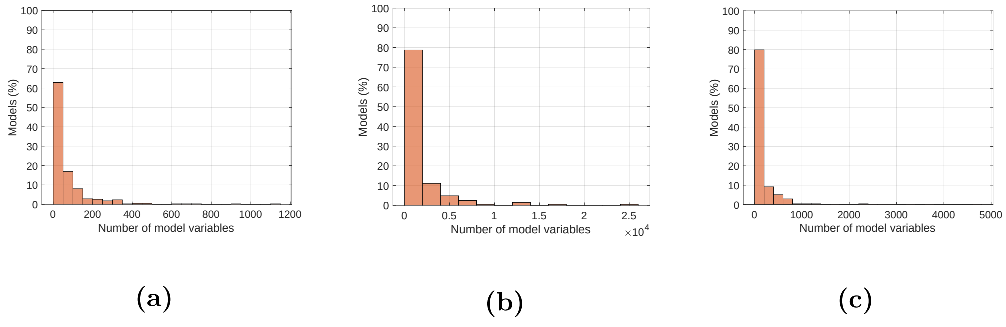


Figure 5.6. Distribution of sizes of models within MSL (a), STS (b) and BMD (c).

from SBML to Modelica and validated in [136]. In total we consider 411 models having 370 model variables on average.

Figure 5.6 shows the distribution of model sizes in terms of the number of model variables.

5.3.2 Experimental setting

All our experiments have been carried out on a High Performance Computing (HPC) infrastructure (*i.e.*, Marconi cluster at CINECA, Italy).

For each Modelica model within our datasets, *i.e.*, MSL, STS and BMD, we generated its FMU using our extended JModelica.

We manually excluded from our datasets those FMUs that are not compatible with get/set state functionality. These are FMUs that use external objects (as described in Section 5.2.2). This led us to exclude 49 (*i.e.*, 12.73%) models from MSL and 20 (*i.e.*, 9.66%) models from STS. All models within BMD have been included. Hence, in total, we excluded the 6.88% of all FMUs within our datasets.

In order to assess correctness and evaluate performance of our implementation of get/set functionality, FMUs have been simulated using the `SUNDIALS CVODE` [96] solver by means of the `PyFMI` library [12]. For each FMU, we set the value of τ to 1% of the FMU default simulation horizon, *i.e.*, the value of FMU `default experiment stop time`.

5.3.3 Correctness evaluation results

Correctness has been evaluated by means of the approach presented in Section 5.2.3. In particular, for each of the 934 FMUs, we ran our Algorithm 2 to assess that our implementation correctly restores a previously-saved FMU state.

To do so, we consider as $\mathbb{B}$ values from 0 to the default simulation horizon of the given FMU and we set $\delta = 8\%$ and $\varepsilon = 2.5\%$. Thus the number of trials has been set to $N = 100$.

During the sampling process, the inequality of FMU states (Line 6 of Algorithm 2) has been checked through a bit-wise comparison. For all our 934 FMUs our Algorithm 2 returned true, proving that our implementation is correct with a degree of statistical confidence of 92%.

5.3.4 Performance evaluation results

In this section, we describe experimental results of our performance evaluation according to our strategy described in Section 5.2.4. In particular, in Section 5.3.4.1 we evaluate performance of our implementation by a preliminary analysis using results of Remark 5.3. Then, in Section 5.3.4.2 we compute the speed-up of a *save-restore visit* with respect to a *without-save-restore visit* on trees with different values of depth and branching factor.

5.3.4.1 Preliminary analysis

As a preliminary analysis, for each FMU, we computed the ratio $R(h, \tau)$ (see Remark 5.3) to compare the execution time of a `set()` with the execution time needed for simulating the given FMU.

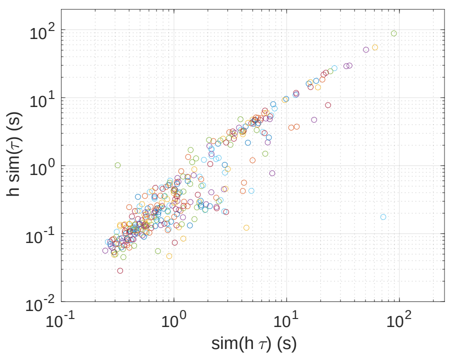
Such an analysis has been conducted by taking into account different values for the depth of the tree, *i.e.*, $h \in [1, 100]$.

To perform such an analysis, we first show that our assumptions (*i.e.*, those described in Section 5.2.4.3) are reasonable for our case-study FMUs. To do so, in Figure 5.7, we compared the execution time of a simulation of length $i \times \tau$ with the execution time of i simulations of length τ. As we see, these two quantities are highly correlated. Indeed, the Pearson correlation coefficient, ρ, for MSL, STS and BMD is 0.88, 0.79 and 0.99, respectively.

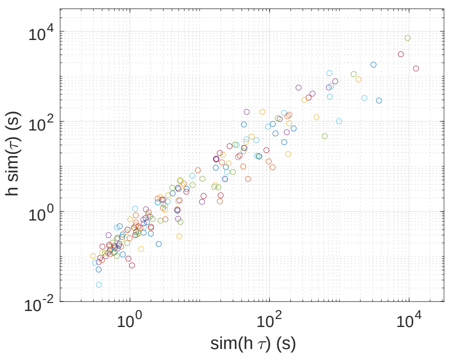
Moreover, in Figure 5.8, for each FMU, we show that the execution time (in seconds) spent to perform a `set()` operation is almost constant during the FMU simulation (from its start time to its simulation horizon).

There is also an atypical behaviour for two FMUs within BMD. In such models, the `delay` operator is used and parameter `tmax` is always set to be as large as possible (much larger than the given FMU simulation horizon). This causes the size of the associated delay buffers to increase after each simulation in order to store all needed delay variable values. For this reason, the size of the internal FMU state also increases together with the execution time needed to perform `set()` (and `get()` as well).

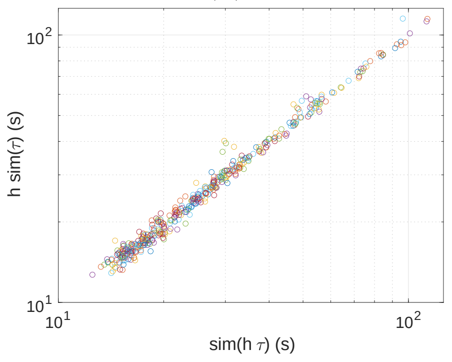
Having clarified that our assumptions are reasonable, in Figure 5.9 we show how $R(h, \tau)$ changes by varying the depth of the tree, for each FMU in our datasets.



(a)

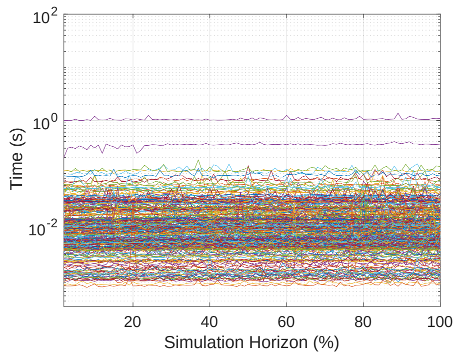


(b)

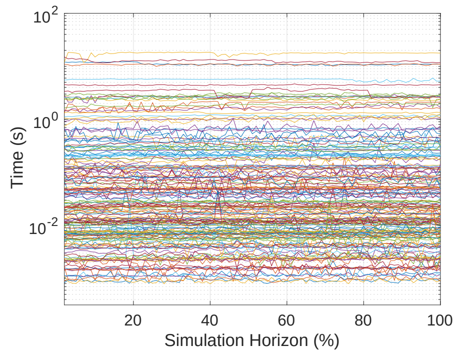


(c)

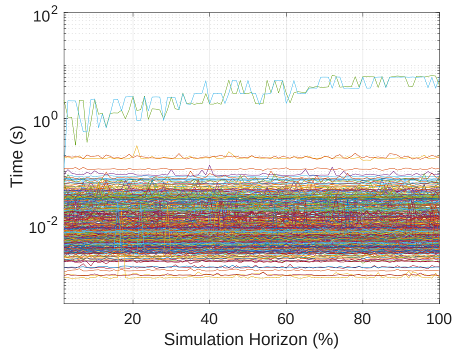
Figure 5.7. Correlation among the execution times (in seconds) of a simulation of duration $h \times \tau$ (on x axis) and h simulations of duration τ (on y axis), where $h = 100$. Each marker represents an FMU within MSL (a), STS (b) and BMD (c).



(a)

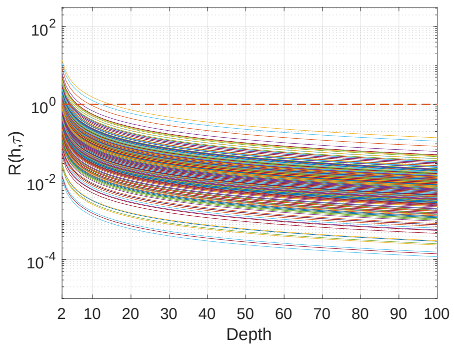


(b)

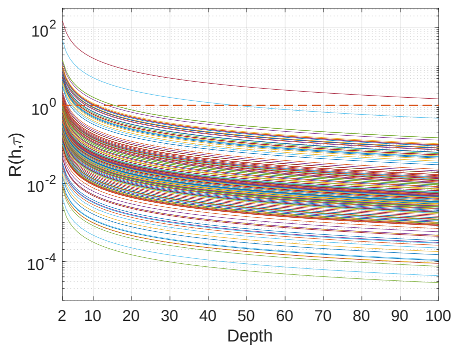


(c)

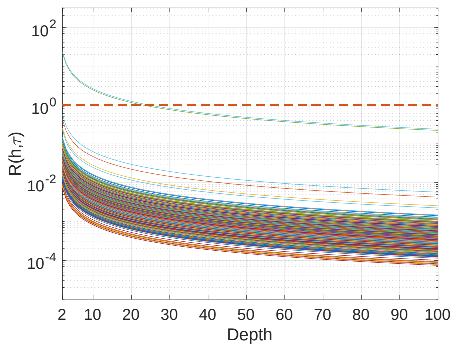
Figure 5.8. Execution times (in seconds) spent for performing `set()` during FMU simulations. Each line represents an FMU within MSL (a), STS (b) and BMD (c).



(a)



(b)



(c)

Figure 5.9. Computation of $R(h, \tau)$, where $h \in [1, 100]$. Dashed line represent threshold of $R(h, \tau)$. Each curve represents an FMU within MSL (a), STS (b) and BMD (c).

As we expected, for the majority of the analysed FMUs, the values of $R(h, \tau)$ are either below 1 or they go below 1 for very small values of depth (namely, < 20). This means that the cost of `set()` is very low with respect to simulations in terms of computation time.

Also, there is one single FMU having $R(h, \tau)$ greater than 1 for all values of depth. Such behaviour is explained by the fact that the larger the state of a given FMU the slower the execution of `set()` in terms of computation time. Hence, a *without-save-restore visit* performs better for those FMUs having a very large state (thousands of variables, as for FMUs within STS) but very fast in simulating. Of course, as we described in Section 5.2.4.3, this is strictly linked also to the depth of the given tree which affects the length of simulations to be performed in a *without-save-restore visit*.

Note that, $R(h, \tau)$ is less than 1 also for those two FMUs having a non-constant execution time for `set()` (see Figure 5.8c).

5.3.4.2 Speed-up analysis

In order to perform a more in-depth analysis, we also computed our cost functions for the *without-save-restore visit*, *i.e.*, Equation (5.1), and the *save-restore visit*, *i.e.*, Equation (5.2), and finally our speed-up formula, *i.e.*, $S(h, b)$ (see Equation (5.3)).

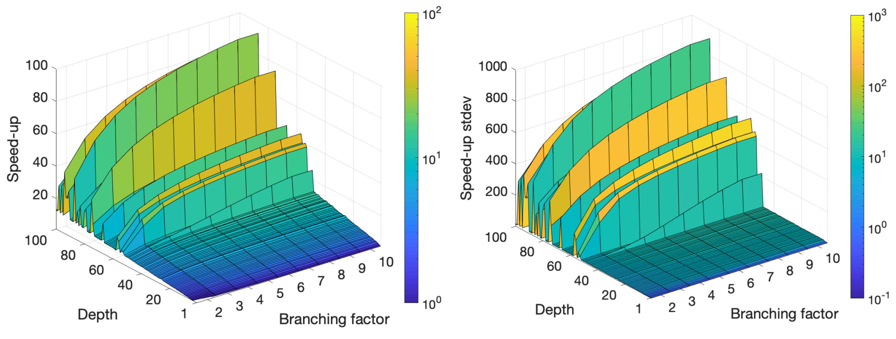
To do so, different values for the branching factor, *i.e.*, $b \in [2, 10]$, and for the depth of the tree, *i.e.*, $h \in [1, 100]$, have been taken into account. Note that, such chosen values are perfectly reasonable for real case studies. For example, in Chapter 4 our simulation-based approach consists of a backtracking-based search employed in a search space defined as a tree of depth equal to 55 and constant branching factor equal to 3.

Figure 5.10 shows the average speed-up computed among FMUs within MSL, STS and BMD when varying both the branching factor and the depth of the tree.

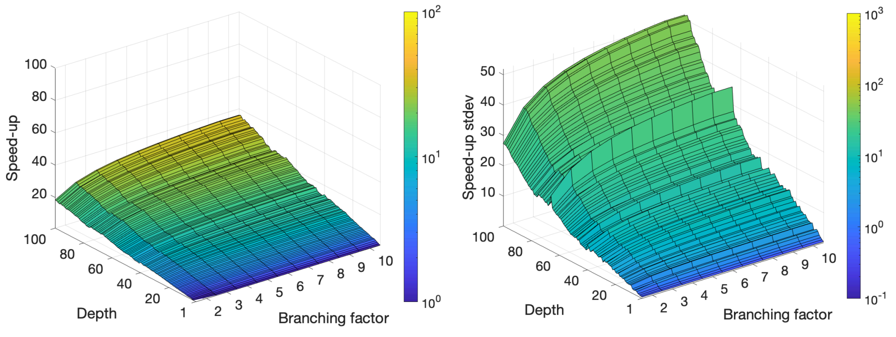
As we expected from our preliminary analysis (Section 5.3.4.1), even with a low value for the branching factor, *i.e.*, 5, and a low depth, *i.e.*, 50, a *save-restore visit* is, on average, 22 times faster than a *without-save-restore visit*, among all our case studies (standard deviation: 15.6). In particular, for FMUs within MSL, STS and BMD, we have on average a speed-up of 11.75 (standard deviation: 12.5), 14.06 (standard deviation: 14.6) and 39.84 (standard deviation: 4.8), respectively.

Furthermore, by keeping constant the given tree depth and branching factor, $h = 100$ and $b = 10$, we reach, on average, a speed-up value of 26.62 (standard deviation: 27.23), 36.11 (standard deviation: 51.44) and 86.37 (standard deviation: 12.04), among all FMUs within MSL, STS and BMD, respectively.

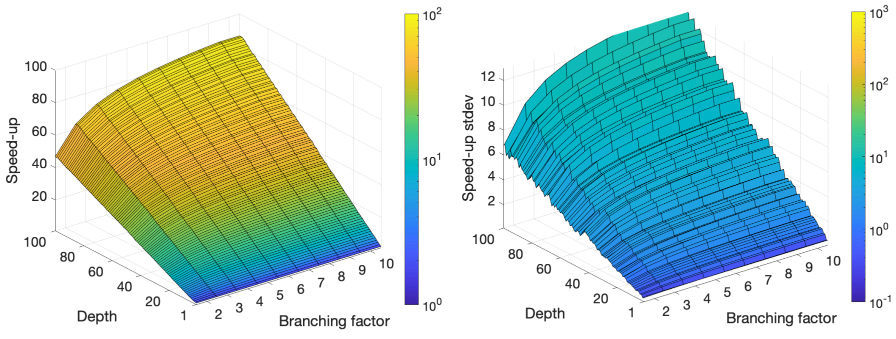
As we can note, FMUs within BMD (Figure 5.10c) achieve higher values of speed-up than FMUs within the other datasets. It is clear (starting from depth equal to 2) that the execution time of `set()` is very low with respect to simulation



(a)



(b)



(c)

Figure 5.10. Average and Standard Deviation of the speed-up (left and right heat maps, respectively) computed among FMUs within MSL (a), STS (b) and BMD (c) by varying the branching factor and the depth of the tree.

execution time. Hence, the deeper the tree (*i.e.*, the number of simulations) the higher the speed-up achieved.

Surprisingly, as Figure 5.10a shows, for some values of h the computed speed-up is around 70. This behaviour is due to the fact that, for some FMUs within MSL the numerical integrator reaches a computationally expensive integration step which requires more time to be solved, *e.g.*, a chattering effect.

Hence, having the FMU at hand equipped with `get()` and `set()` functionalities brings a huge benefit. Indeed, once this complex integration step is solved, the reached FMU state can be saved and restored for further simulations. Also, this explains the high standard deviation value for MSL FMUs.

Figure 5.11 clearly shows this by presenting, for each evaluated FMU, the speed-up achieved when varying the depth of the tree, while keeping constant the value of the branching factor ($b = 10$). In particular, in Figure 5.11a we see that few FMUs reach peaks of speed-up above 10 000.

Furthermore, in both Figures 5.10 and 5.11, we note that the speed-up increases almost linearly with respect to values of the depth of the tree.

Full results of this section are provided at https://bitbucket.org/mclab/getset-fmi-eval-results.

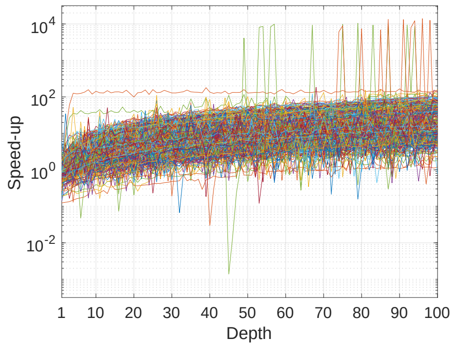
5.4 Related Work

The FMI standard is widely used for model-based design and analysis of CPSs. In the literature we find many examples in different fields (see, *e.g.*, [107, 39, 52, 83]). Both CS and ME paradigms are crucial to handle large-scale systems (see, *e.g.*, [203, 74]). The former is needed to integrate and analyse in a distributed fashion different subsystems having different characteristics as, *e.g.*, the embedded solver needed to simulate the model (see, *e.g.*, [84, 193] for recent surveys). The latter is used to, *e.g.*, import and/or export models as black-box objects among different simulation environments in order to analyse and simulate them with different solvers. As anticipated in Section 5.1, we focus on Modelica-based open-source platforms compliant with the last version of FMI for ME, namely 2.0.

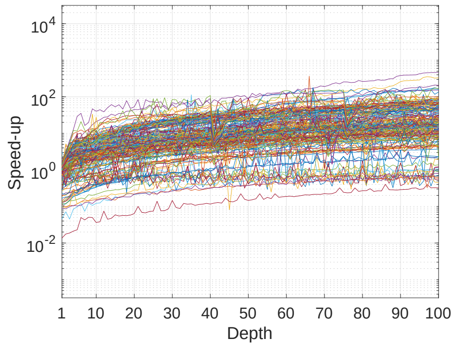
Several commercial and open-source modelling and simulation environments support FMI 2.0 (more than 100, see [27] for a full list).

Among commercial environments, we note Matlab/Simulink [155]: a widely-used and well-known platform for model-based design. It also offers its own API for advancing forward and backward the simulation by rolling back simulator states (thanks to a save/restore functionality). However, being a commercial software, Simulink supports only export of FMUs for CS and does not implement get/set functionality of FMI 2.0.

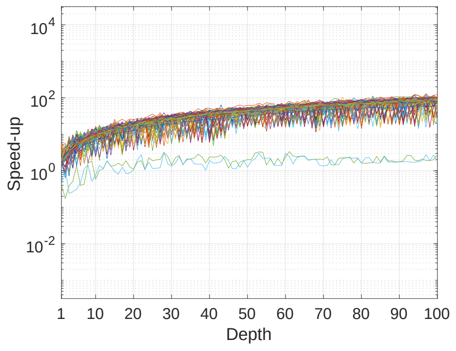
The FMI standard is supported also by some major Modelica-based commercial



(a)



(b)



(c)

Figure 5.11. Speed-up achieved for each FMU within MSL (a), STS (b) and BMD (c) when the branching factor is equal to 10 and the depth takes values within $[1, 100]$.

Tool	Modelica Support	Get/Set FMU state functionality
DACOSSIM	-	-
FMI4J	-	-
QTronic FMUSDK	-	-
JModelica	•	-
OpenModelica	•	-
PythonFMU	-	-
Simulix	-	-
Reference FMUs	-	ME & CS
Our extended JModelica	•	ME

Table 5.1. Open source modelling and simulation environments supporting FMI 2.0.

modelling and simulation environments such as:

1. Dymola 2020 [54], which, to the best of our knowledge, is the only Modelica-based modelling and simulation platform *fully* implementing FMI 2.0 standard. Hence, it generates FMUs for CS and also for ME having get/set state functionality implemented.

2. SystemModeler 12 [221], which allows users to generate their models as FMUs for both CS and ME. However, no FMU get/set state functionality is implemented.

3. SimulationX 4.1 [69], which also provides an implementation to get/set state functionality. However such a functionality is enabled only within FMUs for CS.

As motivated in Section 5.1, we focus on open-source modelling and simulation environments.

In Table 5.1 we compared several open-source platforms which support FMI 2.0. We split these platforms in the following groups:

1. The first group consists of all modelling and simulation environments supporting Modelica. Among those, we note OpenModelica [174] developed by the Open Source Modelica Consortium. OpenModelica translates Modelica models into C code, which, in turn, is compiled in a black-box executable format. Only upon user request it is also possible to wrap such a format into an FMU. Conversely, JModelica is FMI-based. Indeed, it directly generates

FMUs from the input Modelica code. Hence, FMUs are first-class citizens. This is the main reason why we focus on JModelica. However, we would like to clarify that, with a little effort, our proposed methodology can be quickly adapted to the OpenModelica as well.

2. The second group consists of all modelling and simulation environments which support FMI 2.0 but are not based on Modelica. Among those, *Reference FMUs* [159] implements get and set of FMU states. However, besides the fact that it is not Modelica-based, this software is just a small tool used for debugging FMUs.

A work close to ours is the NANDRAND simulation platform [171] which provides and simulates ready-to-use models to measure energy performance of physical buildings. Such models can be exported as FMUs compliant to FMI 2.0 and support get/set state functionality. It is worth noting that NANDRAND imports external FMUs (*e.g.*, generated by a Modelica-based environment) and couples them to NANDRAND models in a CS setting. However, as the authors specify, such a use-case works if and only if the imported FMUs support state saving and restoring.

Another tool related to our work is the FMI++ library [219] which is useful to simulate FMUs for ME. One of the features implemented by this library is a wrapper class, called `RollbackFMU`. Such a class stores, at each integration step, the current state of the input FMU in order to enable a rollback functionality. The functions used to save and restore the state are `fmi2GetContinuousState` and `fmi2SetContinuousState`, respectively. However, as FMI specifications clarify, such functions are not intended to set and get the complete FMU state but just values of continuous variables of the given FMU. As Section 5.2.2 describes in details, this is not enough. The state of an FMU has a complex structure where values of continuous model variables are just a small fraction of it.

5.5 Conclusions

We have presented an extended version of JModelica that implements FMI 2.0 methods to save and restore complete states of FMUs for ME. We have conducted an in-depth evaluation of our implementation on FMUs generated from widely established model libraries, namely MSL, STS and BMD. We have shown the correctness of our proposed implementation and assessed its performance on a demanding application such as simulation-based V&V approaches that drive the input FMU in the space of all simulation scenarios. In doing so, we computed the speed-up of a visit that drives FMUs generated by our extended JModelica implementation with respect to a visit that drives FMUs generated by stand-alone JModelica. We have shown even for a tree with a low branching factor, namely 5, and a low depth, namely 50, the speed-up achieved is, on average, 11.75, 14.06 and 39.84, among all FMUs within MSL, STS and BMD, respectively. Also, the achieved speed-up increases much more for deeper trees. Future directions can be to apply our save-and-restore implementation to FMUs for CS as well as to port our implementation to the OpenModelica simulation environment. Moreover, as the 3.0 version of FMI standard is planned to be released, it could be worth investigating how to adapt and extend our implementation to the new set of envisioned features (*e.g.*, clocks and hybrid co-simulation to handle event-driven dynamics and a new type of model exchange format, *i.e.*, Scheduled Execution, SE).

Results in this chapter have been presented in [195]. Software developed in this chapter is available at the following url: https://bitbucket.org/mclab/jmodelica.org.

Chapter 6

Conclusions

In an In Silico Clinical Trial (ISCT) a Virtual Patient (VP) (*i.e.*, a model of patient physiology) is coupled with a model of the biomedical product under assessment in a closed-loop model. The structure of such a system is similar to many Cyber-Physical System (CPS) models where a model of the physical system is controlled by a software. Due to the complexity of CPSs, simulation-based approaches are typically employed to evaluate system requirements (*e.g.*, safety properties) under *all* possible scenarios (*i.e.*, Verification and Validation, V&V, activity). In the same way, in an ISCT setting, safety and efficacy assessment can be achieved by simulating the effects of the biomedical product on all VPs. In this thesis, leveraging on this similarity, we investigate simulation-based approaches based on artificial intelligence and model checking to support ISCT.

In particular, one of the major obstacles currently hindering exploitation of ISCT is the lack of availability of populations of VPs, which must be representative of the entire spectrum of possible physiological characteristics or drug reactions entailed by the model (completeness). This lack is due to the complexity of patient physiology models at hand that often are also non-identifiable.

In Chapter 3 we have presented methods and software to compute a population of VPs for a given quantitative and non-identifiable model of the human physiology (plus drugs PK/PD). Our computed population yield model evolutions *distinguishable* from each other (different phenotypes), *representative* of the whole spectrum of phenotypes entailed by the model, and properly *stratified*, *i.e.*, organised in levels, each one showing the entire spectrum of phenotypes from different perspectives (*e.g.*, at different levels of abstraction). To this end, our methodology runs a Statistical Model Checking (SMC)–based global search on the space of the input model parameter values that, by exploiting suitable biological and medical knowledge elicited from experts and structural knowledge of the model, intelligently samples model parameter values and recognises physiologically meaningful behaviours (VPs) and different phenotypes. Also, at *any time*, our algorithm continuously provides an upper bound to the probability that further computation will discover new phenotypes (correct with a user-defined confidence level).

Having a complete population of VPs enables personalised medicine, where individualised pharmacological treatments are optimised for specific patients and can be designed before being actually administered. In Chapter 4 we have designed an ISCT to compute individualised pharmacological treatments. Our ISCT is based on an intelligent search on a patient digital twin which we defined by exploiting a quantitative model of the physiology and drugs Pharmacokinetics/Pharmacodynamics (PK/PD), a complete population of VPs and clinical measurements of human patients. Our search algorithm is backtracking-based and intelligently explores the space of possible treatments (sequence of drug administrations) by driving a simulator of the input patient digital twin to seek the lightest (in terms of overall amount of employed drug doses) treatment that is effective for that patient. The possibility to *individualise* a pharmacological treatment that has been designed for the average patient shows the potential of artificial intelligence for model-based personalised medicine.

In Chapter 5 we have proposed an open-source implementation of save and restore Functional Mock-up Interface (FMI) 2.0 functionalities that are currently available only in commercial modelling and simulation environments. Those functionalities are useful to increase efficiency of V&V activity for Functional Mock-up Unit (FMU)-based CPSs. In fact, V&V typically requires exploring different simulation scenarios (*i.e.*, different sequences of exogenous inputs to the CPS under verification). Also, this is true in the setting of ISCT (as the one we have designed in Chapter 4) where different sequences of drug administrations (scenarios) have to be simulated. In both settings, many such scenarios have a shared prefixes. Thanks to the availability of those FMI 2.0 save and restore functionalities, a notable amount of simulation time is saved by simulating such a shared prefix only once.

Bibliography

[1] H. Abbas, G. Fainekos, S. Sankaranarayanan, F. Ivančić, and A. Gupta. Probabilistic temporal logic falsification of cyber-physical systems. *ACM Transactions on Embedded Computing Systems*, 12(2s):95:1–95:30, 2013.

[2] U.G. Abdulla and R. Poteau. Identification of parameters in systems biology. *Math. Biosci.*, 305:133–145, 2018.

[3] W. Abou-Jaoudé, P.T. Monteiro, A. Naldi, M. Grandclaudon, V. Soumelis, C. Chaouiya, and D. Thieffry. Model checking to assess t-helper cell plasticity. *Frontiers in Bionengineering and Biotechnology*, 2:86, 2015.

[4] M. Aftab, C. Chen, C.K. Chau, and T. Rahwan. Automatic hvac control with real-time occupancy recognition and simulation-guided model predictive control in low-cost embedded system. *Energy and Buildings*, 154:141–156, 2017.

[5] G. Agha and K. Palmskog. A survey of statistical model checking. *ACM Transactions on Modeling and Computer Simulation*, 28(1):6:1–6:39, 2018.

[6] G. Agha and K. Palmskog. A survey of statistical model checking. *ACM TOMACS*, 28(1):6, 2018.

[7] Roger K Alexander. Stability of runge–kutta methods for stiff ordinary differential equations. *SIAM journal on numerical analysis*, 31(4):1147–1168, 1994.

[8] V. Alimguzhin, T. Mancini, A. Massini, S. Sinisi, and E. Tronci. In silico clinical trials through AI and statistical model checking. In *1st Workshop on Artificial Intelligence and Formal Verification, Logic, Automata, and Synthesis (OVERLAY 2019)*, volume 2509 of *CEUR Workshop Proceedings*, pages 17–22. CEUR-WS.org, 2019.

[9] V. Alimguzhin, F. Mari, I. Melatti, I. Salvo, and E. Tronci. Linearizing discrete-time hybrid systems. *IEEE Transactions on Automatic Control*, 62(10):5357–5364, 2017.

[10] R.J. Allen, T.R. Rieger, and C.J. Musante. Efficient generation and selection of virtual populations in quantitative systems pharmacology models. *CPT: Pharmacometrics & Systems Pharmacology*, 5(3):140–146, 2016.

[11] R. Alur. Formal verification of hybrid systems. In *Proceedings of 11th International Conference on Embedded Software (EMSOFT 2011)*, pages 273–278. ACM, 2011.

[12] Christian Andersson, Johan Åkesson, and Claus Führer. *Pyfmi: A python package for simulation of coupled dynamic models with the functional mock-up interface.* Centre for Mathematical Sciences, Lund University Lund, 2016.

[13] Ansys Inc. ADVANTAGE Excellence in Engineering Simulation: Best of Healthcare Special Issue, 2018.

[14] G. Arellano, J. Argil, E. Azpeitia, M. Benítez, M. Carrillo, P. Góngora, D.A. Rosenblueth, and E.R. Alvarez-Buylla. Antelope: A hybrid-logic model checker for branching-time boolean grn analysis. *BMC Bioinformatics*, 12(1):490, 2011.

[15] David Arney, Miroslav Pajic, Julian M Goldman, Insup Lee, Rahul Mangharam, and Oleg Sokolsky. Toward patient safety in closed-loop medical device systems. In *Proceedings of the 1st ACM/IEEE International Conference on Cyber-Physical Systems*, pages 139–148, 2010.

[16] Y. Asai, T. Abe, H. Oka, M. Okita, K. Hagihara, S. Ghosh, Y. Matsuoka, Y. Kurachi, T. Nomura, and H. Kitano. A versatile platform for multilevel modeling of physiological systems: SBML-PHML hybrid modeling and simulation. *Advances Biomedical Engineering*, 3:50–58, 2014.

[17] F. Ascione, N. Bianco, C. De Stasio, G. M. Mauro, and G. P. Vanoli. Simulation-based model predictive control by the multi-objective optimization of building energy performance and thermal comfort. *Energy and Buildings*, 111:131–144, 2016.

[18] Avicenna Project. *In silico* clinical trials: How computer simulation will transform the biomedical industry. http://avicenna-isct.org/wp-content/uploads/2016/01/AvicennaRoadmapPDF-27-01-16.pdf, 2016.

[19] M. Bächler, D. Menshykau, Ch. De Geyter, and D. Iber. Species-specific differences in follicular antral sizes result from diffusion-based limitations on the thickness of the granulosa cell layer. *Molecular Human Reproduction*, 20(3):208–221, 2014.

[20] J. Barnat, L. Brim, D. Šafránek, and M. Vejnár. Parameter scanning by parallel model checking with applications in systems biology. In *Proceedings of 9th International Workshop on Parallel and Distributed Methods in Verification and 2nd International Workshop on High Performance Computational Systems Biology (PDMC-HIBI 2010)*, pages 95–104. IEEE, 2010.

[21] E. Bartocci and P. Lió. Computational modeling, formal analysis, and tools for systems biology. *PLoS Computational Biology*, 12(1):e1004591, 2016.

[22] G. Batt, D. Ropers, H. De Jong, J. Geiselmann, R. Mateescu, M. Page, and D. Schneider. Validation of qualitative models of genetic regulatory networks by model checking: analysis of the nutritional stress response in escherichia coli. *Bioinformatics*, 21(suppl_1):i19–i28, 2005.

[23] A. Biere, M. Heule, and H. van Maaren, editors. *Handbook of Satisfiability*, volume 185. IOS Press, 2009.

[24] Armin Biere, Cyrille Artho, and Viktor Schuppan. Liveness checking as safety checking. *Electronic Notes in Theoretical Computer Science*, 66(2):160–177, 2002.

[25] Armin Biere, Alessandro Cimatti, Edmund M Clarke, Ofer Strichman, and Yunshan Zhu. Bounded model checking. 2003.

[26] T. Blochwitz, M. Otter, J. Akesson, M. Arnold, C. Clauss, H. Elmqvist, M. Friedrich, A. Junghanns, J. Mauss, D. Neumerkel, et al. Functional mockup interface 2.0: The standard for tool independent exchange of simulation models. In *Proceedings of the 9th International Modelica Conference*, pages 173–184. Linköping University Electronic Press, 2012.

[27] Torsten Blockwitz, Martin Otter, Johan Akesson, Martin Arnold, Christoph Clauss, Hilding Elmqvist, Markus Friedrich, Andreas Junghanns, Jakob Mauss, Dietmar Neumerkel, et al. Functional mockup interface 2.0: The standard for tool independent exchange of simulation models. In *Proceedings of the 9th International Modelica Conference*, 2012.

[28] P: Bloomingdale, J. Niu, and D.E. Mager. Boolean network modeling in systems pharmacology. *Journal of Pharmacokinetics and Pharmacodynamics*, 45(1):159–180, 2018.

[29] J. Bogdoll, A. Hartmanns, and H. Hermanns. Simulation and statistical model checking for modestly nondeterministic models. In *Proceedings of Measurement Modelling and Evaluation of Computing Systems and Dependability and Fault Tolerance (MMB&DFT 2012)*, Lecture Notes in Computer Science, pages 249–252. Springer, 2012.

[30] S. Bogomolov, D. Magazzeni, S. Minopoli, and M. Wehrle. PDDL+ planning with hybrid automata: Foundations of translating must behavior. In *Proceedings of 25th International Conference on Automated Planning and Scheduling (ICAPS 2015)*, pages 42–46. AAAI, 2015.

[31] S. Bogomolov, D. Magazzeni, A. Podelski, and M. Wehrle. Planning as model checking in hybrid domains. In *Proceedings of 28th National Conference on Artificial Intelligence (AAAI 2014)*, pages 2228–2234. AAAI, 2014.

[32] L. Bordeaux, M. Cadoli, and T. Mancini. A unifying framework for structural properties of CSPs: Definitions, complexity, tractability. *Journal of Artificial Intelligence Research*, 32:607–629, 2008.

[33] L. Bordeaux, M. Cadoli, and T. Mancini. Generalizing consistency and other constraint properties to quantified constraints. *ACM Transactions on Computational Logic*, 10(3):17:1–17:25, 2009.

[34] Hala Borno, Adam Siegel, and Charles Ryan. The problem of representativeness of clinical trial participants: understanding the role of hidden costs, 2016.

[35] D. Bryce, S. Gao, D.J. Musliner, and R.P. Goldman. Smt-based nonlinear PDDL+ planning. In *Proceedings of 29th National Conference on Artificial Intelligence (AAAI 2015)*, pages 3247–3253. AAAI, 2015.

[36] A. Calabrese, T. Mancini, A. Massini, S. Sinisi, and E. Tronci. Generating T1DM virtual patients for in silico clinical trials via AI-guided statistical model checking. In *26th RCRA International Workshop on Experimental Evaluation of Algorithms for Solving Problems with Combinatorial Explosion (RCRA 2019)*, volume 2538 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2019.

[37] F. Casella. Simulation of large-scale models in modelica: State of the art and future perspectives. In *Proceedings of the 11th International Modelica Conference*, pages 459–468, 2015.

[38] R. Caspi, R. Billington, L. Ferrer, H. Foerster, C.A. Fulcher, I.M. Keseler, A. Kothari, M. Krummenacker, M. Latendresse, L.A. Mueller, Q. Ong, S. Paley, P. Subhraveti, D.S. Weaver, and P.D. Karp. The metacyc database of metabolic pathways and enzymes and the biocyc collection of pathway/genome databases. *Nucleic Acids Res.*, 44(D1):D471–D480, 2016.

[39] M. Čech, J. Königsmarková, J. Reitinger, and P. Balda. Novel tools for model-based control system design based on fmi/fmu standard with application in

energetics. In *2017 21st International Conference on Process Control (PC)*, pages 416–421. IEEE, 2017.

[40] N. Chabrier and F. Fages. Symbolic model checking of biochemical networks. In *Proceedings of International Conference on Computational Methods in Systems Biology*, volume 2602 of *Lecture Notes in Computer Science*, pages 149–162. Springer, 2003.

[41] Kai-hui Chang, Valeria Bertacco, and Igor L Markov. Simulation-based bug trace minimization with bmc-based refinement. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 26(1):152–165, 2006.

[42] O.-T. Chis, J.R. Banga, and E. Balsa-Canto. Structural identifiability of systems biology models: A critical comparison of methods. *PLoS ONE*, 6(11), 2011.

[43] Alessandro Cimatti, Edmund Clarke, Fausto Giunchiglia, and Marco Roveri. Nusmv: A new symbolic model verifier. In *International conference on computer aided verification*, pages 495–499. Springer, 1999.

[44] Gordon J. Clapworthy, Peter Kohl, Hans Gregerson, S. R. Thomas, Marco Viceconti, D. R. Hose, D. Pinney, John Fenner, K. McCormack, P. Lawford, S. Van Sint Jan, S. Waters, and P. Coveney. Digital human modelling: A global vision and a european perspective. In Vincent G. Duffy, editor, *Digital Human Modeling*, pages 549–558, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[45] E.M. Clarke, A. Donzé, and A. Legay. On simulation-based probabilistic model checking of mixed-analog circuits. *Formal Methods in System Design*, 36(2):97–113, 2010.

[46] E.M. Clarke and P. Zuliani. Statistical model checking for cyber-physical systems. In *Proceedings of 9th International Symposium on Automated Technology for Verification and Analysis (ATVA 2011)*, volume 11, pages 1–12. Springer, 2011.

[47] Jay S Cohen. Ways to minimize adverse drug reactions: individualized doses and common sense are key. *Postgraduate medicine*, 106(3):163–172, 1999.

[48] A.J. Coles and A.I. Coles. PDDL+ planning with events and linear processes. In *Proceedings of 24th International Conference on Automated Planning and Scheduling (ICAPS 2014)*. AAAI, 2014.

[49] Patricio Colmegna, Ke Wang, Jose Garcia-Tirado, and Marc D. Breton. Mapping data to virtual patients in type 1 diabetes. *Control Engineering Practice*, 103:104605, 2020.

[50] R. Coutinho, B. Fernandez, R. Lima, and A. Meyroneinc. Discrete time piecewise affine models of genetic regulatory networks. *Journal of Mathematical Biology*, 52(4):524–570, 2006.

[51] L.G.E. Cox, S. Loerakker, M.C.M. Rutten, B.A.J.M. De Mol, and F.N. Van De Vosse. A mathematical model to evaluate control strategies for mechanical circulatory support. *Artificial Organs*, 33(8):593–603, 2009.

[52] F. Cremona, M. Lohstroh, D. Broman, M. Di Natale, E.A. Lee, and S. Tripakis. Step revision in hybrid co-simulation with fmi. In *2016 ACM/IEEE International Conference on Formal Methods and Models for System Design (MEMOCODE)*, pages 173–183. IEEE, 2016.

[53] C. Dalla Man, F. Micheletto, D. Lv, M. Breton, B. Kovatchev, and C. Cobelli. The UVA/Padova type 1 diabetes simulator: New features. *Journal of Diabetes Science and Technology*, 8:26–34, 2014.

[54] Dassault Systemes. DYMOLA Systems Engineering. Multi-Engineering Modeling and Simulation based on Modelica and FMI. https://dymola.com.

[55] A. David, D. Du, K.G. Larsen, A. Legay, M. Mikučionis, D.B. Poulsen, and S. Sedwards. Statistical model checking for stochastic hybrid systems. *Electronic Proceedings in Theoretical Computer Science*, 92:122–136, 2012.

[56] G. Della Penna, B. Intrigila, D. Magazzeni, I. Melatti, and E. Tronci. Cgmurphi: Automatic synthesis of numerical controllers for nonlinear hybrid systems. *European Journal of Control*, 19(1):14–36, 2013.

[57] G. Della Penna, B. Intrigila, D. Magazzeni, and F. Mercorio. Planning for autonomous planetary vehicles. In *Proceedings of 6th International Conference on Autonomic and Autonomous Systems (ICAS 2010)*, pages 131–136. IEEE, 2010.

[58] G. Della Penna, B. Intrigila, I. Melatti, E. Tronci, and M. Venturini Zilli. Exploiting transition locality in automatic verification of finite state concurrent systems. *International Journal on Software Tools for Technology Transfer*, 6(4):320–341, 2004.

[59] G. Della Penna, D. Magazzeni, F. Mercorio, and B. Intrigila. Upmurphi: A tool for universal planning on PDDL+ problems. In *Proceedings of 19th Inter-*

national Conference on Automated Planning and Scheduling (ICAPS 2009). AAAI, 2009.

[60] L. S. Dias, R. C. Pattison, C. Tsay, M. Baldea, and M. G. Ierapetritou. A simulation-based optimization framework for integrating scheduling and model predictive control, and its application to air separation units. *Computers & Chemical Engineering*, 113:139–151, 2018.

[61] T. Dierkes, S. Röblitz, M. Wade, and P. Deuflhard. Parameter identification in large kinetic networks with BioPARKIN. *CoRR*, abs/1303.4928, 2013.

[62] David L Dill, Andreas J Drexler, Alan J Hu, and C Han Yang. Protocol verification as a hardware design aid. In *ICCD*, volume 92, pages 522–525. Citeseer, 1992.

[63] Joseph A DiMasi, Henry G Grabowski, and Ronald W Hansen. Innovation in the pharmaceutical industry: new estimates of r&d costs. *Journal of health economics*, 47:20–33, 2016.

[64] Joseph A DiMasi, Ronald W Hansen, and Henry G Grabowski. The price of innovation: new estimates of drug development costs. *Journal of health economics*, 22(2):151–185, 2003.

[65] D. Ding, Q.L. Han, Z. Wang, and X. Ge. A survey on model-based distributed control and filtering for industrial cyber-physical systems. *IEEE Transactions on Industrial Informatics*, 15(5):2483–2499, 2019.

[66] P.D. Docherty, J.G. Chase, and T. David. Characterisation of the iterative integral parameter identification method. *Med. Biol. Eng. Comput.*, 50(2):127–134, 2012.

[67] P.S. Duggirala, S. Mitra, M. Viswanathan, and M. Potok. C2E2: A verification tool for stateflow models. In *Proceedings of 21st International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2015)*, volume 9035 of *Lecture Notes in Computer Science*, pages 68–82. Springer, 2015.

[68] E. Durling, E. Palmkvist, and M. Henningsson. Fmi and ip protection of models: a survey of use cases and support in the standard. In *Proceedings of the 12th International Modelica Conference, Prague, Czech Republic, May 15-17, 2017*, number 132, pages 329–335. Linköping University Electronic Press, 2017.

[69] ESI Group. Simulation Software SimulationX. https://www.simulationx.com.

[70] European Medicines Agency. Reporting of physiologically based pharmacokinetic (PBPK) modelling and simulation, 2019. EMA/CHMP/458101/2016.

[71] K. Eykholt, I. Evtimov, E. Fernandes, B. Li, A. Rahmati, c. Xiao, A. Prakash, T. Kohno, and D. Song. Robust physical-world attacks on deep learning visual classification. In *Proceedings of 2018 IEEE Conference on Computer Vision and Pattern Recognition (IEEE CVPR 2018)*, pages 1625–1634. IEEE, 2018.

[72] A. Fabregat, S. Jupe, L. Matthews, K. Sidiropoulos, M. Gillespie, P. Garapati, R. Haw, B. Jassal, F. Korninger, B. May, M. Milacic, C.D. Roca, K. Rothfels, C. Sevilla, V. Shamovsky, S. Shorser, T. Varusai, G. Viteri, J. Weiser, G. Wu, L. Stein, H. Hermjakob, and P. D'Eustachio. The reactome pathway knowledgebase. *Nucleic Acids Res.*, 46(D1):D649–D655, 2018.

[73] F. Falcini and G. Lami. Challenges in certification of autonomous driving systems. In *2017 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*, pages 286–293, 2017.

[74] A. Falcone and A. Garro. Distributed co-simulation of complex engineered systems by combining the high level architecture and functional mock-up interface. *Simulation Modelling Practice and Theory*, 97:101967, 2019.

[75] Chuchu Fan and Sayan Mitra. Bounded verification with on-the-fly discrepancy computation. In *International Symposium on Automated Technology for Verification and Analysis*, pages 446–463. Springer, 2015.

[76] Yu Feng, Xiaole Chen, and Mingshi Yang. An in silico investigation of a lobe-specific targeted pulmonary drug delivery method. In *Frontiers in Biomedical Devices*, volume 40789, page V001T08A011. American Society of Mechanical Engineers, 2018.

[77] A. Fogliata, F. Belosi, A. Clivio, P. Navarria, G. Nicolini, M. Scorsetti, E. Vanetti, and L. Cozzi. On the pre-clinical validation of a commercial model-based optimisation engine: Application to volumetric modulated arc therapy for patients with lung or prostate cancer. *Radiotherapy & Oncology*, 113(3):385–391, 2014.

[78] M. Fox and D. Long. Modelling mixed discrete-continuous domains for planning. *Journal of Artificial Intelligence Research*, 27:235–297, 2006.

[79] G. Frances, M. Ramírez Jávega, N. Lipovetzky, and H. Geffner. Purely declarative action descriptions are overrated: Classical planning with simulators. In *Proceedings of 26th International Joint Conference on Artificial Intelligence (IJCAI 2017)*, pages 4294–4301, 2017.

[80] Peter Fritzson. *Introduction to modeling and simulation of technical and physical systems with Modelica.* John Wiley & Sons, 2011.

[81] J. Garcia-Tirado, C. Zuluaga-Bedoya, and M.D. Breton. Identifiability analysis of three control-oriented models for use in artificial pancreas systems. *JDST*, 12(5):937–952, 2018.

[82] F.W. Glover and G.A. Kochenberger, editors. *Handbook of Metaheuristics*, volume 57. Springer, 2006.

[83] C. Gomes, B. Meyers, J. Denil, C. Thule, K. Lausdahl, H. Vangheluwe, and P. De Meulenaere. Semantic adaptation for fmi co-simulation with hierarchical simulators. *Simulation*, 95(3):241–269, 2019.

[84] C. Gomes, C. Thule, D. Broman, P. G. Larsen, and H. Vangheluwe. Co-simulation: A survey. *ACM Comput. Surv.*, 51(3):49:1–49:33, 2018.

[85] H. Gong, P. Zuliani, Q. Wang, and E.M. Clarke. Formal analysis for logical models of pancreatic cancer. In *Proceedings of 50th IEEE Conference on Decision and Control (CDC 2011)*, pages 4855–4860. IEEE, 2011.

[86] G. Gottlob, G. Greco, and T. Mancini. Conditional constraint satisfaction: Logical foundations and complexity. In *Proceedings of 20th International Joint Conference on Artificial Intelligence (IJCAI 2007)*, pages 88–93, 2007.

[87] R. Grosu and S.A. Smolka. Monte Carlo model checking. In *Proceedings of 11th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2005)*, volume 3440 of *Lecture Notes in Computer Science*, pages 271–286. Springer, 2005.

[88] W. Guo, G. Yang, W. Wu, L. He, and M. Sun. A parallel attractor finding algorithm based on boolean satisfiability for genetic regulatory networks. *PloS One*, 9(4):e94258, 2014.

[89] S. A. Haque, S. M. Aziz, and M. Rahman. Review of cyber-physical system in healthcare. *international journal of distributed sensor networks*, 10(4):217415, 2014.

[90] B.P. Hayes, I. Melatti, T. Mancini, M. Prodanovic, and E. Tronci. Residential demand management using individualised demand aware price policies. *IEEE Transactions on Smart Grid*, 8(3), 2017.

[91] J. Heath, M.Z. Kwiatkowska, G. Norman, D. Parker, and O. Tymchyshyn. Probabilistic model checking of complex biological pathways. *Theoretical Computer Science*, 391(3):239–257, 2008.

[92] M.P. Hengartner, T.H.C. Kruger, K. Geraedts, E. Tronci, T. Mancini, F. Ille, M. Egli, S. Roeblitz, R. Ehrig, L. Saleh, K. Spanaus, C. Schippert, Y. Zhang, and B. Leeners. Negative affect is unrelated to fluctuations in hormone levels across the menstrual cycle: Evidence from a multisite observational study across two successive cycles. *Journal of Psychosomatic Research*, 99:21–27, 2017.

[93] P. Herrero, P. Georgiou, N. Oliver, M. Reddy, D. Johnston, and C. Toumazou. A composite model of glucagon–glucose dynamics for in silico testing of bihormonal glucose controllers. *Journal of Diabetes Science and Technology*, 7:941–951, 2013.

[94] R. Hester, A. Brown, L. Husband, R. Iliescu, W.A. Pruett, R.L. Summers, and T. Coleman. Hummod: a modeling environment for the simulation of integrative human physiology. *Frontiers in physiology*, 2:12, 2011.

[95] Catherine F. Higham and Dirk Husmeier. A bayesian approach for parameter estimation in the extended clock gene circuit of arabidopsis thaliana. *BMC Bioinformatics*, 14(S-10):S3, 2013.

[96] Alan C Hindmarsh, Peter N Brown, Keith E Grant, Steven L Lee, Radu Serban, Dan E Shumaker, and Carol S Woodward. Sundials: Suite of nonlinear and differential/algebraic equation solvers. *ACM Transactions on Mathematical Software (TOMS)*, 31(3):363–396, 2005.

[97] Gerard J. Holzmann. The model checker spin. *IEEE Transactions on software engineering*, 23(5):279–295, 1997.

[98] G.J. Holzmann. Parallelizing the SPIN model checker. In *Proceedings of 19th International SPIN Symposium on Model Checking of Software (SPIN 2012)*, volume 7385 of *Lecture Notes in Computer Science*, pages 155–171. Springer, 2012.

[99] 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 Novère, 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, and J. Wang. The Systems Biology Markup language (SBML): a medium for representation and exchange of biochemical network models. *Bioinformatics*, 19(4):524–531, 2003.

[100] PJ Hunter. The iups physiome project: a framework for computational physiology. *Progress in biophysics and molecular biology*, 85(2-3):551–569, 2004.

[101] K.C. Iarosz, F.S. Borges, A.M. Batista, M.S. Baptista, R.A.N. Siqueira, R.L. Viana, and S.R. Lopes. Mathematical model of brain tumour with glia–neuron interactions and chemotherapy treatment. *Journal of Theoretical Biology*, 368:113–121, 2015.

[102] Itziar Irurzun-Arana, J. M. Pastor, I. F. Trocóniz, and J.D. Gómez-Mantilla. Advanced boolean modeling of biological networks applied to systems pharmacology. *Bioinformatics*, 33(7):1040–1048, 2017.

[103] S. Ito, T. Ichinose, M. Shimakawa, N. Izumi, S. Hagihara, and N. Yonezaki. Qualitative analysis of gene regulatory networks by temporal logic. *Theor. Comput. Sci.*, 594:151–179, 2015.

[104] P.R. Jackson, J. Juliano, A. Hawkins-Daarud, R.C. Rockne, and K.R. Swanson. Patient-specific mathematical neuro-oncology: Using a simple proliferation and invasion tumor model to inform clinical practice. *Bulletin of Mathematical Biology*, 77(5):846–856, 2015.

[105] P.M. Jeena, W.R. Bishai, J.G. Pasipanodya, and T. Gumbo. *In Silico* children and the glass mouse model: clinical trial simulations to identify and individualize optimal isoniazid doses in children with tuberculosis. *Antimicrobial Agents and Chemotherapy*, 55(2):539–545, 2011.

[106] Eric Jenn, Alexandre Albore, Franck Mamalet, Grégory Flandin, Christophe Gabreau, Hervé Delseny, Adrien Gauffriau, Hugues Bonnin, Lucian Alecu, Jérémy Pirard, et al. Identifying challenges to the certification of machine learning for safety critical systems. In *10th European Congress on Embedded Real Time Systems (ERTS)*, 2020.

[107] P.G. Jensen, K.G. Larsen, A. Legay, and U. Nyman. Integrating tools: Co-simulation in uppaal using fmi-fmu. In *2017 22nd International Conference on Engineering of Complex Computer Systems (ICECCS)*, pages 11–19. IEEE, 2017.

[108] Zhihao Jiang, Miroslav Pajic, Rajeev Alur, and Rahul Mangharam. Closed-loop verification of medical devices with model abstraction and refinement. *International Journal on Software Tools for Technology Transfer*, 16(2):191–213, 2014.

[109] K. Kalajdzic, C. Jégourel, A. Lukina, E. Bartocci, A. Legay, S.A. Smolka, and R. Grosu. Feedback control for statistical model checking of cyber-physical

systems. In *Proceedings of 7th International Symposium on Leveraging Applications of Formal Methods, Verification and Validation: Foundational Techniques (ISoLA 2016)*, volume 9952 of *Lecture Notes in Computer Science*, pages 46–61. Springer, 2016.

[110] S.S. Kanderian, S. Weinzimer, G. Voskanyan, and G.M. Steil. Identification of intraday metabolic profiles during closed-loop glucose control in individuals with type 1 diabetes. *Journal of Diabetes Science and Technology*, 3:1047–1057, 2009.

[111] M. Kanehisa, M. Furumichi, M. Tanabe, Y. Sato, and K. Morishima. Kegg: New perspectives on genomes, pathways, diseases and drugs. *Nucleic Acids Research*, 45(D1):D353, 2017.

[112] J. Kapinski, J. V. Deshmukh, X. Jin, H. Ito, and K. Butts. Simulation-based approaches for verification of embedded control systems: An overview of traditional and advanced modeling, testing, and verification techniques. *IEEE Control Systems Magazine*, 36(6):45–64, 2016.

[113] Akitoshi Kawamura. Lipschitz continuous ordinary differential equations are polynomial-space complete. *Computational Complexity*, 19(2):305–332, 2010.

[114] Kyoung Ae Kim, Sabrina L. Spencer, John G. Albeck, John M. Burke, Peter K. Sorger, Suzanne Gaudet, and Do Hyun Kim. Systematic calibration of a cell signaling network model. *BMC Bioinformatics*, 11:202, 2010.

[115] Jirí Kofránek, Filip Ježek, and Marek Mateják. Modelica language–a promising tool for publishing and sharing biomedical models. In *Proceedings of the 1st American Modelica Conference*, pages 9–10, 2018.

[116] B.P. Kovatchev, M. Breton, C. Dalla Man, and C. Cobelli. In silico preclinical trials: A proof of concept in closed-loop control of type 1 diabetes. *Journal of Diabetes Science and Technology*, 3(1):44–55, 2009.

[117] Andrei Kramer, Ben Calderhead, and Nicole Radde. Hamiltonian monte carlo methods for efficient parameter estimation in steady state dynamical systems. *BMC Bioinformatics*, 15:253, 2014.

[118] M. Krauss, R. Burghaus, J. Lippert, M. Niemi, P. Neuvonen, A. Schuppert, S. Willmann, L. Kuepfer, and L. Görlitz. Using bayesian-pbpk modeling for assessment of inter-individual variability and subgroup stratification. *In Silico Pharmacology*, 1(1):6, 2013.

[119] Orna Kupferman, Nir Piterman, and Moshe Y Vardi. From liveness to promptness. *Formal Methods in System Design*, 34(2):83–103, 2009.

[120] N. Le Novère. Quantitative and logic modelling of molecular and gene networks. *Nature Reviews Genetics*, 16(3):146–158, 2015.

[121] N. Le Novère, B. Bornstein, A. Broicher, M. Courtot, M. Donizelli, H. Dharuri, L. Li, H. Sauro, M. Schilstra, B. Shapiro, J.L. Snoep, and M. Hucka. BioModels Database: A free, centralized database of curated, published, quantitative kinetic models of biochemical and cellular systems. *Nucleic Acids Research*, 34(suppl_1):D689–D691, 2006.

[122] Paola Lecca. *Model Identifiability*, pages 37–48. Springer International Publishing, Cham, 2020.

[123] E. A. Lee. Cyber physical systems: Design challenges. In *2008 11th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC)*, pages 363–369. IEEE, 2008.

[124] E.A. Lee and P. Varaiya. *Structure and Interpretation of Signals and Systems*. LeeVaraiya.org, 2nd edition, 2011.

[125] J. Lee, B. Bagheri, and H.A. Kao. A cyber-physical systems architecture for industry 4.0-based manufacturing systems. *Manufacturing letters*, 3:18–23, 2015.

[126] B. Leeners, T.H.C. Krüger, K. Geraedts, E. Tronci, T. Mancini, M. Egli, S. Röblitz, L. Saleh, K. Spanaus, C. Schippert, Y. Zhang, and F. Ille. Associations between natural physiological and supraphysiological estradiol levels and stress perception. *Frontiers in Psychology*, 10:1296, 2019.

[127] B. Leeners, T.H.C. Kruger, K. Geraedts, E. Tronci, T. Mancini, F. Ille, M. Egli, S. Roeblitz, L. Saleh, K. Spanaus, C. Schippert, Y. Zhang, and M.P. Hengartner. Lack of associations between female hormone levels and visuospatial working memory, divided attention and cognitive bias across two consecutive menstrual cycles. *Frontiers in Behavioral Neuroscience*, 11, 2017.

[128] H.X. Li and B.C. Williams. Generative planning for hybrid systems based on flow tubes. In *Proceedings of 18th International Conference on Automated Planning and Scheduling (ICAPS 2008)*, pages 206–213. AAAI, 2008.

[129] N. Lipovetzky and H. Geffner. Width and serialization of classical planning problems. In *Proceedings of 20th European Conference on Artificial Intelligence (ECAI 2012)*, pages 540–545. IOS Press, 2012.

[130] N. Lipovetzky, M. Ramirez, and H. Geffner. Classical planning with simulators: Results on the atari video games. In *Proceedings of 24th International*

Join Conference on Artificial Intelligence (IJCAI 2015), volume 15, pages 1610–1616, 2015.

[131] J. Lippert, R. Burghaus, A. Edginton, S. Frechen, M. Karlsson, A. Kovar, T. Lehr, P. Milligan, V. Nock, S. Ramusovic, M. Riggs, S. Schaller, J. Schlender, S. Schmidt, M. Sevestre, E. Sjögren, J. Solodenko, A. Staab, and D. Teutonico. Open systems pharmacology community–an open access, open source, open science approach to modeling and simulation in pharmaceutical sciences. *CPT: Pharmacometrics & Systems Pharmacology*, 8(12):878–882, 2019.

[132] Bing Liu, Soonho Kong, Sicun Gao, Paolo Zuliani, and Edmund M Clarke. Towards personalized prostate cancer therapy using delta-reachability analysis. In *Proceedings of the 18th International Conference on Hybrid Systems: Computation and Control*, pages 227–232, 2015.

[133] Y. Liu, Y. Peng, B. Wang, S. Yao, and Z. Liu. Review on cyber-physical systems. *IEEE/CAA Journal of Automatica Sinica*, 4(1):27–40, 2017.

[134] L. Ljung. *System identification: theory for the user*. Prentice-hall, 1987.

[135] Curtis Madsen, Fedor Shmarov, and Paolo Zuliani. Biopsy: an smt-based tool for guaranteed parameter set synthesis of biological models. In *International Conference on Computational Methods in Systems Biology*, pages 182–194. Springer, 2015.

[136] F. Maggioli, T. Mancini, and E. Tronci. SBML2Modelica: Integrating biochemical models within open-standard simulation ecosystems. *Bioinformatics*, 36(7):2165–2172, 2020.

[137] Oded Maler, Dejan Nickovic, and Amir Pnueli. Real time temporal logic: Past, present, future. In *International Conference on Formal Modeling and Analysis of Timed Systems*, pages 2–16. Springer, 2005.

[138] T. Mancini, M. Cadoli, D. Micaletto, and F. Patrizi. Evaluating ASP and commercial solvers on the CSPLib. *Constraints*, 13(4):407–436, 2008.

[139] T. Mancini, P. Flener, and J. Pearson. Combinatorial problem solving over relational databases: View synthesis through constraint-based local search. In *Proceedings of ACM Symposium on Applied Computing (SAC 2012)*, pages 80–87. ACM, 2012.

[140] T. Mancini, F. Mari, A. Massini, I. Melatti, F. Merli, and E. Tronci. System level formal verification via model checking driven simulation. In *Proceedings of 25th International Conference on Computer Aided Verification*

(CAV 2013), volume 8044 of *Lecture Notes in Computer Science*, pages 296–312. Springer, 2013.

[141] T. Mancini, F. Mari, A. Massini, I. Melatti, I. Salvo, S. Sinisi, E. Tronci, R. Ehrig, S. Röblitz, and B. Leeners. Computing personalised treatments through in silico clinical trials. A case study on downregulation in assisted reproduction. In *25th RCRA International Workshop on Experimental Evaluation of Algorithms for Solving Problems with Combinatorial Explosion (RCRA 2018)*, volume 2271 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2018.

[142] T. Mancini, F. Mari, A. Massini, I. Melatti, and E. Tronci. Anytime system level verification via random exhaustive hardware in the loop simulation. In *Proceedings of 17th Euromicro Conference on Digital System Design (DSD 2014)*, pages 236–245. IEEE, 2014.

[143] T. Mancini, F. Mari, A. Massini, I. Melatti, and E. Tronci. System level formal verification via distributed multi-core hardware in the loop simulation. In *Proceedings of 22nd Euromicro International Conference on Parallel, Distributed, and Network-Based Processing (PDP 2014)*, pages 734–742. IEEE, 2014.

[144] T. Mancini, F. Mari, A. Massini, I. Melatti, and E. Tronci. SyLVaaS: System level formal verification as a service. In *Proceedings of 23rd Euromicro International Conference on Parallel, Distributed, and Network-Based Processing (PDP 2015)*, pages 476–483. IEEE, 2015.

[145] T. Mancini, F. Mari, A. Massini, I. Melatti, and E. Tronci. Anytime system level verification via parallel random exhaustive hardware in the loop simulation. *Microprocessors and Microsystems*, 41:12–28, 2016.

[146] T. Mancini, F. Mari, A. Massini, I. Melatti, and E. Tronci. SyLVaaS: System level formal verification as a service. *Fundamenta Informaticae*, 1–2:101–132, 2016.

[147] T. Mancini, F. Mari, I. Melatti, I. Salvo, E. Tronci, J. Gruber, B. Hayes, M. Prodanovic, and L. Elmegaard. Demand-aware price policy synthesis and verification services for smart grids. In *Proceedings of 2014 IEEE International Conference on Smart Grid Communications (SmartGridComm 2014)*, pages 794–799. IEEE, 2014.

[148] T. Mancini, F. Mari, I. Melatti, I. Salvo, E. Tronci, J.K. Gruber, B. Hayes, and L. Elmegaard. Parallel statistical model checking for safety verification in smart grids. In *Proceedings of 2018 IEEE International Conference on Smart Grid Communications (SmartGridComm 2018)*. IEEE, 2018.

[149] T. Mancini, F. Mari, I. Melatti, I. Salvo, E. Tronci, J.K. Gruber, B. Hayes, M. Prodanovic, and L. Elmegaard. User flexibility aware price policy synthesis for smart grids. In *Proceedings of 18th Euromicro Conference on Digital System Design (DSD 2015)*, pages 478–485. IEEE, 2015.

[150] T. Mancini, E. Tronci, I. Salvo, F. Mari, A. Massini, and I. Melatti. Computing biological model parameters by parallel statistical model checking. In *Proceedings of 3rd International Conference on Bioinformatics and Biomedical Engineering (IWBBIO 2015)*, volume 9044 of *Lecture Notes in Computer Science*, pages 542–554. Springer, 2015.

[151] K. Margellos and J. Lygeros. A simulation based mpc technique for feedback linearizable systems with input constraints. In *49th IEEE Conference on Decision and Control (CDC)*, pages 7539–7544. IEEE, 2010.

[152] F. Mari, I. Melatti, I. Salvo, and E. Tronci. Model based synthesis of control software from system level formal specifications. *ACM Transactions on Software Engineering and Methodology*, 23(1):1–42, 2014.

[153] S. Martínez, S. Gerard, and J. Cabot. On the need for intellectual property protection in model-driven co-engineering processes. In *Enterprise, Business-Process and Information Systems Modeling*, pages 169–177. Springer, 2019.

[154] M. Mateják and J. Kofránek. Physiomodel – An integrative physiology in Modelica. In *Proceedings of 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC 2015)*, pages 1464–1467. IEEE, 2015.

[155] Mathworks. Simulink. https://mathworks.com.

[156] M. Mazo, A. Davitian, and P. Tabuada. PESSOA: A tool for embedded controller synthesis. In *Proceedings of 22nd International Conference on Computer Aided Verification (CAV 2010)*, volume 6174 of *Lecture Notes in Computer Science*, pages 566–569. Springer, 2010.

[157] Antoni Miró, Carlos Pozo, Gonzalo Guillén-Gosálbez, Jose A. Egea, and Laureano Jiménez. Deterministic global optimization algorithm based on outer approximation for the parameter estimation of nonlinear dynamic biological systems. *BMC Bioinformatics*, 13:90, 2012.

[158] N. Miskov-Zivanov, P. Zuliani, E.M. Clarke, and J.R. Faeder. Studies of biological networks with statistical model checking: Application to immune system cells. In *Proceedings of ACM Conference on Bioinformatics, Computational Biology and Biomedical Informatics (ACM-BCB 2013)*, page 728. ACM, 2013.

[159] Modelica Association Project. Reference FMUs. https://github.com/modelica/Reference-FMUs.

[160] Modelica Association Project. SSP Standard. https://ssp-standard.org.

[161] Modelon. JModelica. https://jmodelica.org.

[162] C.G. Moles, P. Mendes, and J.R. Banga. Parameter estimation in biochemical pathways: a comparison of global optimization methods. *Genome Research*, 13(11):2467–2474, 2003.

[163] S. Montani, L. Portinale, G. Leonardi, R. Bellazzi, and R. Bellazzi. Case-based retrieval to support the treatment of end stage renal failure patients. *Artif. Intell. Med.*, 37(1):31–42, 2006.

[164] M. Montazeri-Gh, M. Nasiri, M. Rajabi, and M. Jamshidfard. Actuator-based hardware-in-the-loop testing of a jet engine fuel control unit in flight conditions. *Simulation Modelling Practice and Theory*, 21(1):65–77, 2012.

[165] A.M. Mood, F.A. Graybill, and D.C. Boes. *Introduction to the Theory of Statistics (3rd Edition).* McGraw-Hill, 1974.

[166] Steve Morgan, Paul Grootendorst, Joel Lexchin, Colleen Cunningham, and Devon Greyson. The cost of drug development: a systematic review. *Health policy*, 100(1):4–17, 2011.

[167] R. Moss, T. Grosse, I. Marchant, N. Lassau, F. Gueyffier, and S. R. Thomas. Virtual patients and sensitivity analysis of the guyton model of blood pressure regulation: towards individualized models of whole-body physiology. *PLoS Computational Biology*, 8(6):e1002571, 2012.

[168] D.R. Mould and R.N. Upton. Basic concepts in population modeling, simulation, and model-based drug development. *CPT: pharmacometrics & systems pharmacology*, 1(9):1–14, 2012.

[169] L. O. Müller and E. F. Toro. A global multiscale mathematical model for the human circulation with emphasis on the venous system. *International Journal for Numerical Methods in Biomedical Engineering*, 30(7):681–725, 2014.

[170] Anitha Murugesan, Michael W Whalen, Sanjai Rayadurgam, and Mats PE Heimdahl. Compositional verification of a medical device system. In *Proceedings of the 2013 ACM SIGAda annual conference on High integrity language technology*, pages 51–64, 2013.

[171] A. Nicolai and A. Paepcke. Co-simulation between detailed building energy performance simulation and modelica hvac component models. In *Proceedings of the 12th International Modelica Conference*, number 132, pages 63–72. Linköping University Electronic Press, 2017.

[172] L. Nigro and P. F. Sciammarella. Qualitative and quantitative model checking of distributed probabilistic timed actors. *Simulation Modelling Practice and Theory*, 87:343–368, 2018.

[173] M.S. Nobile, A. Tangherloni, L. Rundo, S. Spolaor, D. Besozzi, G. Mauri, and P. Cazzaniga. Computational intelligence for parameter estimation of biochemical systems. In *IEEE CEC*, pages 1–8. IEEE, 2018.

[174] Open Source Modelica Consortium (OSMC). OpenModelica. https://openmodelica.org.

[175] Yannis Pantazis, Markos A. Katsoulakis, and Dionisios G. Vlachos. Parametric sensitivity analysis for biochemical reaction networks based on pathwise information theory. *BMC Bioinformatics*, 14:311, 2013.

[176] F. Pappalardo, G. Russo, F.M. Tshinanu, and M. Viceconti. In silico clinical trials: concepts and early adoptions. *Brief. Bioinf.*, 2018.

[177] Vinay Prasad and Sham Mailankody. Research and development spending to bring a single cancer drug to market and revenues after approval. *JAMA internal medicine*, 177(11):1569–1575, 2017.

[178] W. Raghupathi and V. Raghupathi. Big data analytics in healthcare: promise and potential. *Health Information Science and Systems*, 2(1):3, 2014.

[179] R. Rajkumar, I. Lee, L. Sha, and J. Stankovic. Cyber-physical systems: the next computing revolution. In *Design Automation Conference*, pages 731–736. IEEE, 2010.

[180] M. Ramirez, M. Papasimeon, L. Benke, N. Lipovetzky, T. Miller, and A.R. Pearce. Real–time UAV maneuvering via automated planning in simulations. In *Proceedings of 26th International Join Conference on Artificial Intelligence (IJCAI 2017)*, pages 5243–5245, 2017.

[181] H Ren and R Kumar. Simulation-based verification of bounded-horizon safety for hybrid systems using dynamic number of simulations: extended version. 2018.

[182] B. Ribba, G. Kaloshi, M. Peyre, D. Ricard, V. Calvez, M. Tod, B. Čajavec-Bernard, A. Idbaih, D. Psimaras, L. Dainese, J. Pallud, S. Cartalat-Carel,

J.-Y. Delattre, J. Honnorat, E. Grenier, and F. Ducray. A tumor growth inhibition model for low-grade glioma treated with chemotherapy or radiotherapy. *Clinical Cancer Research*, 18(18):5071–5080, 2012.

[183] T.R. Rieger, R.J. Allen, L. Bystricky, Y. Chen, G.W. Colopy, Y. Cui, A. Gonzalez, Y. Liu, R.D. White, R.A. Everett, H.T. Banks, and C.J. Musante. Improving the generation and selection of virtual populations in quantitative systems pharmacology models. *Progress in biophysics and molecular biology*, 139:15–22, 2018.

[184] S. Röblitz, C. Stötzel, P. Deuflhard, H.M. Jones, D.-O. Azulay, P. van der Graaf, and S.W. Martin. A mathematical model of the human menstrual cycle for the administration of GnRH analogues. *Journal of Theoretical Biology*, 321:8–27, 2013.

[185] F. Rossi, P. Van Beek, and T. Walsh, editors. *Handbook of Constraint Programming*. Elsevier, 2006.

[186] P.P. Roy and K. Roy. Molecular docking and QSAR studies of aromatase inhibitor androstenedione derivatives. *Journal of Pharmacy and Pharmacology*, 62(12):1717–1728, 2010.

[187] M. Rungger and M. Zamani. SCOTS: A tool for the synthesis of symbolic controllers. In *Proceedings of 19th ACM International Conference on Hybrid Systems: Computation and Control (HSCC 2016)*, pages 99–104. ACM, 2016.

[188] S. Schaller, J. Lippert, L. Schaupp, T. R. Pieber, A. Schuppert, and T. Eissing. Robust pbpk/pd-based model predictive control of blood glucose. *IEEE Transactions on Biomedical Engineering*, 63(7):1492–1504, July 2016.

[189] S. Schaller, S. Willmann, J. Lippert, L. Schaupp, T.R. Pieber, A. Schuppert, and T. Eissing. A generic integrated physiologically based whole-body model of the glucose-insulin-glucagon regulatory system. *CPT Pharmacometrics Syst. Pharmacol.*, 2(8):1–10, 2013.

[190] J.-F. Schlender, M. Meyer, K. Thelen, M. Krauss, S. Willmann, T. Eissing, and U. Jaehde. Development of a whole-body physiologically based pharmacokinetic approach to assess the pharmacokinetics of drugs in elderly individuals. *Clinical Pharmacokinetics*, 55(12):1573–1589, 2016.

[191] L. Schmiester, Y. Schälte, F. Fröhlich, J. Hasenauer, and D. Weindl. Efficient parameterization of large-scale dynamic models based on relative measurements. *Bioinformatics*, 36(2):594–602, 2019.

[192] Viktor Schuppan and Armin Biere. Liveness checking as safety checking for infinite state spaces. *Electronic Notes in Theoretical Computer Science*, 149(1):79–96, 2006.

[193] G. Schweiger, C. Gomes, G. Engel, I. Hafner, J. Schoeggl, A. Posch, and T. Nouidui. An empirical survey on co-simulation: Promising standards, challenges and research needs. *Simulation Modelling Practice and Theory*, 2019.

[194] E.M. Shockley, J.A. Vrugt, and C.F. Lopez. Pydream: high-dimensional parameter inference for biological models in python. *Bioinformatics*, 34(4):695–697, 2018.

[195] S. Sinisi, V. Alimguzhin, T. Mancini, and E. Tronci. Reconciling interoperability with efficient Verification and Validation within open source simulation environments. *Simulation Modelling Practice and Theory*, 2021.

[196] S. Sinisi, V. Alimguzhin, T. Mancini, E. Tronci, and B. Leeners. Complete populations of virtual patients for in silico clinical trials. *Bioinformatics*, 2020.

[197] S. Sinisi, V. Alimguzhin, T. Mancini, E. Tronci, F. Mari, and B. Leeners. AI-guided synthesis of personalised pharmacological treatments via in silico clinical trials. In *2nd Workshop on Artificial Intelligence and Formal Verification, Logic, Automata, and Synthesis (OVERLAY 2020)*, 2020.

[198] S. Sinisi, V. Alimguzhin, T. Mancini, E. Tronci, F. Mari, and B. Leeners. Optimal personalised treatment computation through in silico clinical trials on patient digital twins. *Fundamenta Informaticae*, 174(3–4):283–310, 2020.

[199] Stefano Sinisi, Vadim Alimguzhin, Toni Mancini, Enrico Tronci, Federico Mari, and Brigitte Leeners. Optimal personalised treatment computation through in silico clinical trials on patient digital twins. *Fundamenta Informaticae*, 174(3-4):283–310, 2020.

[200] E.D. Sontag. *Mathematical Control Theory: Deterministic Finite Dimensional Systems (2nd Ed.)*. Springer, 1998.

[201] P. Stapor, D. Weindl, B. Ballnus, S. Hug, C. Loos, A. Fiedler, S. Krause, S. Hroß, F. Fröhlich, and J. Hasenauer. Pesto: Parameter estimation toolbox. *Bioinformatics*, 34(4):705–707, 2018.

[202] D. Teutonico, F. Musuamba, H.J. Maas, A. Facius, S. Yang, M. Danhof, and O. Della Pasqua. Generating virtual patients by multivariate and discrete re-sampling techniques. *Pharmaceutical research*, 32(10):3228–3237, 2015.

[203] C. Thule, K. Lausdahl, C. Gomes, G. Meisl, and P. G. Larsen. Maestro: The into-cps co-simulation framework. *Simulation Modelling Practice and Theory*, 92:45–61, 2019.

[204] N.A. Trayanova, P.M. Boyle, and P.P. Nikolov. Personalized imaging and modeling strategies for arrhythmia prevention and therapy. *Current Opinion in Biomedical Engineering*, 5:21–28, 2018.

[205] E. Tronci, T. Mancini, I. Salvo, S. Sinisi, F. Mari, I. Melatti, A. Massini, F. Davi', T. Dierkes, R. Ehrig, S. Röblitz, B. Leeners, T.H.C. Krüger, M. Egli, and F. Ille. Patient-specific models from inter-patient biological models and clinical records. In *Proceedings of 14th International Conference on Formal Methods in Computer-Aided Design (FMCAD 2014)*, pages 207–214. IEEE, 2014.

[206] N. Tsiantis, E. Balsa-Canto, and J.R. Banga. Optimality and identification of dynamic models in systems biology: an inverse optimal control framework. *Bioinformatics*, 34(14):2433–2440, 2018.

[207] U.S. Society of Assisted Reproductive Technology (SART). 2015 report, 2015.

[208] U.S.A. Food and Drug Administration. Advancing regulatory science at FDA: A strategic plan. https://www.fda.gov/downloads/ScienceResearch/SpecialTopics/RegulatoryScience/UCM268225.pdf, 2011.

[209] U.S.A. Food and Drug Administration. Reporting of computational modeling studies in medical device submissions, 2016. FDA-2013-D-1530.

[210] U.S.A. Food and Drug Administration. Physiologically based pharmacokinetic analyses – format and content guidance for industry, 2018. FDA-2016-D-3969.

[211] M. Vallati, D. Magazzeni, B. De Schutter, L. Chrpa, and T.L. McCluskey. Efficient macroscopic urban traffic models for reducing congestion: A PDDL+ planning approach. In *Proceedings of 30th National Conference on Artificial Intelligence (AAAI 2016)*, pages 3188–3194. AAAI, 2016.

[212] S.C. van Dijkman, S.G. Wicha, M. Danhof, and O.E. Della Pasqua. Individualized dosing algorithms and therapeutic monitoring for antiepileptic drugs. *Clinical Pharmacology & Therapeutics*, 103(4):663–673, 2018.

[213] S.V. Vaseghi. *Advanced Digital Signal Processing and Noise Reduction*. John Wiley & Sons, 2009.

[214] H. Wang, R.J. Sové, M. Jafarnejad, S. Rahmeh, E.M. Jaffee, V. Stearns, E.T. Roussos Torres, R.M. Connolly, and A.S. Popel. Conducting a virtual

clinical trial in HER2-negative breast cancer using a quantitative systems pharmacology model with an epigenetic modulator and immune checkpoint inhibitors. *Frontiers in Bioengineering and Biotechnology*, 8:141, 2020.

[215] L. Wang, Y. Zhang, C. Yin, H. Zhang, and C. Wang. Hardware-in-the-loop simulation for the design and verification of the control system of a series–parallel hybrid electric city-bus. *Simulation Modelling Practice and Theory*, 25:148–162, 2012.

[216] Q. Wang and E. M. Clarke. Formal modeling of biological systems. In *Proceedings of 2016 IEEE International High Level Design Validation and Test Workshop (HLDVT)*, pages 178–184. IEEE, 2016.

[217] T. Wekerle, A. Pfouga, J. Stjepandic, and P. Mai. Intellectual property protection in smart systems engineering on exchange of simulation models. *Advances in transdisciplinary engineering*, 7:198–207, 2018.

[218] D.S. Weld. Recent advances in ai planning. *AI magazine*, 20(2):93–93, 1999.

[219] E. Widl, W. Müller, A. Elsheikh, M. Hörtenhuber, and P. Palensky. The fmi++ library: A high-level utility package for fmi for model exchange. In *Workshop on modeling and simulation of cyber-physical energy systems (MSCPES)*, pages 1–6. IEEE, 2013.

[220] S. Willmann, J. Lippert, M. Sevestre, J. Solodenko, F. Fois, and W. Schmitt. PK-Sim$^{\H{o}}$: a physiologically based pharmacokinetic 'whole-body' model. *BIOSILICO*, 1(4):121–124, 2003.

[221] Wolfram. SystemModeler. https://www.wolfram.com/system-modeler.

[222] Wu Hsiung Wu, Feng Sheng Wang, and Maw Shang Chang. Dynamic sensitivity analysis of biological systems. *BMC Bioinformatics*, 9(S-12), 2008.

[223] F. Yalcinkaya, E. Kizilkaplan, and A. Erbas. Mathematical modelling of human heart as a hydroelectromechanical system. In *Proceedings of 8th International Conference on Electrical and Electronics Engineering (ELECO 2013)*, pages 362–366. IEEE, 2013.

[224] Håkan LS Younes, Marta Kwiatkowska, Gethin Norman, and David Parker. Numerical vs. statistical probabilistic model checking. *International Journal on Software Tools for Technology Transfer*, 8(3):216–228, 2006.

[225] P. Zuliani, A. Platzer, and E.M. Clarke. Bayesian statistical model checking with application to Stateflow/Simulink verification. *Formal Methods in System Design*, 43(2):338–367, 2013.

List of Acronyms

AI	Artificial Intelligence
aNMAE	Average Normalised Mean Absolute Error
API	Application Programming Interface
APP	All-Different Phenotype Population
BFS	Breadth-First Search
BMD	BioModels Database
CAPP	Complete All-Different Phenotype Population
CPS	Cyber-Physical System
CS	Co-Simulation
CSP	Constraint Satisfaction Problem
DFT	Discrete Fourier Transform
E2	Estradiol
EMA	European Medicines Agency
FDA	U.S. Food & Drugs Administration
FMI	Functional Mock-up Interface
FMU	Functional Mock-up Unit
FSH	Follicle-Stimulating Hormone
GnRH	Gonadotropin-Releasing Hormone
HPC	High Performance Computing

HPG	Hypothalamic–Pituitary–Gonadal
iid	independent and identically distributed
IP	Intellectual Property
ISCT	In Silico Clinical Trial
KPI	Key Performance Indicator
LH	Luteinizing Hormone
MAPE	Mean Absolute Percentage Error
MBSE	Model-based Systems Engineering
ME	Model Exchange
MPC	Model Predictive Control
MSL	Modelica Standard Library
NED	Normalised Energy Difference
ODE	Ordinary Differential Equation
P4	Progesterone
PD	PharmacoDynamics
PHML	Physiological Hierarchy Markup Language
PK	PharmacoKinetics
PK/PD	Pharmacokinetics/Pharmacodynamics
SBML	Systems Biology Markup Language
SMC	Statistical Model Checking
SMT	Satisfiability Modulo Theories
SSP	System Structure and Parametrization
STS	Scalable Test Suite
T1DM	Type-I Diabetes Mellitus
UML	Unified Modelling Language

V&V	Verification and Validation
VP	Virtual Patient
VPH	Virtual Physiological Human
XML	Extensible Markup Language