

©Copyright 2010  
Maxwell Lewis Neal



**Modular, semantics-based composition of biosimulation models**

**Maxwell Lewis Neal**

**A dissertation submitted in partial fulfillment of the requirements for the degree of**

**Doctor of Philosophy**

**University of Washington**

**2010**

**Program Authorized to Offer Degree:  
Medical Education and Biomedical Informatics**

UMI Number: 3421901

All rights reserved

**INFORMATION TO ALL USERS**

The quality of this reproduction is dependent upon the quality of the copy submitted.

In the unlikely event that the author did not send a complete manuscript and there are missing pages, these will be noted. Also, if material had to be removed, a note will indicate the deletion.



UMI 3421901

Copyright 2010 by ProQuest LLC.

All rights reserved. This edition of the work is protected against unauthorized copying under Title 17, United States Code.



ProQuest LLC  
789 East Eisenhower Parkway  
P.O. Box 1346  
Ann Arbor, MI 48106-1346

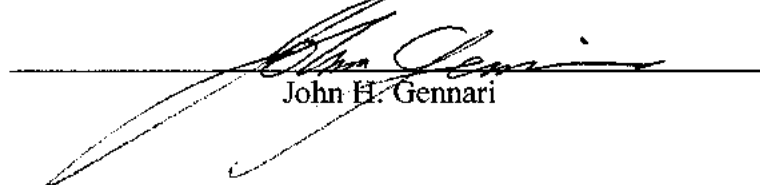
University of Washington  
Graduate School

This is to certify that I have examined this copy of a doctoral dissertation by

Maxwell Lewis Neal

and have found that it is complete and satisfactory in all respects, and that any and all  
revisions required by the final examining committee have been made.

Chair of the Supervisory Committee:

  
John H. Gennari

Reading Committee:

  
John H. Gennari

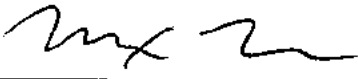
  
Daniel L. Cook

  
James F. Brinkley

  
Herbert M. Sauro

Date: 9 June 2010

In presenting this dissertation in partial fulfillment of the requirements for the doctoral degree at the University of Washington, I agree that the Library shall make its copies freely available for inspection. I further agree that extensive copying of the dissertation is allowable only for scholarly purposes, consistent with "fair use" as prescribed in the U.S. Copyright Law. Requests for copying or reproduction of this dissertation may be referred to ProQuest Information and Learning, 300 North Zeeb Road, Ann Arbor, MI 48106-1346, 1-800-521-0600, to whom the author has granted "the right to reproduce and sell (a) copies of the manuscript in microform and/or (b) printed copies of the manuscript made from microform."

Signature 

Date 6/10/2010

University of Washington

**Abstract**

Modular, semantics-based composition of biosimulation models

Maxwell Lewis Neal

Chair of the Supervisory Committee:

Dr. John H. Gennari

Medical Education and Biomedical Informatics

Biosimulation models are valuable, versatile tools used for hypothesis generation and testing, codification of biological theory, education, and patient-specific modeling. Driven by recent advances in computational power and the accumulation of systems-level experimental data, modelers today are creating models with an unprecedented level of complexity. These researchers need tools that manage this complexity and scale across biological levels of organization and physical domain. Historically, many industries have addressed the issue of complexity by adopting a *modular* product design. In order to apply this approach to the field of biosimulation, existing models must be cast as interoperable components. However, modelers today use a variety of simulation languages so that interoperability is the exception rather than the rule.

For my dissertation research I have worked on the challenges of modularity and interoperability within biosimulation. I helped develop a modular, multi-scale, multi-domain modeling approach called SemSim that provides broad model interoperability. The SemSim approach includes a declarative model description format that can capture the computational *and semantic* information in existing models, thereby converting them into interoperable,

reusable components. Because they interoperate at the semantic level, SemSim models offer opportunities to automate common composition and decomposition tasks beyond currently available methods. For my dissertation work I created and tested a software tool called SemGen that helps automate the modular composition and decomposition of SemSim models. With this tool, users can 1) convert existing models into the SemSim format and annotate them with semantic data, 2) automatically decompose SemSim models into interoperable sub-models, 3) semi-automatically merge SemSim models into larger systems, and 4) encode SemSim models in an executable simulation format. As a proof-of-concept demonstration of modular modeling, I used SemGen to perform a set of model composition and decomposition tasks using models of hemodynamics, neural signaling, molecular diffusion, and chemical pathway kinetics. This demonstration establishes SemGen's capabilities for automating the modular composition and decomposition of biosimulation models across physical scales and physical domains. Thus, SemGen has the potential to advance the entire field of biosimulation by spurring the development of complex models for biological research, drug target identification, and patient-specific modeling.



# Table of Contents

List of Figures.....	v
List of Tables.....	vi
Acknowledgements.....	vii
Dedication.....	viii
Chapter 1 : Introduction .....	1
1.1 Motivation for research.....	1
1.2 Solution approach.....	2
1.3 Research tasks .....	4
1.4 Conclusions .....	6
1.5 Dissertation overview .....	7
Chapter 2 : Biosimulation Overview and Current Challenges .....	9
2.1 The utility of biosimulation models.....	9
2.1.1 Biosimulation models are used to represent, codify and test biological theory ..	10
2.1.2 Biosimulation as an alternative to <i>in vivo</i> animal testing.....	10
2.1.3 Biosimulation for medical education.....	11
2.1.4 Patient-specific modeling for clinical decision support.....	12
2.2 Current challenges in biosimulation: complexity.....	13
2.3 Current challenges in biosimulation: interoperability and reuse.....	15
2.3.1 The Systems Biology Markup Language (SBML).....	16
2.3.2 CellML .....	17
2.3.3 Mathematical Modeling Language (MML) .....	18
2.4 Summary .....	19
Chapter 3 : Addressing the challenges of complexity and interoperability in biosimulation with a modular design approach.....	20
3.1 Component-based software engineering.....	21
3.2 Simulation interoperability .....	22
3.2.1 Levels of interoperability for simulation compositions .....	23
Chapter 4 : The SemSim approach for semantic interoperability of biosimulation models ...	26

4.1 SemSim provides syntactic interoperability by capturing a model's computational aspects .....	26
4.2 SemSim provides semantic interoperability of models by capturing the biological meaning of their codewords .....	27
4.3 Reference ontologies for annotating SemSim models.....	31
4.3.1 The Foundational Model of Anatomy .....	32
4.3.2 The Ontology of Physics for Biology.....	33
4.3.3 The Chemical Entities of Biological Interest Ontology.....	34
4.4 Composite annotations .....	34
4.5 The SemSim model template .....	37
4.6 The SemSim approach to interfaces between models.....	38
4.7 Preliminary work validating the SemSim approach.....	40
Chapter 5 : The SemGen tool for semantic composition of biosimulation models.....	44
5.1 Annotator .....	44
5.1.1 Specifying composite annotations.....	49
5.1.1.1 Searching for OWL classes in ontologies .....	50
5.1.1.2 Creating custom annotation classes .....	53
5.1.2 Non-composite annotations.....	53
5.1.3 Human-readable annotations.....	54
5.2 Extractor .....	54
5.2.1 Previewing the extracted model with a visualization tool .....	56
5.2.2 Extraction by physical entity annotation.....	57
5.2.3 Extraction by specific model codewords.....	59
5.2.4 Extraction by network cluster identification.....	60
5.3 Merger .....	64
5.3.1 Automatic detection of semantic overlap.....	64
5.3.2 Manual mappings between codewords.....	66
5.3.3 Resolving points of semantic overlap.....	67
5.3.4 Resolving points of syntactic overlap.....	69
5.3.5 Resolving conflicts with units.....	70

5.4	Coder.....	70
5.5	Summary .....	71
Chapter 6 : Using SemGen for semantic composition of models: proof of concept .....		72
6.1	Task 1: Merging the CV, BARO and VSM models.....	73
6.1.1	Merging the CV and BARO models.....	74
6.1.2	Merging the CV+BARO and VSM models .....	76
6.2	Task 2: Merging the Nielsen et al. glycolysis model with the Chassagnole et al. pentose phosphate pathway model.....	79
6.2.1	Creating SemSim versions of the metabolism models .....	81
6.2.1.1	Errors and inconsistencies in curation of the metabolism models.....	82
6.2.1.2	Limitations in annotating the metabolism models.....	83
6.2.2	Extracting the pentose phosphate pathway from the Chassagnole et al. model..	84
6.2.3	Merging the PPP shunt with the Nielsen et al. glycolysis model.....	85
6.3	Summary .....	91
Chapter 7 : Results from applying the cluster identification method to test models.....		93
7.1	Results of clustering analysis on the CV model.....	93
7.2	Results of clustering analysis on the Chassagnole et al. carbon metabolism model..	96
7.3	Discussion .....	100
Chapter 8 : Limitations, future directions, and summary.....		101
8.1	Research limitations .....	101
8.1.1	Limitations in semantic annotation.....	101
8.1.2	Limitations in SemGen's interoperability with other coding languages.....	103
8.1.3	Limitations in merging models.....	104
8.1.4	Limitations in the types of models used for this project.....	105
8.2	Future directions.....	106
8.2.1	The annotation bottleneck .....	106
8.2.1.1	Finding the right annotation component .....	106
8.2.1.2	Reusing composite annotations .....	107
8.2.2	Merging Wizard.....	107
8.2.3	User testing and evaluation .....	108

8.3 Summary ..... 109

## List of Figures

Figure 1. The vision for modular interoperability and reuse of SemSim models.....	4
Figure 2. An illustration of a complex biosimulation model.....	14
Figure 3. The SemSim architecture.....	30
Figure 4. Initial merging test using the SemSim architecture .....	41
Figure 5. The SemGen Annotator tool.....	48
Figure 6. The “Find OWL class” dialog within SemGen’s composite annotation editor.....	50
Figure 7. The SemGen Extractor tool .....	57
Figure 8. The interface for identifying computational clusters within a SemSim model .....	62
Figure 9. Clustering analysis on a lumped parameter cardiovascular model.....	63
Figure 10. The SemGen Merger tool .....	65
Figure 11. Comparison between the simulation results of the hand-coded and SemGen-merged CV+BARO+VSM models.....	79
Figure 12. Structure of the Nielsen et al. glycolysis model.....	80
Figure 13. Structure of the Chassagnole et al. carbon metabolism model .....	81
Figure 14. Structure of the target Glycolysis+PPP model.....	86
Figure 15. Numerical results from the Glycolysis+PPP merged model .....	91
Figure 16. Structure of the Chassagnole et al. metabolic model showing the groupings of chemical species when the clustering analysis identified ten distinct clusters .....	98

## List of Tables

Table 1. Semantic equivalencies automatically identified by the Merger tool when merging the pentose phosphate pathway model with the Nielsen et al. glycolysis model .....	87
Table 2. Two levels of network clustering performed on the CV model.....	95
Table 3. Grouping of chemical entities in the Chassagnole et al. model when the clustering algorithm identified ten distinct clusters. ....	99

## **Acknowledgements**

I would like to express my sincere thanks to my mentors and collaborators Drs. Daniel Cook and John Gennari. Their patience, dedication, and tenacity elevated my research beyond what I could ever accomplish alone. I would also like to give special acknowledgements to the National Library of Medicine and the American Heart Association for funding my graduate research. Also, to all my supportive family and friends - together you were the sounding board, the cheerleader, and counselor that helped make the last four years a success.

**Dedicated to the memories of Don and Marge Lewis.**



## **Chapter 1: Introduction**

For decades researchers in the biomedical sciences have used quantitative computational models to understand the dynamics of biological processes. These models are versatile biomedical tools used to codify biological theory, test hypotheses about biological systems, identify pharmacological targets, educate students, and, more recently, to provide clinical decision support through patient-specific modeling. As computational power has increased over the years, models have become increasingly complex. While the first biosimulation models related only a handful of computational variables, models today can include hundreds and sometimes thousands of variables and equations. These models hold vast potential for use in biological research and medical decision support; however, significant challenges exist within the biosimulation field that prevent researchers from realizing this potential. For my dissertation research I developed and tested a modular biosimulation modeling solution designed to address these challenges.

### ***1.1 Motivation for research***

Increasing computational power gives modelers the means to more thoroughly codify complex biological systems. However, as their simulations grow in complexity, modelers find it more difficult to manage, edit and debug their models by hand. Currently, more automated tools for managing simulation model complexity are scant: most models are still manually encoded, and due to a lack of scalable standards for model coding and reuse, many modelers cannot readily share and build upon previously coded models. The biosimulation community therefore has a growing need for tools that will help them more efficiently build, manage and reuse their models. Such tools are crucial for accelerating research on complex, systems-level biological processes, and integrating the vast amounts of systems-level biological data currently available. Additionally, such technology has the potential to catalyze the growth of the emerging field of patient-specific modeling industry, wherein clinicians use models tuned to match individual patient data for decision support [1]. Given the complexities of human physiology and anatomy, and the many steps required to develop and deploy models for clinical use, a more automated approach to model building, testing,

and management would stimulate the production of patient-specific models.

In this dissertation I describe the implementation of a *modular*, semantics-based methodology for model composition that addresses the current problems of model complexity and interoperability. This implementation is built on the premise that a modular approach to biosimulation modeling will reduce the burdens of building complex models and advance the creation of increasingly complex and robust simulations.

As in the automobile, aircraft and electronics industries, the field of biosimulation began with hand-made, custom designed products. Eventually, however, the former industries eventually migrated to a modular production approach that streamlined product design, creation, refinement and recomposition in order to meet broad customer demand. These once cottage industries evolved into large, modern manufacturing disciplines that drastically changed the way people travel, communicate, etc. The field of biosimulation stands at a similar point in its evolution. The recent explosion of biological data has created a demand for complex dynamic models that help researchers understand biological chemical networks, pharmaceutical companies are increasingly interested in *in silico* drug discovery methods, and researchers eager to tap the vast potential of patient-specific modeling require increasingly efficient methods for creating and validating their models. A modular approach to modeling offers solutions to these demands, and provides the biosimulation field an opportunity to evolve into a larger, more integrated discipline that can meet the challenges inherent in modeling biological complexity.

## ***1.2 Solution approach***

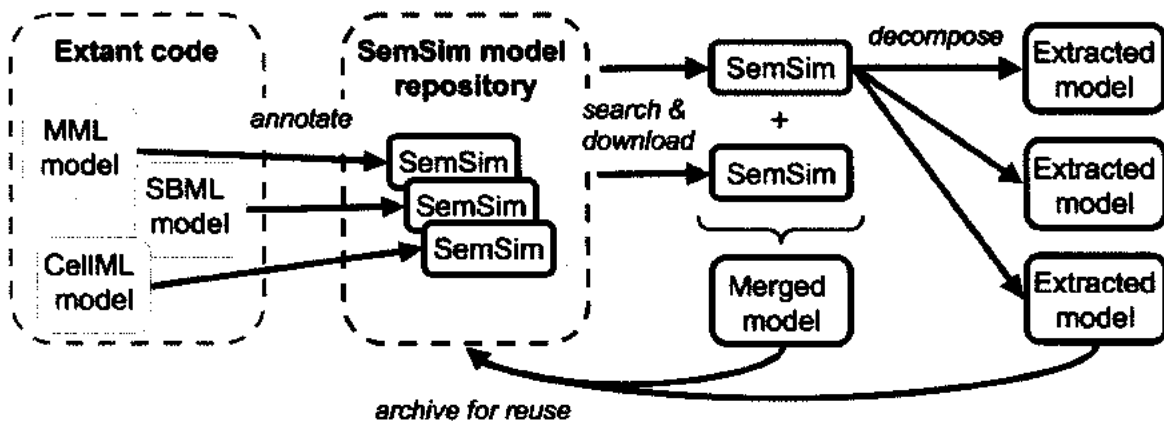
Researchers have recently advocated a modular approach to biosimulation modeling [2-6]; however, they have not been able to reach a broad consensus on how to implement a modular modeling paradigm that scales to all levels of biological organization and physical domains. Any modular approach to modeling must allow models to interoperate. However, modelers use a variety of different simulation languages, and model interoperability among the biosimulation community is currently the exception rather than the rule. In this dissertation I

describe the Semantic Simulation (SemSim) approach to modularizing biosimulation models, which my colleagues and I developed to address the challenge of model interoperability. The SemSim approach is unique and unprecedented in that it scales to all levels of biological organization and all physical domains. The approach casts biological models in a format that makes them interoperable on a semantic level. As I will demonstrate in this dissertation, this level of interoperability offers more opportunities to automate model composition and decomposition tasks, providing a powerful approach to modular modeling. Semantic interoperability relies on machine-readable definitions of model components. In other words, a computer must be able to process the *biological meaning* of a model's contents in order for it to interoperate at the semantic level. To implement semantic interoperability in the SemSim format I used Semantic Web technologies like the Web Ontology Language (OWL) and web services. These technologies provide crucial tools for linking the contents of biosimulation models to web-based knowledge repositories containing information needed for machine-readable annotations.

The machine-readable definitions used in SemSim models are the key to semantic interoperability, and thus, modularity within biosimulation. In this dissertation I detail the advancements my colleagues and I made in determining how to construct these machine-readable definitions in order to unambiguously define the semantics of model codewords. Of primary importance is our development of a scheme for creating *composite annotations* wherein users link concepts from multiple reference ontologies to form single annotations. These composite annotations provide annotators the expressivity needed to unambiguously define the wide range of biological concepts used in biosimulation models. At the same time, the SemSim approach of linking concepts in reference ontologies ensures that machine-readable definitions in SemSim models use standardized, well-defined biological terms, an important feature for a semantic interoperability framework that must identify semantic equivalencies between models.

The ultimate, long-term goal of the SemSim project is to create a repository of semantically interoperable biosimulation models that, together, represent biological processes across

physical scales and physical domains. Modelers will use this repository as a bank of reusable, interoperable component models that can be composed together to build complex biosimulation models, much like a collection of Lego bricks. Figure 1 shows a block diagram illustrating this vision. Because SemSim models are semantically interoperable, a computer system will be able to automatically identify the points of coupling between merged models, prompt the user to resolve overlaps, and then automatically merge the models. The SemSim architecture represents biosimulation models in a declarative format - one that can be readily translated into executable simulation code in a variety of languages such as Mathematical Modeling Language (MML), MATLAB, Systems Biology Markup Language (SBML), CellML, etc. Eventually, modelers will be able to select SemSim models from the repository, use them for composition or decomposition tasks, and then translate new SemSim models into whatever simulation language they are most comfortable using.



**Figure 1. The vision for modular interoperability and reuse of SemSim models. Annotating extant model code produces SemSim models that are stored in a searchable repository. Users download the SemSim models and perform model composition and decomposition tasks. New models are in turn added to the repository for reuse.**

### 1.3 Research tasks

While the SemSim approach provides a knowledge representation framework for making biosimulation models interoperable, ultimately users need a software tool that helps automate

the modular composition and decomposition of SemSim models. For my dissertation work I developed a software tool called **SemGen** that provides this functionality. SemGen provides intelligent support to help modelers convert existing models into the SemSim format, annotate them with semantic data, merge them, and decompose them into reusable sub-models. Thus, SemGen represents a significant step towards the longer-term goals of the SemSim vision illustrated in Figure 1.

As a proof-of-concept demonstrating SemGen's utility as a modular modeling tool, I performed two end-to-end demonstrations of modular modeling. First, I used SemGen to convert three previously hand-coded MML models related to the human cardiovascular system into the SemSim format. Then, after annotating their codewords with machine-readable definitions, I merged these models together in a semi-automated, modular fashion to produce a new, executable, multi-scale cardiovascular model. In my second end-to-end demonstration I converted two chemical network models coded in the Systems Biology Markup Language (SBML) into the SemSim format. Both models simulate the dynamics of cellular metabolism: the first, by Nielsen et al. [7] simulates the *Saccharomyces cerevisiae* glycolysis pathway. The second, by Chassagnole et al. [8] simulates the pentose phosphate pathway (PPP) in addition to glycolysis in *Escherichia coli*. Using SemGen I automatically extracted the PPP component of the Chassagnole et al. model into its own SemSim model, then merged the PPP component with the Nielsen et al. model to produce a novel chemical network model that couples the Nielsen et al. representation of glycolysis with the Chassagnole et al. representation of the PPP shunt.

In addition to these proof-of-concept demonstrations, I also investigated the decompositions produced by the SemGen extraction method that identifies modular clusters within a SemSim model. This method provides a means of discovering modular components that already exist within a model's computational network. In order to determine whether this clustering method might provide a general means of separating a model's components into *semantically* related modules, I used it to identify clusters within the CV and Chassagnole et al. models and examined whether the resulting decompositions reflected the models' semantic

architectures.

### ***1.4 Conclusions***

Together, my two end-to-end demonstrations establish SemGen's utility for composing and decomposing models across physical scales (molecules, tissues, organ parts) and across physical domains (chemical reaction kinetics, electrochemical kinetics, diffusion kinetics and fluid kinetics). SemGen can convert a wide variety of biosimulation models into the SemSim format, making the models interoperable and available for modular composition and decomposition tasks. By promoting model interoperability and modularity, SemGen addresses the current challenges of model complexity and interoperability mentioned above.

My results in applying the cluster identification extraction tool suggest that computational connectedness within a model does not always reflect its semantic connectedness, and there is no guarantee that a model will decompose modularly according to a user's conception of its semantic architecture. However, the cluster identification algorithm does delineate useful, semantically related modules in some cases.

My research has a potentially broad impact within the fields of patient-specific modeling, bioengineering, systems biology, synthetic biology, medical education, and *in silico* pharmacology. Provided with a repository of semantically interoperable SemSim models and SemGen's composition/decomposition tools, modelers in these fields will have the opportunity to construct increasingly sophisticated simulations in a more automated, less error-prone fashion. Additionally, because developing SemGen was a multidisciplinary task that bridged the disciplines of bioengineering and biomedical informatics, the impact of my research extends to the latter field as well. In particular, the SemGen scheme for creating *composite*, machine-readable annotations is applicable not only to models, but also to the annotation of biomedical data in general. SemGen's composite annotation scheme utilizes Semantic Web technologies; therefore it offers the greater biomedical research community a standardized means of integrating and reusing data sources. SemGen represents a technology that will become increasingly valuable as the biological sciences shift toward more

collaborative, less isolated research projects [9] and the need for tools that integrate scientific data and computational models grows.

### ***1.5 Dissertation overview***

- I introduce the field of biosimulation in Chapter 2, provide an overview of the utility of biosimulation modeling, and discuss the current challenges of complexity and interoperability within the field.
- In Chapter 3, I discuss modularity as a design principle, and how its application to the field of modeling provides solutions to these current challenges.
- In Chapter 4 I detail the SemSim architecture along with the levels of model interoperability it provides. I focus particularly on the concept of composite annotations, which provide a means of unambiguously defining model components using linked reference ontology terms.
- In Chapter 5 I describe SemGen, focusing on its four main components: the Annotator, Extractor, Merger and Coder tools. I discuss how SemGen can be used to compose and decompose SemSim models in a modular fashion.
- In Chapter 6 I provide two end-to-end demonstrations of modular modeling using SemGen. In the first demonstration I composed three separate models related to the cardiovascular system into a single, multi-scale model. In the second demonstration I extracted out the pentose phosphate pathway component of one carbon metabolism model and merged it with the glycolysis pathway represented in another. These two demonstrations validate the SemSim approach to modular modeling and show the utility of SemGen as a multi-scale, multi-domain model composition tool.
- In Chapter 7 I describe the decompositions that result from applying the cluster identification extraction method on two test models. From these results I assess the

method's ability to identify semantically related modules within SemSim models.

- In Chapter 8 I discuss my project's limitations, outline the future work needed to meet the vision of broad biosimulation model interoperability, and summarize the conclusions of my research.



## **Chapter 2: Biosimulation Overview and Current Challenges**

In this chapter I provide background on the field of biosimulation in order to give context for my dissertation research. As a comprehensive history of biosimulation is too vast a topic for the scope of my dissertation, I instead discuss how researchers have used biosimulation models since they appeared in the late 1800's [10] and provide some relevant examples of models within this context. I then discuss the current state of the art in biosimulation technology and present the current challenges in the field that motivated my research on modular modeling.

### ***2.1 The utility of biosimulation models***

A biosimulation model is a mathematical or computational representation of some biological phenomena. These models can be very simple, perhaps consisting of a single equation, or significantly complex, where thousands of equations describe a biological system. These models are versatile tools used in various ways within the fields of biological research, medicine, and public health, and researchers use models to simulate phenomena across all levels of biological organization, from the interaction of atoms within a molecule to processes involving whole populations of organisms. In the following sections I discuss the use of biosimulation models for 1) codification and testing of biological theory, 2) replacing animal models, 3) biological and medical education, and 4) patient-specific clinical decision support.

In order to focus my discussion in this chapter, I consider those biosimulation models that represent phenomena at physical scales below the whole-organism level but above the atomic level. That is, I consider models that simulate the interaction of molecules up to the interaction of whole organ systems. I choose this scope for my discussion based on my interest in patient-specific modeling, which primarily includes models within these physical scales, and also because the test models used in my final end-to-end demonstration of modular modeling fall within these bounds.

### **2.1.1 Biosimulation models are used to represent, codify and test biological theory**

In biomedical research, biosimulation models provide a means of representing and codifying theories of dynamic biological systems. Each biosimulation is a hypothesis about how some biological system behaves, and when researchers create such models, they represent biological theory in terms of a network of computational codewords and dependencies. This systems-level approach to biological research is in contrast to the more traditional and widespread reductionist approach that attempts to understand biological phenomena in terms of more qualitative, correlation-based relationships. Members of the relatively new field of systems biology, who focus on characterizing the interplay between components within biological networks rather than characterizing these components in isolation, have espoused the systems-level approach to biological research as their field has grown [11].

Researchers such as systems biologists use models as hypotheses testing tools by comparing their output to empirical data collected from the actual biological system being simulated. For example, Hodgkin and Huxley [12] developed their well-known squid axon action potential model for precisely this purpose – to refine and codify their theory of the biological mechanics responsible for neuronal firing.

If model results diverge unsatisfactorily from the data they are meant to reproduce, modelers must refine their hypothesis about how the biological system behaves. This usually requires tuning the values of model inputs, rewriting equations to alter the dependencies between computational variables, and/or increasing or decreasing the granularity of the model. This model validation process is iterative and often time-consuming. My hope is that my research on modular modeling will provide a method for streamlining this process, allowing biological researchers to complete modeling studies more efficiently.

### **2.1.2 Biosimulation as an alternative to *in vivo* animal testing**

Biological researchers also use biosimulation models as surrogates for live animal models. For decades researchers have relied on animal testing to identify potential therapies for

pathologies. However, there are substantial costs associated with animal testing including animal housing/feeding and laboratory supplies used for tissue fixation and preservation. Research labs also incur the responsibility of complying with the ethical guidelines associated with animal use. As computational power has grown, researchers – especially those studying pharmacodynamics and pharmacokinetics - have increasingly looked to *in silico* modeling as a means of reducing these costs. Theoretically, with a detailed enough simulation model, one could efficiently identify potential drug targets by perturbing the computer model with simulated compounds. Again, one of the advantages of using a computer model in this situation lies in the fact that it provides a quantitative, systemic view of the biology, one that cannot be fully held in, or quantitatively predicted by, a human mind. The disadvantage is that in order for an *in silico* model to give an accurate prediction of a drug's effectiveness, modelers must have some *a priori* understanding of the drug's biological impact and often times, insufficient knowledge exists about the pathways affected by pharmacological interventions. Nonetheless, researchers in the field of *in silico* pharmacological simulation have made significant progress over the past decades, and the field continues to grow. A recent review of *in silico* pharmacology by Ekins et al. [13] describes a number of modeling success stories and asserts “The chemistry–biology–informatics triad has now evolved into a life of its own and is bringing pharmacology to new heights.”

### **2.1.3 Biosimulation for medical education**

Biosimulation models are also valuable tools used for biological and medical education. For instance, computer simulations of human physiology help medical students understand the interplay between various physiological processes without the need for live subjects. As an example, medical students often learn the theory behind Starling's Law of the heart [14] early in their education. The key concepts behind this principle, which relates cardiac chamber filling to contractile force, can be readily captured in a simple computational model based on physical principles of fluid flow. (Such hemodynamics models have, in fact, existed for decades [10, 15-17].) By perturbing these models, students learn about the interplay between various physiological components. By increasing a model input value controlling

pulmonary arterial resistance, for example, a sufficiently detailed cardiovascular model will show an increase in the end-diastolic volume of the right ventricle, and an increased ventricular ejection force. By familiarizing themselves with such a simulation, students gain a more intuitive understanding of how the different factors in the cardiovascular system affect each other.

Simulation modeling is prominent within anesthesiology education as well, as it supplies a means of training anesthesiologists in realistic operative scenarios without requiring live human subjects. Anesthesia models simulate a patient's vital sign responses to infusion of anesthetics or other interventions. In this way, anesthesiologists learn to anticipate the effects of infused compounds on patients during real clinical situations. Model-based anesthesia simulators are commercially available from Anesoft [18] and Advanced Simulation Corporation [19].

#### **2.1.4 Patient-specific modeling for clinical decision support**

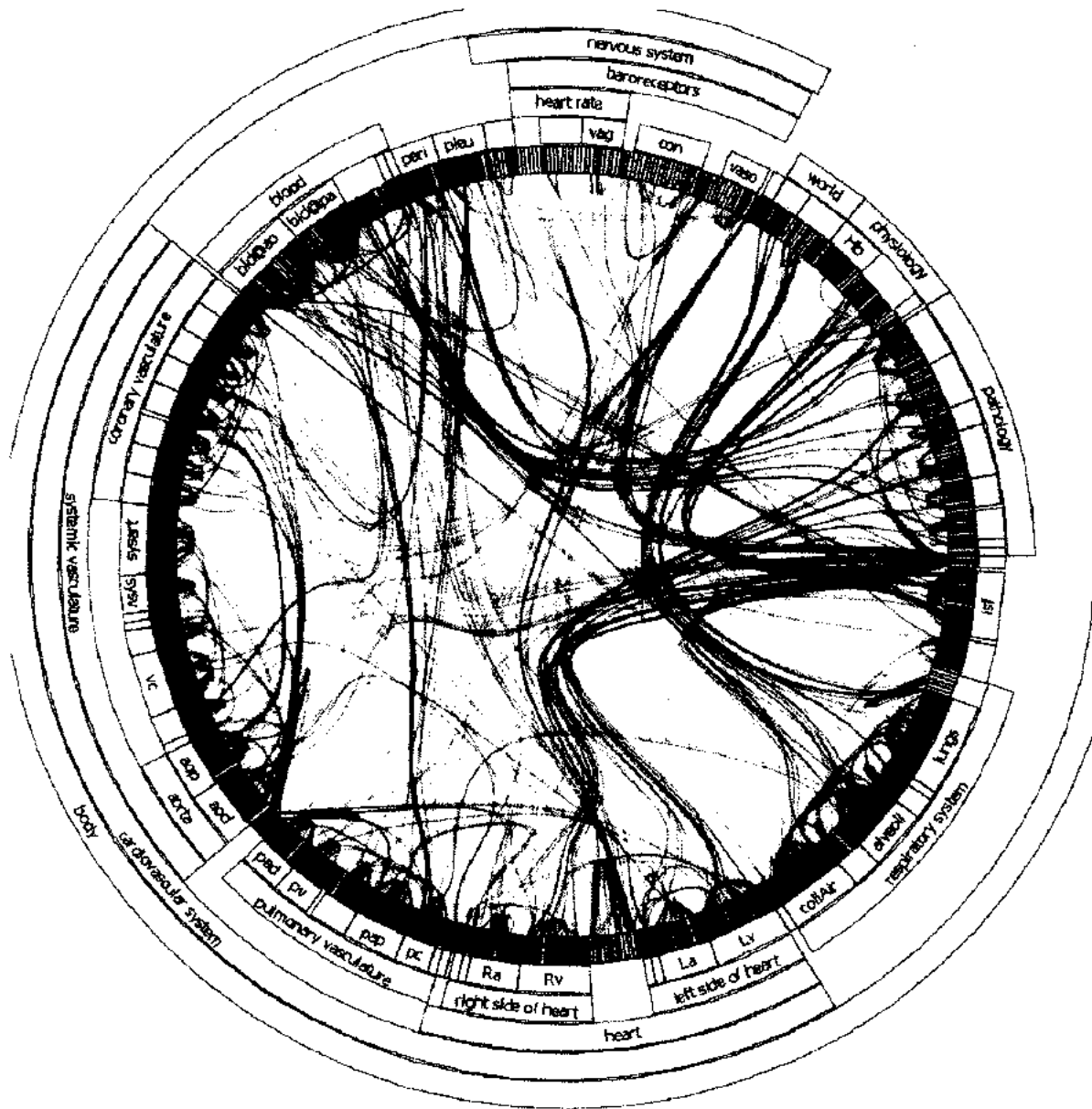
Finally, biosimulation models play the central role in the emerging field of patient-specific (PS) modeling, which involves the creation, validation and application of dynamic, usually pathological, models that are tuned to match patient-specific data. Medical practitioners can use these tuned models to investigate a patient's biology using an *in silico* avatar rather than the patient themselves. The wide range of potential applications within PS modeling includes tailoring patient therapies through *in silico* intervention testing, estimating occult physiological variables in real-time [20, 21], and determining disease etiology [22].

As discussed recently by Neal and Kerckhoffs [1], extant PS models simulate a wide variety of processes in the human body, including phenomena within the cardiovascular, musculoskeletal, and nervous systems and pathologies such as tumor growth. Currently, most PS modeling efforts are at the validation stage, where researchers are confirming the fidelity of their models by comparing simulation output to empirical data. Although PS modeling is an active area of research, there are relatively few published studies that assess the clinical effectiveness of a PS model once deployed in a medical care facility. This is not due to the

inaccuracy or ineffectiveness of PS models, but rather, to the large number of steps required to build a PS modeling system, validate it, deploy it in a clinical environment, and test its impact on clinical outcomes. PS modeling researchers, therefore, currently need tools that reduce the number of steps in this process. Such tools will accelerate PS modeling work and will move the field toward realizing its potential within clinical decision support. The need for these tools was one of the primary motivations behind my dissertation research. My hope is that by applying the modular compositional approach to biosimulation modeling that I describe in later chapters, PS modelers will remove many of the bottlenecks associated with creating, managing, and validating their models, thereby accelerating research within the field.

## ***2.2 Current challenges in biosimulation: complexity***

Given the explosion of biological information within the last 20 years, along with advances in computing power, modelers currently have the means to simulate biological systems on an unprecedented scale. Models published today may contain hundreds or thousands of equations and variables. However, as their simulations grow in scope and fidelity, modelers are increasingly challenged to manage the complexity of these computational systems. For example, consider the multi-scale “Highly-integrated Human” (HiH) model [23] I developed for DARPA’s Virtual Soldier Project [24]. This complex model simulates the interplay between hemodynamics, pulmonary mechanics, baroreceptor feedback, chemoreceptor feedback, and other canonical physiological processes. The model includes 473 codewords that are related through 461 equations. Figure 2 illustrates the model’s complexity. In this image, create by Dr. Gary Yngve [25] and used here with his permission, the segments of the innermost ring represent the model’s codewords, which are grouped according to an anatomical hierarchy (outer ring segments). The curved lines in the center of the ring represent the mathematical dependencies between the codewords.



**Figure 2. An illustration of a complex biosimulation model. The curved lines represent the computational connectivity between codewords in a model of integrated human physiology. Image create by Dr. Gary Yngve as part of his dissertation work at the University of Washington, used with permission.**

My work with this and other models led me to the realization that a modular approach to modeling, one built on model interoperability and reuse, would have facilitated many of my most difficult and error-prone modeling tasks. Furthermore, after developing the HiH model,

I decomposed it into component models so they could be available online as part of the Physiome project [26, 27], the international endeavor to describe human physiology using computational models. Had I initially constructed the HiH model using a modular modeling framework, I could have accomplished this task in a more automated fashion, saving valuable time and effort. In later chapters I describe the products of my dissertation research that provide more automated methods for accomplishing these kinds of model decomposition tasks.

Today, many researchers beside myself are developing models with a high level of complexity (for examples, see [28-32]). However, biosimulation modelers as a community have not yet adopted the kind of standards for model coding, curation and reuse that would minimize the burden of creating, managing and reusing complex models. Rather, modelers mainly code by hand in their computational language of choice, and existing repositories of biological models [33-36] focus on particular simulation languages and research domains. I addressed these challenges for my dissertation research by developing a modular, semantics-based, language-independent modeling approach. In Chapter 3 I discuss in more detail how the design principle of modularity can address the challenges associated with complexity in biosimulation, and in later chapters I describe the implementation of a modular modeling approach using the SemSim architecture and my own software SemGen.

### ***2.3 Current challenges in biosimulation: interoperability and reuse***

Applying a modular modeling approach is one way to meet the challenge of increasing model complexity within the field of biosimulation; however, in order for models to be composed and decomposed in a modular fashion, component models must first be interoperable. Presently there are scores of different modeling languages and platforms used within biosimulation: MATLAB, Mathematica, JSim, Systems Biology Markup Language (SBML), CellML, SCAMP, XPPAUT, Java, C, C++, FORTRAN (to name a few), and no single language provides a *lingua franca* that makes all biosimulation models interoperable and reusable. Therefore, reproducing model results across research labs and simulation languages remains a difficult, error-prone process that often requires extensive hand coding and

necessitates direct communication with original model developers [37]. Therefore, the biosimulation community as a whole stands to benefit from modeling standards that allow models at various physical scales and research domains to interoperate in a modular way. Such standards would simultaneously address the challenges posed by increasing model complexity and the heterogeneity of simulation languages.

### **2.3.1 The Systems Biology Markup Language (SBML)**

Although the biosimulation community as a whole has not adopted the standards that allow widespread interoperability and reuse of models, researchers have made some important progress in the last ten years on this front. For example, a consortium of chemical network modelers developed SBML [3] in order to make their models more interoperable and reusable. An SBML file, based on the eXtensible Markup Language (XML), is a declarative representation of a chemical network model that exists separately from numerical integration and output visualization methods. An SBML model includes a list of chemical species involved in a network and the reactions in which they participate. Simulation engines that import SBML construct a set of algebraic and ODE-based equations from the declared data in the SBML file in order to generate runnable simulation code. Members of the chemical pathway modeling community recognize SBML as an important model description standard, and over 180 software packages support this modeling standard [38], including the popular MATLAB software [39]. The SBML community also developed the *Biomodels.net* database; an online repository of curated SBML models that allows modelers to search for, investigate and download previously published chemical network models [40]. Currently the repository contains about 250 curated models.

Although SBML is an important, well-adopted modeling standard, its scope is constrained to ODE-based chemical network models. The SBML framework was not been designed to scale to other levels of biological organization or other research domains, and so the biological knowledge represented in SBML models remains focused on chemical pathways. Furthermore, much of the semantic information, specifically the physical properties of model codewords, remains implicit in SBML models. Therefore, while SBML provides chemical



network modelers with a valuable modeling tool aimed specifically at their modeling needs, it does not provide a scalable, general solution to the problem of biosimulation interoperability.

### 2.3.2 CellML

CellML is another XML-based model description standard developed by researchers at the University of Auckland as part of the International Union of Physiological Sciences Physiome project [27]. Unlike SBML, CellML does not focus on one particular modeling domain, but rather, provides a more general framework for capturing the mathematics of algebraic and ODE-based models. It also provides a means of capturing model metadata, like model author and bibliographic information and, in a limited capacity, the semantics of model codewords. The CellML specification also allows model writers to decompose a model into components, so that these sub-models can be reused separately. Although CellML is a more general solution to biosimulation model interoperability than SBML, its adoption has not been as widespread. This may be due to the SBML community's dissemination of APIs and library packages (such as libSBML [41]) that facilitate the independent development of SBML-friendly tools. CellML's developers recently made an API available [42], and this resource may increase CellML's adoption in the future. Regardless, CellML is an important modeling resource, and, because it is a more general approach to declarative model representation, the European Commission's Virtual Physiological Human Project recently adopted it as a common modeling language.

While CellML models scale across levels of biological organization, they do not contain the knowledge architecture required to advance automated model composition beyond the syntactic level. Although modelers can include some semantic information within CellML models, this information is insufficient to unambiguously define model concepts. In order to reach beyond current levels of model interoperability and automate model composition as much as possible, modelers need a more robust and thorough semantic annotation scheme for their models. In Chapter 4 I discuss the SemSim annotation scheme developed by my colleagues and I, which provides this powerful level of model interoperability.

### 2.3.3 Mathematical Modeling Language (MML)

Researchers at the University of Washington's National Simulation Resource developed MML as part of a general-purpose simulation environment called JSim [43]. Like CellML, MML was created in the interest of providing a modeling tool that could meet the modeling demands of the Physiome project. Therefore, MML accommodates modeling across physical scales and domains. Unlike CellML and SBML, MML is not an XML-based language, but rather a custom, human-readable syntax designed to facilitate model equation writing within JSim. However, the JSim software does include a function for translating an MML model into an XML-based representation (see section 5.1).

As with SBML and CellML models, MML models are distinct entities, separate from the numerical integrators used to solve them and the graphing tools used to visualize their output. In addition to algebraic and ODE equations, MML accommodates partial differential equations (PDEs) and procedural code, providing modelers more mathematical expressivity than either CellML or SBML. MML also includes a valuable unit-checking algorithm providing modelers the means to ensure unit-balance in their models. JSim can also readily import SBML and CellML models into the MML format, an important feature that I utilized for my research (see sections 5.1 and 6.2.1).

In the interests of modular modeling, the developers of MML and JSim created a method for reusing a subset of physiological models that simulate metabolite flow and exchange. These Biological Component Language (BCL) models can be imported into MML and reused any number of times within the code, much like using a class in an object-oriented programming language. Although BCL offers this kind of modularity, as with SBML and CellML, it does not currently include the knowledge architecture required for making biosimulation models *semantically* interoperable. MML and BCL can only capture the meaning of model components using in-line comments and ad-hoc variable properties. Therefore, these languages only offer a kind of *syntactic* interoperability. Within the BCL framework, interfaces between model components are immutable and users must specify them by hand in order to ensure their biological validity. A model description format that provides structured,

machine-readable semantic information, on the other hand, would automate such model composition tasks, offer a maximum number of potential interfaces between components, and make repositories of encoded models more searchable via querying tools. The SemSim framework, described in Chapter 4, provides these features, all of which are important for realizing broader modular interoperability within the field of biosimulation.

## ***2.4 Summary***

As discussed above, a modular approach to biosimulation modeling that employs semantically interoperable component models would address the challenges of both complexity and interoperability within biosimulation. Such technology goes beyond current levels of model interoperability, and would have an impact within all the disciplines mentioned above that utilize biosimulation models. In the next chapter I address in more detail the issue of modularity as a design principle, and its implementation for meeting the current challenges in biosimulation.

### **Chapter 3: Addressing the challenges of complexity and interoperability in biosimulation with a modular design approach**

In this chapter I discuss modularity as a design principle and how it can address the challenge of increasing model complexity within biosimulation. I also provide historical background on the implementation of modular, interoperable software and discuss the different levels of software interoperability that are possible with a modular design approach.

As a design principle, modularity is used widely in other disciplines for simplifying the creation and management of complex systems [44]. For example, aircraft, automobile, electronics, and arms manufacturing industries all rely on modular product development, although historically they began with one-off, non-modular development. For the most part, simulation modeling has stalled at this non-modular, initial phase, although, as discussed in the previous chapter, biosimulation has moved in a more modular direction over the past five to ten years thanks to the development of SBML and CellML [3, 5, 45].

Modelers could substantially reduce the burdens of building complex biosimulation models if they adopted a modular design approach. Generally speaking, modularity allows a complex system to be broken up into more manageable units that are “powerfully connected among themselves and relatively weakly connected to elements in other units.” [46] Rather than relying on the traditional, hand-coded, custom-made, one-off modeling approach that has been the standard for decades, modelers using a modular approach would be able to compose complex systems more efficiently, and would have access to a powerful tool for simulating integrated biological systems. Furthermore, modular modeling lends itself to a separation of concerns and can leverage the distributed, specialized nature of biological modeling work: individual researchers would be free to refine their models of interest and could avoid the responsibilities of making externally-developed models interoperable with their own.

Researchers could then share and reuse models readily, eliminating the costly efforts of re-coding model components.

Many researchers have recognized the value of modularity to the biosimulation field, (some have even called it the “Holy Grail” of biosimulation [4]), but there has been much debate on how models should be modularized [6]. In order to act as modules, simulation models must first be syntactically interoperable, i.e. they must share a common syntax so they can exchange information once linked together programmatically. However, modelers use a wide variety of simulation languages that are often created to meet specific research goals rather than promote general model interoperability. Therefore, even when models are syntactically interoperable, the modeler must provide explicit statements in the code that create biologically meaningful interfaces between modules. Modelers need a more advanced and powerful level of interoperability in order to automate these kinds of composition tasks: *semantic interoperability*. While syntactic interoperability ensures that models can communicate with each other, semantic interoperability ensures that compositions of models are biologically meaningful.

### ***3.1 Component-based software engineering***

Biosimulation models are software components, and thus simulation interoperability can be viewed as a subset of a larger body of research begun in the 1960’s on component-based software engineering (CBSE). The goal of CBSE is to develop interoperable software components that can be reused effectively among users. Some examples of component-based software architectures, also called “component models,” include Visual Basic, the Common Object Request Broker Architecture (CORBA), Enterprise JavaBeans, and Microsoft’s Component Object Model (COM). Each of these component models represents a successful application of a modular, interoperable software system for a particular domain such as network communication (CORBA) or business technology (Enterprise Java Beans). As I discuss in Chapter 4, my dissertation research employs a modeling framework called SemSim that provides a knowledge architecture for transforming existing biosimulation models into interoperable, reusable components. The SemSim framework falls within the

wider family of software component models listed above, but is unique in that it addresses specific needs in biosimulation.

It appears that solutions to common problems in simulation interoperability (or “composability”) and CBSE evolved independently [47], and it is only within recent years that simulation interoperability was labeled as a subset of the larger problem of software reuse. Many of the requirements for simulation model interoperability are the same as those for software composability in general: independently deployable components, structured component-component interfaces, and an architecture that defines how a composition is created, for example [48]. However, the unique user needs of simulation modelers present specific challenges and opportunities in developing modular simulation systems as compared to the wider field of CBSE. For example, Carnahan et al.[49] found that managing a model’s internal concept of time and its dependence on simplifying assumptions can aid simulation reuse. These same authors also discuss how modelers often treat their simulations as white boxes rather than black boxes. “Whereas users of other software only make use of their programs’ outputs, users of simulation study the internal state of their simulations in order to gain insight.” I considered this characteristic of model use when developing the SemSim framework and the SemGen tool because it suggests that information hiding, while a vital feature of modular design and CBSE, should be graded depending on a modeler’s information needs.

### ***3.2 Simulation interoperability***

Although CBSE as a research field has existed for decades, researchers have examined the problem of simulation interoperability only more recently. Much of this recent work was motivated by the simulation needs of the U.S. military and has led to the development of the High Level Architecture (HLA), a component model that integrates simulations for training, decision making, etc. in a predominantly military context [50].

Researchers in other domains have also developed composition systems for simulating electronic circuits [51], computer hardware [52], mechatronic devices (CD players, missile seeker heads, VCRs) [53], and environmental systems [54]. Simulation composition systems

for biological and clinical research have lagged behind these other fields and have only been addressed in recent years [37, 55, 56]. Researchers in biosimulation interoperability therefore have the benefit of learning from the mistakes and successes of previous work within the domains mentioned above.

Kasputis and Ng [57] go beyond Carnahan et al. [49] to articulate that an additional challenge that emerged from recent research on simulation composability: “We are discovering that unless models are designed to work together – they don’t (at least not easily and cost effectively). Without a robust, theoretically-grounded framework for design, we are consigned to repeat this problem for the foreseeable future.” One of the aims of my dissertation work was to perform preliminary tests on whether the SemSim architecture provides this “robust, theoretically-grounded framework” needed to realize the benefits of modular biosimulation composition. (I address this issue in Chapter 4.) As stated by other recent authors, researchers should be realistic about what they promise in a simulation composition system [58]. My own research on merging models [37] has shown that translating from one simulation language to another can be very difficult to automate (especially between a procedural and a declarative language), thus validating Kasputis and Ng’s point. Nonetheless, as I will show in later chapters, the results of my dissertation research demonstrate that the SemSim methodology makes a vast amount of existing biosimulation models interoperable and provides the framework for a multi-scale, multi-domain modular modeling approach.

### **3.2.1 Levels of interoperability for simulation compositions**

As in CBSE, simulation interoperability researchers recognize that there is not one, but several levels of software interoperability. Most research in CBSE addresses syntactic and semantic interoperability, but Tolk et al. [59] define six levels for simulation systems: technical, syntactic, semantic, pragmatic, dynamic, and conceptual. (Descriptions of each level listed below are summarized from the source paper.)

- **Technical interoperability:** A protocol exists for exchanging data (bits) between participating components.
- **Syntactic interoperability:** A common data format is applied to share information between components.
- **Semantic interoperability:** Components share the meaning of the information they exchange.
- **Pragmatic interoperability:** Components share a concept of the context and purpose of their application.
- **Dynamic interoperability:** Components react to time-dependent changes in their internal assumptions and constraints. Components share the effect of the system's operation over time.
- **Conceptual interoperability:** Components share a common understanding of the assumptions and constraints of a simulation's abstraction of reality.

Presently, most interoperability solutions in software engineering and simulation only provide the technical and syntactic levels. One notable exception is the semanticSBML merging tool, which provides semantic interoperability for SBML-based chemical network models [45]. However, semanticSBML's scope is limited. It relies on the fact that SBML models only represent a narrow subset of biological phenomena and the system does not scale to levels of biological organization beyond chemical pathways. In the following chapters I will describe the SemSim knowledge framework and the SemGen software tool I developed for my doctoral work that provide *semantic* interoperability of biosimulation models across multiple physical scales and physical domains.

The overall goal of my dissertation research was to develop and validate software tools that provide a robust, efficient solution for modular biosimulation composition. With these tools, biosimulation researchers will have access to a powerful new technology for codifying vast amounts of quantitative biological knowledge. As in the many other industries that have migrated from a custom to a modular product design, I anticipate that this new technology will profoundly advance any discipline making use of biosimulation models. Modelers will



be able to simulate complex biological phenomena in greater detail and fidelity, leading to radical improvements in areas like physiological research, *in silico* drug discovery, and patient-specific modeling.

## **Chapter 4: The SemSim approach for semantic interoperability of biosimulation models**

Over the past few years, in collaboration with Dr. Daniel Cook, Dr. John Gennari and Michal Galdzicki, I helped design a theoretical framework for modular biosimulation composition: the Semantic Simulation (SemSim) architecture [37, 55]. My colleagues and I designed this architecture so that models coded in a variety of languages could be translated into a standard, lightweight ontology-based model description format, independent of the physical scale(s) and physical domain(s) represented in the model. The SemSim framework provides a multi-scale, multi-domain ontology architecture that can capture all the computational aspects of an existing biosimulation model along with all the explicit and implicit semantic information about the biological meaning of model variables and equations. SemSim models consist of a logical class hierarchy and a set of logical relationships that defines model components and describes their mathematical and semantic interdependence. SemSim models are readily reusable, semantically interoperable, modular components designed to improve the automation of common model composition and decomposition tasks. In terms of the interoperability levels discussed in section 3.2.1, SemSim models are both *syntactically and semantically interoperable*: they can exchange data in a standard way, and they share biologically meaningful interfaces.

### ***4.1 SemSim provides syntactic interoperability by capturing a model's computational aspects***

Models that are syntactically interoperable are able to exchange data through interface points via a common data format [59, 60]. These models must therefore either be coded in a common simulation language, or be translated on the fly from one language to another at runtime. Given that there are a wide variety of simulation languages in existence, and modelers tend to use particular languages that help them meet the unique goals of their own research, the likelihood that the biosimulation community as a whole will choose to adopt

any single simulation language as an international standard is low. Therefore, a true *lingua franca* for syntactic composability in physiological modeling should account for the variety of languages that are used by the modeling community rather than attempt to replace them. The SemSim solution is to create a rich, language-independent, ontology-based model description format that captures the computational aspects of existing models and then allows that knowledge to be translated back into a variety of other simulation languages. My colleagues and I plan to eventually incorporate these language-specific parsing and encoding tools into SemGen in order to enable syntactic interoperability for models across multiple simulation languages. However, parsing and encoding tools require a significant amount of work to develop because programmers must thoroughly map one model description format to another. This requires in-depth knowledge of both formats involved in the conversion process. For my project I chose to focus on developing tools that only convert SemSim models to and from MML because my primary research aim was to demonstrate, in principle, that SemGen provides an environment for modular modeling at the semantic level of interoperability, not to create a set of tools that accommodates various model description formats. Of course, following this proof-of-concept demonstration, the next logical step in SemGen's development is to outfit it with a number of language conversion tools that will make the software more widely useable by members of the biosimulation community.

#### ***4.2 SemSim provides semantic interoperability of models by capturing the biological meaning of their codewords***

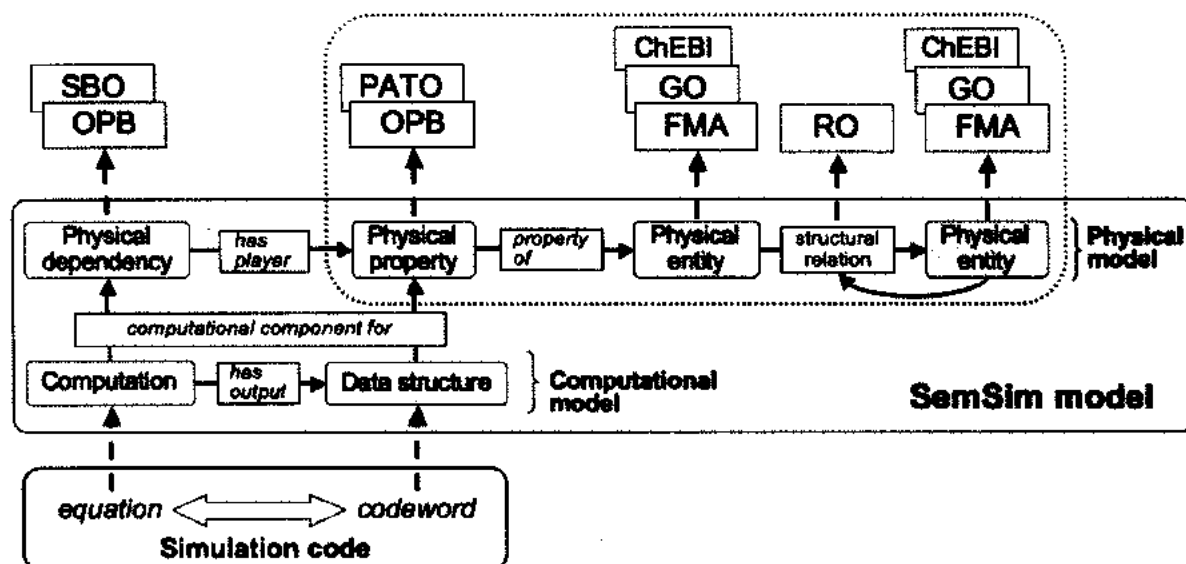
Petty et al. [60] define semantic interoperability as “a question of whether the models that make up the composed simulation system can be meaningfully composed, i.e., if their combined computation is semantically valid. It is possible that two components may be syntactically linked, so that one can pass data to the other, but semantically invalid, if the data produced by the first component is not within the bounds of what the other can validly accept.”

Biosimulation modelers must ensure that interfaces between two component models are semantically valid. For example, one would not want capillary blood volume used as an input

pressure to a baroreceptor model, because volume and pressure represent two disjoint physical concepts. This issue is especially problematic when users attempt to compose models developed by other authors and within other research domains. Aside from whatever documentation is provided with a model, its semantics often remain hidden, implicit or embedded in cryptic annotations. As I know from personal experience, this opacity leads to costly modeling errors and necessitates direct communication with model authors during model composition and decomposition tasks. Semantic interoperability ensures that component models are linked through biologically meaningful interfaces; thereby reducing modeling errors and helping modelers grasp a model's semantics and assumptions quickly and easily. Because SemSim models are semantically interoperable, a computer can automatically recognize the interface points between component models, and I demonstrate this capability in Chapter 6. Furthermore, users can perform sophisticated semantic searches on SemSim model repositories, and employ ontology-based computational reasoning engines to further automate model construction and parameterization.

Currently, semantic interoperability is regarded as a difficult problem in the field of component-based software engineering [47, 61]. The challenge of semantic interoperability in biosimulation lies in making the meaning of model codewords and equations machine-readable so that during composition, a computer can automatically recognize semantically valid interfaces between models. To meet this challenge the SemSim framework forms machine-readable definitions of model codewords and equations using URI pointers to classes in web-accessible reference ontologies. These ontology classes provide a set of standardized, unambiguous, machine-readable terms for model annotation. Figure 3 provides a diagrammatic introduction to the SemSim framework, and shows how model codewords link to semantic concepts in reference ontologies. As shown in the bottom of the figure, the SemSim framework captures a biosimulation model's codewords and equations as members of the SemSim *Data structure* and *Computation* classes. These classes are part of the SemSim computational model, which stores information about a simulation model's computational aspects. Individual members within the *Data structure* and *Computation* classes link to corresponding members in the SemSim *Physical property* and *Physical*

*dependency* classes. These classes are part of the SemSim physical model, which stores semantic information about codewords and equations. Individual members within the *Physical property* and *Physical dependency* classes are annotated using concepts from reference ontologies (blue boxes atop Figure 3). Annotations for *Physical dependency* individuals are singular: they refer to only one concept in external reference ontologies that contain quantitative dependency information, such as the Ontology of Physics for Biology (OPB – described in section 4.3.2) or the Systems Biology Ontology (SBO). However, while users can annotate model codewords using singular annotations, current reference ontologies do not provide a rich set of pre-composed terms for such annotations. Therefore, fully disambiguating the meaning of codewords more often requires the expressive capabilities of *composite* annotations (Figure 3, dotted border), which link *Physical property* and *Physical entity* terms via standardized semantic relationships. I discuss composite annotations for model codewords further in section 4.4 after introducing the main reference ontologies used for my project.



**Figure 3. The SemSim architecture.** Computational aspects of an existing simulation model (bottom) are captured within an OWL-based SemSim model (center) and annotated with machine-readable semantics from reference ontologies (top). The dotted border delineates the logical architecture of composite annotations for model codewords. Abbreviations: ChEBI - Chemical Entities of Biological Interest), FMA - Foundational Model of Anatomy, GO - Gene Ontology, OPB - Ontology of Physics for Biology, PATO - Phenotype and Trait Ontology, RO - Relations Ontology, SBO - Systems Biology Ontology.

Reference ontologies required for annotating model equations are less numerous and less mature than those that provide concepts for annotating codewords. Although the SBO provides such annotations for chemical network modeling (“Henri-Michaelis-Menten rate law,” for example), I required a more comprehensive set of multi-scale, multi-domain dependency terms to thoroughly annotate the equations contained in the models used for my research. Therefore, I did not annotate SemSim model equations as part of my dissertation work, only codewords. Additionally, some model codewords represent the physical properties of physical *processes* (heart rate, for example), and while the SemSim developers eventually plan to make it possible to add process information to SemSim models, this will require a level of computable, multi-scale process knowledge that is also unavailable.

Therefore, I did not develop tools for incorporating processes within composite annotations for my project, and in Chapter 8 I discuss the consequences of these limitations. Despite these constraints, I was ultimately successful in performing a set of model composition and decomposition tasks that demonstrate the utility of the SemSim architecture (Chapter 6), and the success of this demonstration indicates the utility and primary importance of property-entity codeword annotations within the context of semantic composition.

### ***4.3 Reference ontologies for annotating SemSim models***

As machine-readable collections of biomedical knowledge, biomedical reference ontologies provide a wealth of concepts for use in annotating SemSim models. The SemSim framework accommodates biosimulation models across physical scales precisely because numerous ontologies of biological structure already exist. Taken together, these structural ontologies represent physical entities at multiple physical levels within an organism. For example, the Foundational Model of Anatomy (FMA) [62] represents organs, tissues and cells, and the Chemical Entities of Biological Interest (ChEBI) ontology [63] represents molecules. Orthogonally, the SemSim framework accommodates biosimulation models across multiple physical *domains* through the use of the Ontology of Physics for Biology (OPB), which contains a rich set of physical principles applicable across biosimulation modeling domains.

These ontologies provide a rich source of logically defined biophysical concepts for use in annotating SemSim models. They are freely available online in various knowledge representation formats, including OWL, and many are accessible through the National Center for Biomedical Ontology's (NCBO) "BioPortal" website. Therefore, these ontologies can be readily queried, and are programmatically accessible for annotation purposes. In the following sections I discuss in more detail the three main ontologies used for my project: the FMA, OPB and ChEBI. While these ontologies contain nearly all of the annotation material needed for my own project, there are scores of other ontologies available that can meet the annotation needs of SemSim modelers working in other modeling domains.

### 4.3.1 The Foundational Model of Anatomy

The FMA is a large reference ontology that represents the structural entities of the canonical human body and the relationships between these entities [62]. Dr. Cornelius Rosse and Jose Mejino, M.D. developed the ontology, which presently contains 80,000 classes connected by 2.5 million relationships. The FMA's creators did not design the ontology to meet the needs of any specific user group or to provide content for any specific computer application. Rather, they conceived the FMA as a highly principled ontology that provides a "foundational" set of computable, anatomic knowledge that enforces a strict set of rules in categorizing and defining anatomical terms. In this way the FMA differs from other biomedical ontologies like the Gene Ontology (GO), which have more utilitarian, consensus-driven designs.

Since the FMA became available for licensed use, it has been widely adopted as a standard reference ontology. The Virtual Soldier [24], Virtual Physiological Human (VPH) [64] and Terminologia Anatomica [65] Projects all adopted the FMA as part of their ontology framework, and the FMA has served as an ontological template for the development of the Common Anatomy Reference Ontology (CARO) [66], the Disease Ontology (DO) [67], the Human Phenotype Ontology (HPO) [68] and the Cell Ontology (CL) [69], among others. The FMA has also been used to support computer applications like BodyParts3D (Database Center for Life Science, Japan) and Anatomy Lens (IBM).

As one of the largest and most principled biomedical ontologies in existence, the FMA is a critical element in the SemSim architecture. The thoroughness and multi-scale nature of the FMA make it well suited for biosimulation model codeword annotation, and I relied heavily on FMA terms when annotating the codewords in the cardiovascular, baroreceptor and vascular smooth muscle models used in my final demonstration of modular modeling. Additionally, the sheer size of the FMA has prompted the creation of tools that make information retrieval from large biomedical ontologies more efficient. As part of Dr. Jim Brinkley's research on increasing the utility of the FMA, members of his lab built a web service for quickly querying large reference ontologies like the FMA, ChEBI and the



National Cancer Institute (NCI) Thesaurus [70]. This web service became an integral part of SemGen's Annotator tool, which I built for annotating SemSim models. I discuss this important web service further in section 5.1.1.1.

#### **4.3.2 The Ontology of Physics for Biology**

Driven by the use case of biosimulation model annotation, Dr. Daniel Cook developed the OPB as a means of formally representing the physical principles involved in biological sciences [71]. Based on classical physics, systems dynamics and network thermodynamics, and built according to the ontological principles of the Open Biomedical Ontologies (OBO) [72], the OPB represents physics concepts from various kinetic modeling domains (chemical, fluid, electrical, etc.) that can be used in annotating SemSim models. Whereas structural ontologies like the FMA and ChEBI represent physical entities like blood and glucose, the OPB represents the physical *properties* these entities can possess, such as fluid pressure and chemical amount. Because biosimulation model codewords represent physical *properties*, the OPB is vital to the SemSim architecture in order to completely specify the biological meaning of SemSim model codewords.

The OPB organizes properties within seven different kinetic domains: chemical kinetics, diffusion kinetics, electrochemistry, fluid kinetics, heat kinetics, field potentials, and solid kinetics. Thus, the OPB provides annotation content for biosimulation models across a wide range of modeling domains – from molecular pathway networks to macroscopic solid dynamics models. While the abundance of structural ontologies makes the SemSim framework multi-scale, the content of the OPB makes the framework multi-domain as well. As an indicator of the OPB's utility, the VPH project coordinators recently adopted the ontology as one of their standard reference sources.

Although Dr. Cook has been the primary architect of the OPB, my own research has helped test and de-bug the ontology throughout its development. Throughout my dissertation work, I collaborated closely with Dr. Cook to ensure that the OPB contained all the relevant classes necessary to annotate the models used in my demonstrations of modular modeling. I used an

OPB physical property annotation as part of every composite annotation created for these demonstrations, testifying to the OPB's utility and expressivity for annotating biosimulation concepts.

### 4.3.3 The Chemical Entities of Biological Interest Ontology

Researchers at the European Bioinformatics Institute (EBI, Cambridge, UK) created the freely available ChEBI ontology in order to provide structured definitions of “smaller,” chemicals that either occur naturally in organisms or are synthetically produced to intervene in organismal processes. ChEBI is orthogonal to other knowledge resources that focus on macromolecular biopolymers (DNA and proteins, e.g.) such as the EMBL Nucleotide Sequence Database, UniProt and GO. It is also orthogonal to the FMA, which does not represent entities below the sub-cellular level. ChEBI's creators wanted “to provide a high quality, thoroughly annotated controlled vocabulary to promote the correct and consistent use of unambiguous biochemical terminology throughout the molecular biology databases at the EBI.” [63] They built ChEBI out of the need for a well-defined collection of molecular information that extends to more fundamental chemical levels than extant knowledge resources. The molecular entities represented in ChEBI include atoms, molecules, ions, ion pairs, radicals, radical ions, complexes, and conformers, among others.

I made heavy use of ChEBI in my project because it provided nearly all the physical entity annotations required for the second arm of my final modular modeling demonstration. For this part of the demonstration I converted two SBML models of carbon metabolism into the SemSim format and annotated their codewords so the models would be semantically interoperable. The chemical entities that participate in these metabolic models are all relatively small, non-polymerized biomolecules, and they all have corresponding entries in the ChEBI ontology.

## 4.4 Composite annotations

The goal of multi-scale, dynamic biosimulation models is to simulate the physical *properties* possessed by the physical entities that participate in a biological system. For example, a

cardiovascular model does not simply simulate aortic blood, it simulates the *fluid pressure* or *fluid volume* of aortic blood. A metabolic SBML model does not simply simulate glucose, it simulates the *chemical concentration* of glucose. Therefore, the SemSim framework requires physical property information in order to fully capture the meaning of biosimulation model codewords. There are two approaches to this annotation challenge: pre-coordinate annotation terms or post-coordinate them using existing knowledge sources. Pre-coordination requires creating and hand-curating a knowledge repository that comprehensively pre-composes the entire set of merged physical property-physical entity combinations within biology (for example, “fluid pressure of aortic blood”). This presents an intractable, non-scalable annotation problem because the contents of the repository would include the cross product of all terms within large ontologies such as the FMA and ChEBI with multiple property terms from the OPB. As a scalable alternative, the SemSim composite annotation framework provides a flexible, post-coordination approach allowing annotators to create complex definitions of model codewords by linking singular concepts from existing reference ontologies.

Composite annotations consist of a directed list of terms from multiple ontologies that are linked via standardized relationships (Figure 3, dotted border). Because biosimulation model codewords simulate physical properties, the directed list always begins with the physical property represented by the codeword. This property then links to a physical entity via a *physical property of* relationship. This physical entity may in turn be related to other physical entities via structural relations such as *contained in* or *part of* from the Relations Ontology (RO) [73].

To demonstrate the need for our composite annotation scheme, and to show why it is impractical to pre-coordinate all property-entity cross products in advance, consider the codeword “PSysVeins” from the cardiovascular model described by Gennari et al. (CV – [55]) which represents the pressure of blood in the systemic veins. The physical property annotation for this codeword is straightforward – the codeword simulates an *OPB:Fluid pressure*. However, the physical entity annotation is more complicated. While the FMA does

define a class *Portion of blood* that subsumes a set of blood portions throughout the circulation (*Blood in aorta*, for example), it does not define blood in the systemic venous system. The pre-coordination solution to this problem would be to pre-compose the term *Blood in systemic veins*, add it to the FMA, and continue to do so for all circulatory compartments and sets of compartments that contain blood. However, defining all the portions of fluid that can be contained in all the compartments (or compartment sets) of the human body presents an intractable recombination of physical entities. Alternatively, with the necessary programmatic tools, annotators can create these kinds of complex anatomical terms from existing terms in a post-coordinated fashion. As another example, a researcher annotating an MRI image might require the concept *Urine in the ureter* to annotate an image region. While this concept is not in the FMA, the researcher could create it using a composite annotation that links *FMA:Portion of urine* to *FMA:Lumen of ureter* via an *RO:contained in* relation. In my final demonstration of modular modeling, I used the SemSim composite annotation scheme in this manner to annotate the term for blood in the systemic veins mentioned above. I composed the annotation for “PSysVeins” as *OPB:Fluid pressure <physical property of> FMA:Portion of blood <RO:contained in> FMA:Lumen of systemic venous tree organ*. I created similar composite annotations for defining all the properties of blood within the eight circulatory compartments of the CV model.

The SemSim composite annotation scheme is a scalable framework capable of generating very precise definitions for a wide variety of biosimulation model terms. The scheme harmonizes nicely with the Open Biomedical Ontology (OBO) Foundry’s vision of interoperable, orthogonal reference ontologies. In this vision, reference ontologies are scoped so their content does not overlap (so ontologists avoid the burdens of mapping equivalent ontology terms) and new terms are created using existing terms from multiple ontologies. The SemSim composite annotation scheme and its associated tools are the first of their kind to provide a means of linking terms in orthogonal ontologies for the purposes of model annotation. Working independently, Gkoutos et al. [74, 75] developed a similar, convergent composite annotation approach for annotating phenotype data used in genetic research. As in the SemSim framework, their composite annotations consist of a structural entity and the

physical property – or “quality,” in their terms – possessed by the entity. They created the Phenotype and Trait Ontology (PATO) [74, 75] as a collection of qualities for use in their composite annotations, just as the SBP group created the Ontology of Physics for Biology to provide physical property annotations. The SBP and the PATO groups are currently collaborating as part of the European Commission’s Virtual Physiological Human Project, to harmonize the two composite annotation approaches and explore the open issue of how to store composite annotations for reuse (see section 4.4).

Although the information contained in reference ontologies is vast, annotators should not be limited to reference ontology terms when creating composite annotation components. If they need to create custom physical entity terms to define, say, a pathological feature like an atrial-septal defect, annotators can do so. Just as with physical entity terms from reference ontologies, modelers can link physical property or physical entity annotations to these custom entity terms. To annotate a model codeword that simulates the flow of blood through an atrial-septal defect, an annotator might create the composite annotation *OPB:Fluid flow* <physical property of> *FMA:Portion of blood* <RO:contained in> *Atrial-septal defect*. In this example the annotator creates the *Atrial-septal defect* term as a custom physical entity. This term only exists within a SemSim model and does not refer directly to any outside reference ontology concept. The SemSim developers recommend, however, that annotators link their custom entity annotations to one or more reference ontology concepts via some standardized ontological relationship to (such as *RO:contained in*, *RO:part of*, etc.). This helps to better define the entity logically, allowing computer programs some means of determining its semantic relationship with other entities in the SemSim model. Assessing these relationships programmatically for further automating SemSim model composition and decomposition tasks will be important part of SemGen’s future evolution (see section 8.1.3).

#### ***4.5 The SemSim model template***

The SemSim model template is an implementation of the SemSim architecture in Figure 3. The template is currently implemented as a Web Ontology Language (OWL) [76] file that provides the general class structure and relationships for capturing all the computational and

semantic information of a specific biosimulation model. When SemGen imports an existing biosimulation model into the SemSim format, it adds OWL individuals to this template to represent specific model data structures and computations. When using SemGen to annotate a SemSim model's codewords, SemGen adds OWL individuals to the *Physical entity*, *Physical property* and *Property dependency* classes within the SemSim template.

Suppose "Paorta" is a codeword of data type decimal number that simulates aortic blood pressure in a cardiovascular model. To capture this knowledge in the SemSim format, SemGen adds a new individual member of the *Decimal* class (a subclass of *Data structure*) called "Paorta" to the SemSim template along with a new *Computation* individual that describes how the codeword "Paorta" is solved in the model (refer to Figure 3). The user then annotates the "Paorta" individual, either in a composite or non-composite manner (depending on user preference and annotation availability), using one or more references to classes in external ontologies. During annotation, SemGen creates new individuals within the *Physical entity* and *Physical property* classes as needed in order to store the user's annotations. Each individual annotation component added to the SemSim model contains a URI pointer to an external ontology class, and together they form "Paorta's" machine-readable definition.

#### ***4.6 The SemSim approach to interfaces between models***

Because there is no way to anticipate how modelers will merge individual SemSim models, the SemSim architecture does not specify the interface(s) that one model should provide to others ahead of time. Instead, each annotated codeword in a SemSim model acts as a potential interface point with other SemSim models. In this way, the SemSim approach to module interfaces differs slightly from more familiar examples of modular components used in, for example, the automobile, aircraft and electronics industries.

To illustrate this point, consider the following examples of SemSim model composition. A researcher might combine a cardiovascular model with a lung model to create a cardiopulmonary system. These models might have an interface involving the dynamics of the pulmonary circulation, or the pleural cavity. Another modeler might take that same

cardiovascular model and merge it with a model of gas exchange in systemic capillaries. In this case the interface between the models would involve systemic capillary blood dynamics. By not specifying the interfaces between SemSim models *a priori*, SemSim allows this kind of flexibility in model composition.

In the automobile, aircraft and electronics industries, designers specify the interfaces between product components ahead of time so that different production teams can focus on a particular component, yet remain assured it will interface properly with others no matter how much they change their component's internal structure. Production teams maintain the efficiency inherent in a separation of concerns by keeping the number of interfaces between components specific and small relative to the internal connections within the individual components. In the SemSim architecture, every model codeword becomes a potential interface point once it is annotated, and thus, the model can be interfaced with others in a variety of ways. We believe this approach accommodates the existing separation of concerns in biological modeling, and simultaneously provides the level of interoperability needed to encourage broad model reuse. Although SemSim models have a high number of *potential* interface points, they are still modular in a general sense because the computational interfaces established between merged models remain small relative to the merged system's internal computational network, unless one merges extremely similar models.

For modular components that have specific interfaces, it is useful to delineate the inputs to and outputs from the component. Biosimulation models often specify their computational inputs and outputs as well, and one may be tempted to think that SemSim models only interface according to these input-output specifications. While the computational, or syntactic, aspects of SemSim models preserve this input-output structure, there are no inputs or outputs specified at the semantic level, only equivalencies or non-equivalencies. Therefore, because they are semantically interoperable, SemSim models interoperate outside the constraints of the input-output structure of the computational model. Coupling between merged SemSim models occurs because the models contain codewords that are semantically equivalent, a condition independent of any input-output designation attached to the codeword

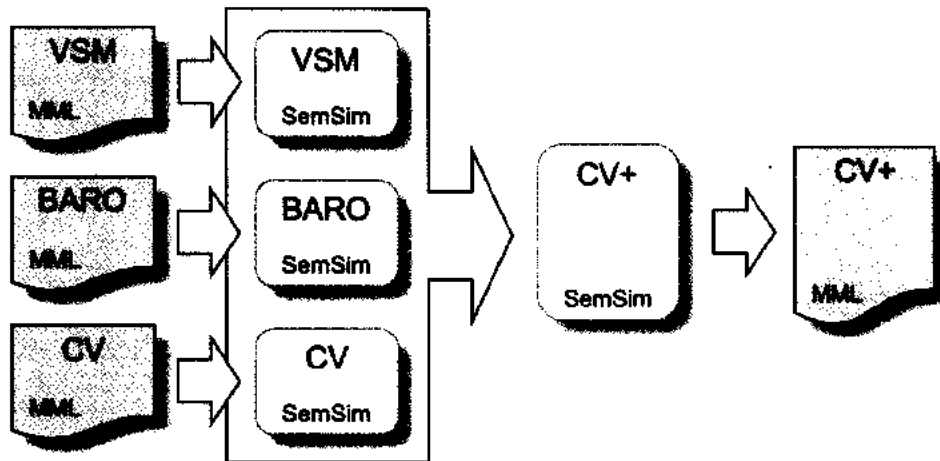
at the computational level.

#### ***4.7 Preliminary work validating the SemSim approach***

Prior to developing any automated annotation or merging tools for SemSim modeling, my colleagues and I performed two initial tests of the SemSim architecture. In the first test we demonstrated that three existing, independently-coded Mathematical Modeling Language (MML) models related to the cardiovascular system could be translated into the SemSim format (previously called the “Application Model Ontology” format), then merged in a semi-automated, modular way to produce a new multi-scale cardiovascular model (Figure 4). We performed this demonstration to test the theory behind our merging approach and to anticipate the programmatic features needed for constructing the automated annotation and merging tools that I built for my dissertation work (see Chapter 5).

The three models used in this demonstration simulated 1) hemodynamics in an eight-compartment, closed-loop cardiovascular system (CV), 2) the control of blood pressure via baroreceptor feedback (BARO), and 3) calcium dynamics in a vascular smooth muscle cell and their impact on arteriolar resistance (VSM). I coded the first model, Dr. Daniel Beard coded the second and Drs. Daniel Cook and Brian Carlson collaborated to produce the third.





**Figure 4. Initial merging test using the SemSim architecture. The SemSim team first converted MML models of the cardiovascular system (CV), baroreceptor control of blood pressure (BARO) and calcium dynamics in a vascular smooth muscle cell (VSM) into the SemSim format. Then, we merged the SemSim models into a new multi-scale cardiovascular model (CV+) using the Prompt ontology mapping tool as a guide.**

To merge these models into a single multi-scale system we first translated each into the SemSim model format. We used the Protégé knowledge-editing environment to manually encode all the computational and semantic information in the original MML files, whether explicit or implicitly stated, needed for merging. We manually added semantic data to the Protégé files that defined the meaning of all model codewords representing physical properties. For these codewords we formed an annotation pair comprised of one reference to a structural entity from the FMA and one reference to a physical property from the OPB - a primitive implementation of composite annotations that we later expanded. In this case, we retrieved FMA annotation URIs either via the browser-based FMA explorer tool [77] or via direct communication with Jose Mejino, the FMA's curator. Dr. Cook provided us with the relevant OPB terms directly.

We used the Protégé plug-in "Prompt" [78] to analyze the points of semantic overlap between the three hand-encoded models. Prompt is a general-purpose ontology-comparison tool used for merging, mapping and aligning ontologies. When merging models, the Prompt

interface displays the suggested mappings between them in a central column within the interface. The class hierarchy of one model flanks the central column on the left; the hierarchy of the other model flanks the right, and red and blue color-coding distinguishes one ontology's classes from the other. Inspired by the Prompt interface, I adopted several of these visual elements when creating my more focused tool for merging SemSim models (see section 5.3).

We tailored our experiment so that Prompt identified semantic equivalencies across SemSim models based on the FMA-OPB duples that defined the codewords. Using the Prompt analysis as a guide, we manually merged the three test models together into a single, functional, multi-scale MML model (CV+). This work demonstrated that the SemSim architecture provides a sufficient set of logical constructs needed for semantics-based composition of MML models.

In the SemSim team's second major test of the SemSim architecture, we attempted to extract the systemic arterial component from an externally-developed cardiovascular model coded in MATLAB [39] and use it in place of the semantically equivalent portion of our own CV model. Whereas the three original models in the first test were all coded in MML, we wanted to test the architecture by merging models coded in different languages. We showed that the extracted MATLAB model could be cast in the SemSim format (again, using the Protégé frames implementation), and then semi-automatically recombined with our existing CV model to produce a new model that gives a more realistic arterial pressure curve.

Whereas procedural languages like MATLAB specify the steps computers must take to reach a desired outcome, including calls to subroutines and ODE solvers, declarative languages like SemSim, MML, SBML, and CellML simply specify the simulation without specifying a procedure for solution. This distinction resulted in a number of major challenges during the MATLAB-to-SemSim translation:

- Models may solve equations and update variables at more than one time scale.
- Data structures may not translate from one simulation environment to another.
- Piece-wise computation and variable passing through functions may obscure the dependencies in procedural code that are difficult, if not impossible, to represent in declarative code.

This study suggested that automatically generating interoperable SemSim models from existing models would be easier if the latter were coded in a declarative language. While procedural languages allow modelers more precise control over the programmatic tasks needed for their modeling studies, this can lead to significant challenges in terms of model interoperability and reuse. Declarative modeling formats, although newer and less programmatically flexible, emphasize interoperability and reuse, but sometimes at the expense of model customization.

Besides demonstrating the feasibility of using the SemSim architecture for biosimulation composition, both of the studies discussed here also helped my colleagues and I identify some common steps needed to resolve and merge two models. These steps, which include allowing the user to equate two semantically different codewords and to encode time-dependent variables as static parameters, helped me identify some of the programmatic tasks needed to automate model merging as much as possible within SemGen.

Once the SemSim group demonstrated that the SemSim architecture was a feasible solution for modular biosimulation composition, the next logical step was to implement software tools to automate the creation, composition, decomposition, and recoding of SemSim models. My dissertation research represents this next step in SemSim research. It is only with such tools that modelers will be able to realize the timesaving benefits of modeling within the SemSim framework, since creating and composing SemSim models by hand with existing knowledge representation tools is cumbersome. In Chapter 5 I discuss these tools in detail and demonstrate their utility in Chapters 6 and 7.

## Chapter 5: The SemGen tool for semantic composition of biosimulation models

In this chapter I detail my implementation of SemGen and discuss its features for creating, decomposing, merging and encoding SemSim models. I discuss the four main components of SemGen: the *Annotator*, which imports and annotates existing biosimulation models, making them into semantically interoperable SemSim models; the *Extractor*, which provides several methods for decomposing a SemSim model into interoperable sub-models; the *Merger*, which helps automate the merging of SemSim models; and the *Coder*, which automatically converts a SemSim model into executable code for numerical simulation.

I have coded SemGen in Java in order to leverage several important Java-based APIs that are crucial to the tool's function and to avoid recoding the program for different operating systems. Because SemSim models are implemented as OWL ontologies, I use the OWL API [79], created by the OWL developers at the University of Manchester, to manage, read, and edit SemSim models within SemGen. I also use the JDOM API [80] for XML parsing during the MML-to-SemSim translation process (see next section), the prefuse API [81] for visualizing the computational networks of SemSim models (section 5.2.1), and the Java Universal Network Graphing (JUNG) API [82] for identifying and visualizing computational clusters (section 5.2.4) within a SemSim model. I also utilize a set of Java classes developed by Dr. Jim Brinkley's research team for accessing their vSPARQL web service [83](section 5.1.1.1).

### 5.1 Annotator

The purpose of the Annotator tool is to 1) automatically convert an existing, compilable biosimulation model into the SemSim format and 2) provide an interface for annotating SemSim model codewords with composite, singular and human-readable annotations. The Annotator will eventually be capable of annotating model equations as well as codewords, but due to the lack of multi-scale physical dependency ontologies, it only provides the latter

function presently (see section 4.2). Converting and annotating existing biosimulation models is the first necessary step in turning them into semantically interoperable SemSim models that can be decomposed into smaller sub-models using the Extractor and/or recombined with other SemSim models using the Merger.

As I discussed in section 4.1, my colleagues and I developed the SemSim architecture so that it can accommodate the various modeling languages used by the biosimulation community. Ultimately the Annotator tool will include parsing algorithms for many simulation languages so they can be readily converted into the SemSim format. However, the focus of my dissertation was not to develop these parsing methods, but rather to demonstrate a successful implementation of a modular, multi-scale, multi-domain modeling system. Therefore, I developed a parser for a single language, the Mathematical Modeling Language (MML) developed for the JSim simulation environment, which would allow me to perform this demonstration.

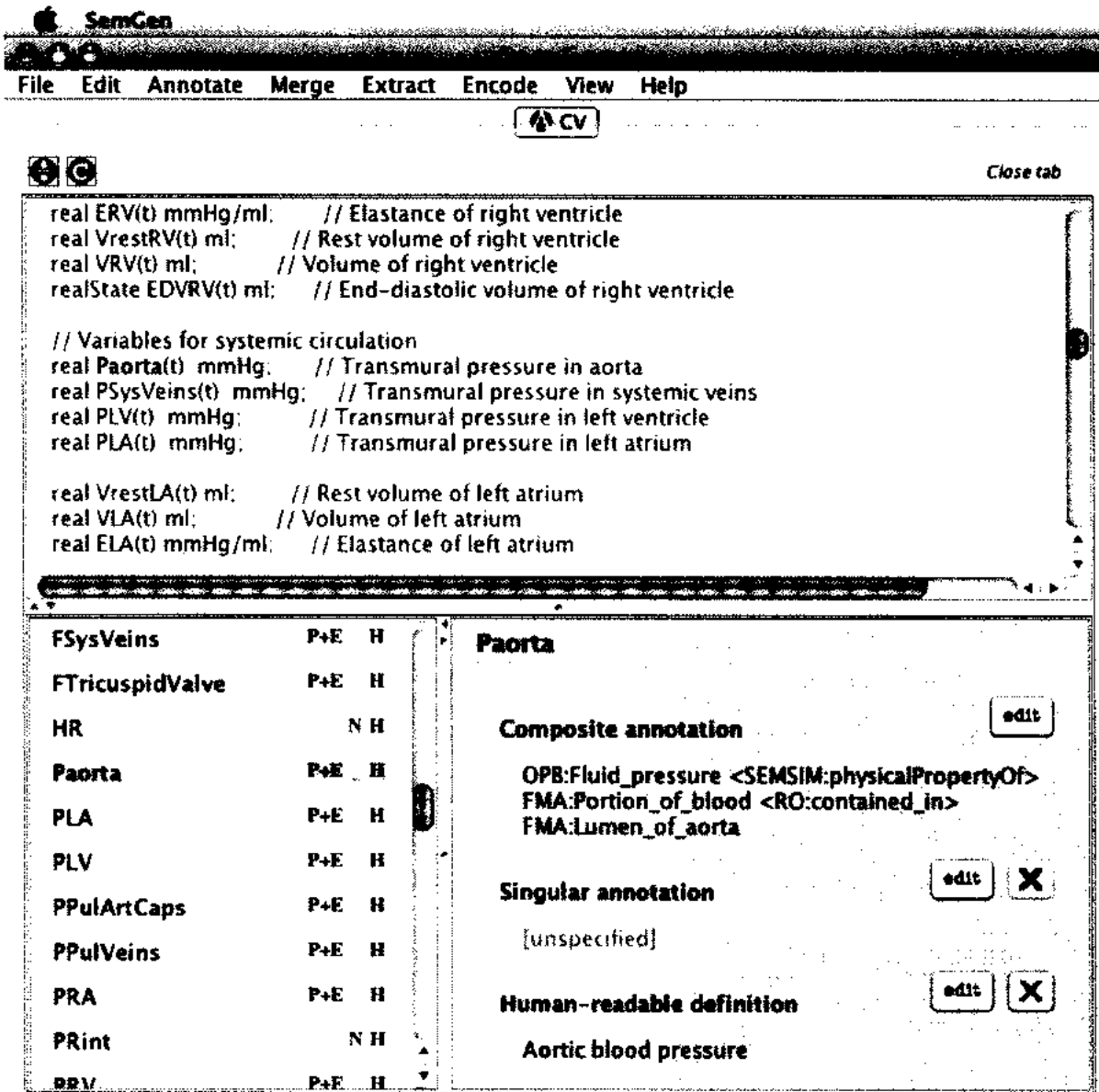
MML is a freely available, general purpose, multi-scale, multi-domain modeling language developed by Dr. Jim Bassingthwaight's research group at the University of Washington. I chose to use MML models for my research project because Dr. Bassingthwaight is one of the main developers and promoters of the Physiome vision, and thus MML supports modeling at all levels of biological organization and within all physical domains. MML models have a declarative representation and are separated from the numerical solvers and visualization tools used at model runtime. This facilitates the MML-to-SemSim translation by avoiding the cumbersome tasks associated with parsing a procedural modeling language like MATLAB [37] (following nested function calls, untangling solver and data visualization tools from the model representation, etc. – see section 4.7). Additionally, the JSim developers created tools that automatically translate models coded in other declarative modeling languages such as SBML and CellML into the MML format. Therefore, many SBML and CellML models can be readily converted into the SemSim format using MML as an interlocutor.

Once a user opens an existing MML model for annotation, the Annotator uses a command function built into the JSim software called “jsbatch” to parse the model into an XML format called XMML (eXtensible Mathematical Markup Language). This XMML format provides a declarative representation of the model codewords, their units and their mathematical relationships. The Annotator then uses functions and classes provided in the JDOM and OWL APIs to parse the XMML representation of the original model and instantiate the model codewords, units and mathematical dependencies (the *computational* aspects of the model) as individuals within the classes of the OWL-based SemSim template file. The XMML representation also specifies which codewords are needed as inputs to solve other codewords, and the Annotator represents these computational dependencies via *has input* and *is input for* relationships between codewords in the SemSim model.

During the MML-to-SemSim conversion process the Annotator also examines the physical unit declarations of all codewords in order to automatically add semantic information to the new SemSim model. The physical units of codewords map to classes in the OPB (for example, millimeter of mercury maps to *OPB:Fluid pressure*). Thus, in order to streamline the semantic annotation process as much as possible, the Annotator automatically adds a physical property annotation for those codewords that map to only one OPB class. This spares the user the task of annotating many of the physical properties in the model. In Chapter 6 I provide concrete examples of this annotation process, including the automatic addition of physical property annotations.

Once the automatic conversion process completes, the Annotator saves the new SemSim file to a user-specified location. The Annotator then places the text of the original model in a scroll pane at the top of the SemGen interface and lists the model’s codewords at the bottom left (Figure 5). The three columns to the right of the codeword indicate the presence of the composite annotation, non-composite (singular) annotation, and the human-readable annotation. A key that explains the meaning of the characters that occupy these slots is below:

- **P**: Physical property present in composite annotation
- **P+E**: Physical property and physical entity (or entities) present in composite annotation
- **N**: Non-composite annotation present
- **H**: Human-readable annotation present
- **\_**: Annotation not present



**Figure 5. The SemGen Annotator tool. Main interface showing the annotations for the codeword “Paorta” from the cardiovascular model described in Gennari et al. [55]**

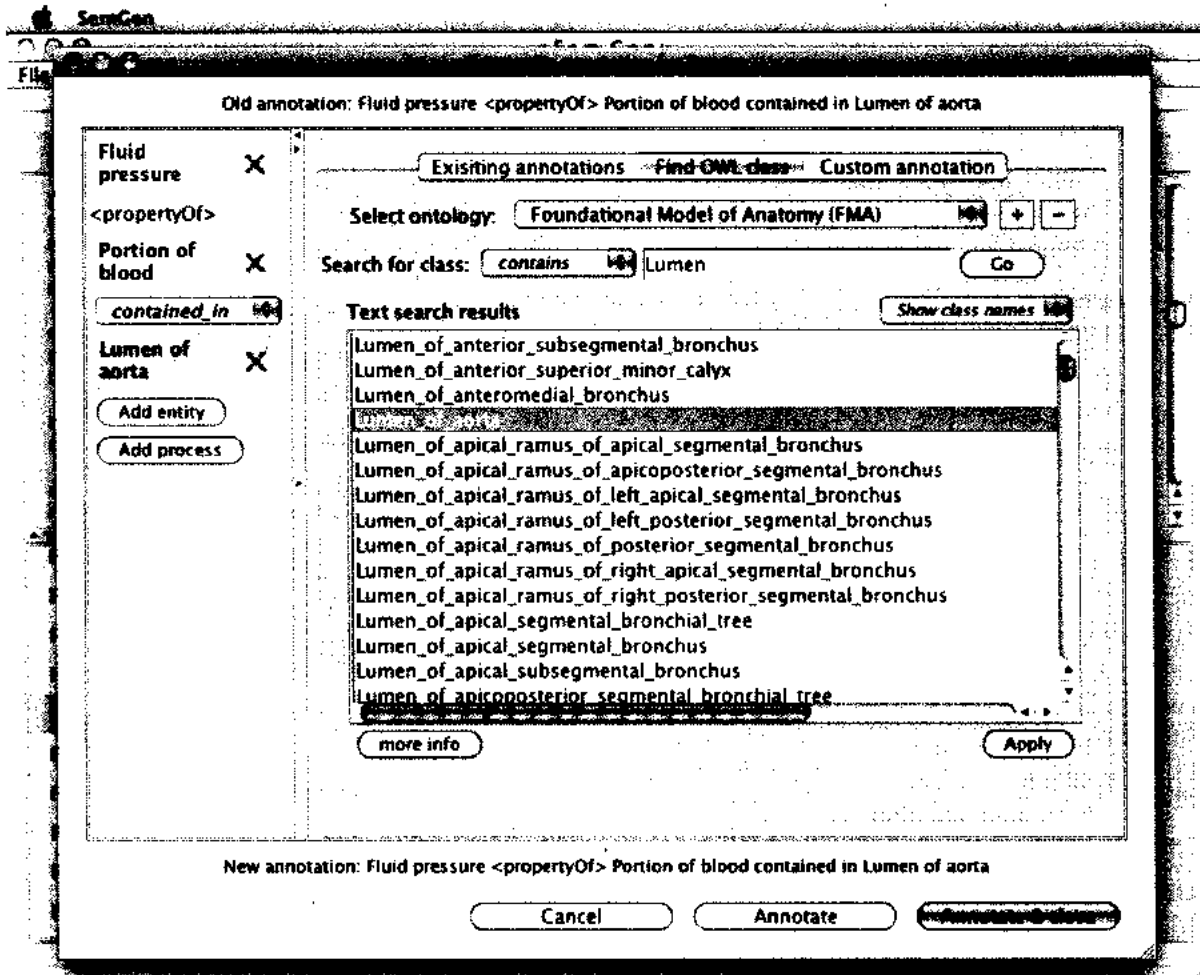
When the user selects a codeword from the list, SemGen displays all the semantic annotations for that codeword in the lower right panel. By clicking on the “edit” buttons, the user can add or change the existing annotations.



In order for a SemSim model to become semantically interoperable with other SemSim models, it must be annotated using machine-readable codeword definitions. Not every codeword must be annotated for a SemSim model to be useful, but the more annotations the user provides, the more interoperable the model becomes. Thus, the annotation process is an extremely important aspect of the SemSim methodology. In the next three sections I detail the Annotator's features that facilitate this process.

### **5.1.1 Specifying composite annotations**

As I described in Chapter 4, composite annotations provide a highly expressive means of creating unambiguous, machine-readable definitions of model codewords. In all five models used for my dissertation research, I captured each model's semantics primarily using composite annotations. As mentioned in section 4.4, a composite annotation consists of a directed series of annotation components with a physical property annotation linked to a singular or composite physical entity annotation (Figure 3, Chapter 4). When the user edits a composite annotation, SemGen presents a dialog for constructing the annotation using concepts found in web-based ontologies and/or other SemSim models (Figure 6). The Annotator lists the components of the composite annotation, including the relations between them, along the left side of the composite annotation dialog. Using the tabbed pane in the center of the dialog, the user can edit the codeword's composite annotation using 1) annotation components that are already included in the model, 2) concepts returned from searches over web-based ontologies and 3) custom-created concepts. Given our goal of improving semantic interoperability, the second annotation method is the most crucial, because it provides a means of connecting model codewords to standardized, unambiguous, machine-readable concepts in reference ontologies. These links are what enable SemGen to automatically recognize semantic equivalencies between models. Below I discuss how a user can search for and link model codewords to classes in reference ontologies and reuse annotations from external SemSim models.



**Figure 6.** The “Find OWL class” dialog within SemGen’s composite annotation editor. In this example the composite annotation for “Paorta” has been constructed using an ordered list of singular annotations along the left side of the interface. The individual annotations are connected by standardized ontology relationships. The dialog shows the results from a string search query sent to the Foundational Model of Anatomy.

#### 5.1.1.1 Searching for OWL classes in ontologies

Given the large size of many of biomedical reference ontologies such as the FMA, NCI Thesaurus and ChEBI, requiring model annotators to load local copies of these ontologies onto their machines would strain modelers’ computational resources. Therefore, in order to provide a less memory-intensive method of accessing ontology classes for annotation, the Annotator accesses OWL ontologies remotely using an online querying service. The Annotator queries these ontologies using the vSPARQL web service created by Dr. Jim

Brinkley's research group [83]. Dr. Brinkley's team created this web service, which extends the functionality of the SPARQL query language [84], as part of their research on using reference ontologies as a basis for the semantic web.

To access an OWL ontology remotely, the user selects the ontology they wish to query from a drop down menu in the composite annotation dialog and then enters a search string. By selecting a search type from another drop down menu, users can search for class names that contain the input string, class names that start with the input string, or class names that match the input string exactly. While the first search option provides a more general method for finding reference classes to use as annotations, the latter two streamline string searches and limit the number of returned matches, since results can be numerous given the size and expressivity of larger reference ontologies. When the Annotator searches for a string match it looks at both the IDs and RDF labels of classes within the selected ontology because some reference ontologies such as ChEBI and GO do not use human-readable class IDs. When the query completes, the Annotator lists positive string matches in the "Text search results" panel. The user can further investigate a term selected from this list by clicking the "More info" button below the search results. This feature opens a new window that shows all the properties of the selected term and their values. This is provided so that users can better understand the exact meaning and context of a given ontology term.

By selecting an annotation component on the left of the dialog and then double-clicking one of the search results in the list (or by pressing the "Apply" button), the Annotator adds the reference class to the composite annotation. This reference class then becomes available for reuse within the SemSim model so that users do not have to repeat queries for the same term.

The vSPARQL web service can, provided with enough server memory, automatically retrieve any web accessible OWL ontology for query processing. This approach works best for smaller ontologies, because the ontologies must be loaded into the web service server's internal memory before they are queried. To reduce the time required to query larger ontologies, Dr. Brinkley's group can translate their OWL versions into Jena databases stored

on their servers. They currently have databased versions of the FMA, NCI Thesaurus and ChEBI on their servers. When a user queries one of these larger ontologies using the Annotator, the vSPARQL web service accesses the pre-loaded, databased version. The web service loads smaller ontologies like the Systems Biology Ontology and the OPB locally, as they create a minimal expense in local memory. At no time are any of the queried ontologies loaded on the SemGen user's local machine.

In the future, my colleagues and I plan to avoid the problems associated with loading ontologies into local memory by querying web services provided by the NCBO BioPortal website [85, 86]. BioPortal's creators endeavor to provide a comprehensive repository of biomedical ontologies that can be easily searched and browsed remotely. In the future, SemGen could use the BioPortal querying services and avoid lag time associated with loading ontologies into local memory, since most ontologies of interest will already be stored in a query-ready database and will not require local storage.

If a user wishes to query an OWL ontology that is not provided in the drop down list, they can add it by clicking the "+" button. The user then enters either the URL of an online ontology or locates one on their local machine, and the Annotator adds the ontology to the list. This feature makes the content of all local and online OWL reference ontologies available for annotation purposes. Additionally, because SemSim models are OWL ontologies, they can be queried as well, making their annotations reusable. Currently, only singular annotation components can be reused between SemSim models, not composite annotations. My colleagues and I are presently working on a composite annotation storage framework for the VPH project that will allow annotators to reuse composite annotations as well. I discuss this work further in Chapter 8.

Querying an ontology multiple times for the same term can become tedious, and so SemGen ensures that once an annotation has been used once in a SemSim model, it can be instantly retrieved and used again later. Once a user adds an annotation term to a composite annotation, the term becomes available for internal reuse within the model. In the composite

annotation editor, users can select from these previously-used annotations listed under the “Existing annotations” tab. This list only shows physical property or physical entity annotations, depending on whether the user is attempting to edit a physical property or physical entity annotation.

#### **5.1.1.2 Creating custom annotation classes**

Although there is a wealth of available annotations stored in biomedical ontologies, users will nonetheless sometimes need to create custom annotation components to fully specify the meaning of their model codewords – e.g. to describe a unique kind of equipment (e.g. “Continuous stirred flow tank reactor” from the Nielsen et al. glycolysis model) or a unique grouping of physical entities (“Lumen of systemic arteries and capillaries” from the CV model). The Annotator tool provides a means of creating these custom annotations. Selecting the “Custom annotation” tab in the composite annotation dialog provides a basic interface for creating custom annotations and applying them to the machine-readable definitions of model codewords. Because custom annotations by themselves have no semantic context or logical definition, and are therefore less useful for model reuse and composition, this interface also allows the user to connect their custom annotations to reference ontology classes via standardized relationships. For example, the user can assert that the custom annotation *Lumen of systemic arteries and capillaries*  $\langle RO:contains \rangle$  *FMA:Lumen of systemic arterial tree*. With such links, reasoning tools that compute over SemSim models have a means of detecting the semantic distance between custom annotations and reference ontology annotations. I discuss the future importance of this implementation within the context of model merging in Chapter 8.

#### **5.1.2 Non-composite annotations**

While composite annotations provide a powerful, expressive way to create machine-readable definitions for model codewords, even the cross-product of all available biomedical ontology concepts cannot currently, and may never, provide the necessary components for annotating the semantics of all biosimulation codewords. Sometimes singular annotations fill in the gaps within this cross product. Additionally, users may find it easier to use pre-coordinated,

singular annotations rather than construct composite annotations. Hence, the Annotator provides a simple editor, similar to the composite annotation editor, for annotating a codeword with a singular concept from a reference ontology.

For example, the CV model contains the codeword “PRint” which represents the amount of time between atrial contraction (the P wave in an electrocardiogram) and ventricular contraction (the R wave). Annotating such a term using the composite annotation approach would require physical property concepts that are not yet available in the OPB. Therefore I annotated this codeword using the singular “PR Duration” class found in the Electrocardiogram Ontology [87].

Just as with composite annotations, SemGen examines the non-composite annotations between codewords during model merging. This ensures that SemGen recognizes the semantic equivalency between codewords annotated against the same, singular reference ontology concept.

### **5.1.3 Human-readable annotations**

Because composite and non-composite annotations can be difficult or awkward to read, the Annotator also supports straight text annotation for codewords. By manually entering text in a simple dialog, the user can add a human-readable annotation that is stored as the *rdfs:comment* for the annotated codeword. These annotations are useful in creating more reader-friendly descriptions of codewords as in, for example, block commented reference tables at the end of SemGen-generated MML files. However, SemGen does not currently leverage human-readable annotations when decomposing or merging SemSim models.

## **5.2 Extractor**

The purpose of the Extractor tool is to decompose existing SemSim models into reusable, interoperable sub-models. Modelers will often only be interested in using a *portion* of a previously coded model and will want to generate a module that excludes extraneous computations. For example, when I collaborated with Dr. Roy Kerckhoffs on building a

multi-scale, closed loop mammalian cardiovascular model [30], Dr. Kerckhoffs wanted to couple his finite element heart model to the pulmonary and systemic circulatory components of a full, lumped parameter cardiovascular model that I had coded previously. After sending him my model, Roy spent a significant amount of time extracting out these portions of the circulation by hand, translating them into his simulation language (FORTRAN), and connecting them to his model.

The Extractor facilitates such tasks by providing modelers with a flexible, powerful tool that can help them “carve up” existing models into interoperable pieces. Given the popularity of modularity as a design principle, model builders often conceptualize their models as a set of interconnected, semantically distinct components while constructing them (see, for example, [21, 31, 88]), even in the absence of a modular modeling framework. The Extractor tool allows models constructed in this manner to be readily decomposed according to the model designer’s modular conception. Models built in a non-modular fashion will often be less amenable to crisp, semantically distinct decompositions, so the Extractor tool provides methods that allow the user to precisely delineate the model component they wish to extract. However, models built without a modular design may nonetheless contain “naturally occurring” modular components that arise unintended. The Extractor tool provides methods for identifying and extracting out such components.

There are many ways of decomposing a biosimulation model into modules, and the Extractor currently provides three different of extraction methods that can be used independently or in concert. First, the user can decompose a model by selecting specific physical entities that they want to preserve in the extracted model (e.g. blood in the aorta, glucose-6-phosphate). This method addresses the use case mentioned above, where Dr. Kerckhoffs extracted out the components of the model related to the systemic and pulmonary circulatory systems. Second, users can select specific model input and output variables to preserve in order to fine-tune their extraction at the most detailed level possible. Third, using a network cluster identification algorithm [89], the user can discover and extract out the portions of a model that are most tightly interconnected computationally. This method will be useful to modelers

who are interested in identifying modular portions of simulation code written without a modular architecture in mind. Whereas the first two extraction methods are user-guided, and tailored to meet specific modeling tasks, the third automatically identifies candidate modules for extraction within the parent model. In this regard the third extraction type can act as a module-discovery tool for modelers who may be ambivalent or unsure about how to decompose a parent model into useful components. I discuss these three extraction methods separately in sections 5.2.2 through 5.2.4.

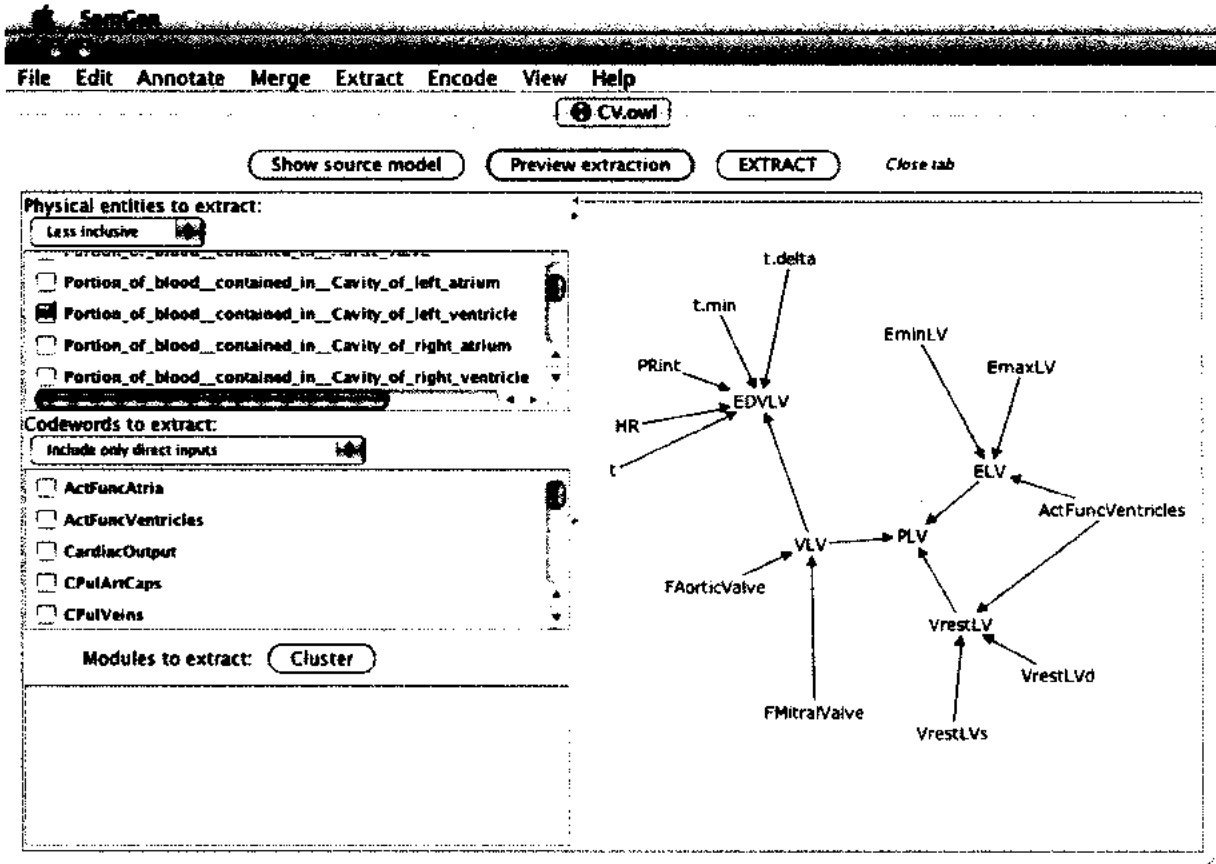
### **5.2.1 Previewing the extracted model with a visualization tool**

The three extraction methods that I will describe can be used individually or in concert to extract a sub-model from a parent SemSim model. However, in order to help ensure that users extract out the portions of the parent model they want, the Extractor provides a means of previewing an extracted model before saving it. Therefore, I implemented a graph visualization tool within the Extractor that shows the computational dependency network to be generated based on the physical entities, codewords and/or computational modules the user elects to preserve (Figure 7). This tool is built on the *prefuse* API [81], which provides classes and functions for general-purpose graph visualization. The vertices in the graphs generated by the Extractor visualization tool represent the individual codewords in a model. A directed edge pointing from one codeword to another indicates that the value of the first is needed to compute the value of the second. The Extractor establishes these relationships based on the computational dependencies asserted in the SemSim model (section 5.1).

I customized and extended the *prefuse* graph visualization tool so that hovering over the names of codewords in the graph displays their semantic annotations and the equations used for their computation. This way a user can familiarize themselves with the meaning and use of a model's codewords. Clicking the "Preview extraction" button displays the computational network of the sub-model to be extracted. The Extractor generates this preview by collecting all the codewords associated with the user's selections in the three extractor type scroll panes and building a graph based on their computational dependencies. For comparison purposes, users can also click the "Show source model" button and visualize the dependency network



of the parent model.



**Figure 7. The SemGen Extractor tool. The left panel provides three methods of decomposition: by physical entity, model codeword, and computational cluster. The right panel displays node and edge graphs of computational dependencies for previewing extracted and source models. Here the Extractor is previewing an extraction of the left ventricular components of the CV model described in Gennari et al. [55]**

### 5.2.2 Extraction by physical entity annotation

When a user loads a SemSim model into the Extractor tool, SemGen identifies and lists the set of physical entities represented in the simulation. Presently, in order to simplify the interface, the Extractor treats composite physical entity terms as single physical entities; it does not decompose composite entity annotations. Thus, it would list the composite term *Portion of blood contained in cavity of left ventricle* as a single entity rather than its

components *Portion of blood* and *Cavity of left ventricle*. With this list, users can select physical entities for inclusion in the extracted model. For example, if a user is interested in extracting out only those components of the cardiovascular model described by Gennari et al. [55] that are specifically related to the left ventricle, they can select those physical entities in the list specific to the left ventricle, then extract out the left ventricular sub-model as a new SemSim model. As shown in Figure 7, the user selects these physical entities from the upper left panel of the Extractor tool. As part of my final end-to-end demonstration of model modularity, I successfully used this extraction method to isolate the chemical species that are involved in the pentose phosphate pathway (PPP) shunt of the metabolic model published by Chassagnole et al.[8] In Chapter 6, I describe the process of extracting the PPP module and merging it with a different metabolic model [7] to produce a novel metabolic system that retains features of both models.

The default setting for the physical entity decomposition method identifies only those codewords that are annotated with the selected entity or entities, and includes them in the extracted model along with all codewords that are direct inputs needed for their calculation. However, the user also has the option to extend the scope of this type of extraction. By changing the drop down box above the list of physical entities to “More inclusive”, the Extractor will include not only those codewords annotated with selected physical entities but also all codewords that require them as inputs to their computation. For example, in the left ventricular extraction example above, the default extraction does not include the term “Vtotal” (total blood volume in the circulation) because it is not annotated using any of the selected physical entities, and is not an input to any codewords that are. In the more inclusive extraction, Vtotal *is* included because it depends on “VLV”, a codeword that is annotated using one of the selected physical entities. In other words, while the “Less inclusive” extraction option provides the user with all model codewords directly related to the selected physical entities, the “More inclusive” option also includes codewords that are computed directly from them.

### 5.2.3 Extraction by specific model codewords

Modelers may wish to preserve specific model codewords in an extracted model. For example, a user may extract physical entities related to the left ventricle (as above), but may need to include the “CardiacOutput” term so they can compute a model-derived cardiac output. Although “CardiacOutput” is closely connected to the dependencies of the left ventricle, it is annotated as blood flow through the aortic valve and is not included in this physical entity-based extraction. To address this need, I created an interface for selecting specific model codewords to preserve in an extraction. Upon loading the source model, the Extractor identifies all model codewords and lists them in the middle scroll pane along the left side of the Extractor interface (Figure 7). Thus, if the user performing the above extraction also selects “CardiacOutput” from the list of codewords to include in the extraction, this term will be included in the resulting extracted model. This provides an example of how the user can combine the Extractor’s extraction methods to build a sub-model up step by step until it contains exactly what they want to extract.

When a user selects a codeword to preserve in the extracted model, they have the option of a) simply preserving the codeword and the direct inputs required for its computation or b) preserving the entire computational dependency tree that is required to compute the codeword. Whereas the former option provides the desired codewords and turns all their computational inputs into user-defined inputs (if they are not already), the latter option recursively follows the full chain of computational dependencies that are required to compute the selected codeword. For highly interconnected models, the latter option may result in an extracted model that differs little from the source. These two output extraction options are available so that the user can more easily tailor the inputs and outputs of the extracted model to their design. The former case, which only preserves the selected outputs and parameterizes its inputs, gives the user more control over the computational dependency network in the extracted model. The latter helps the user preserve the full extent of a codeword’s dependencies.

#### 5.2.4 Extraction by network cluster identification

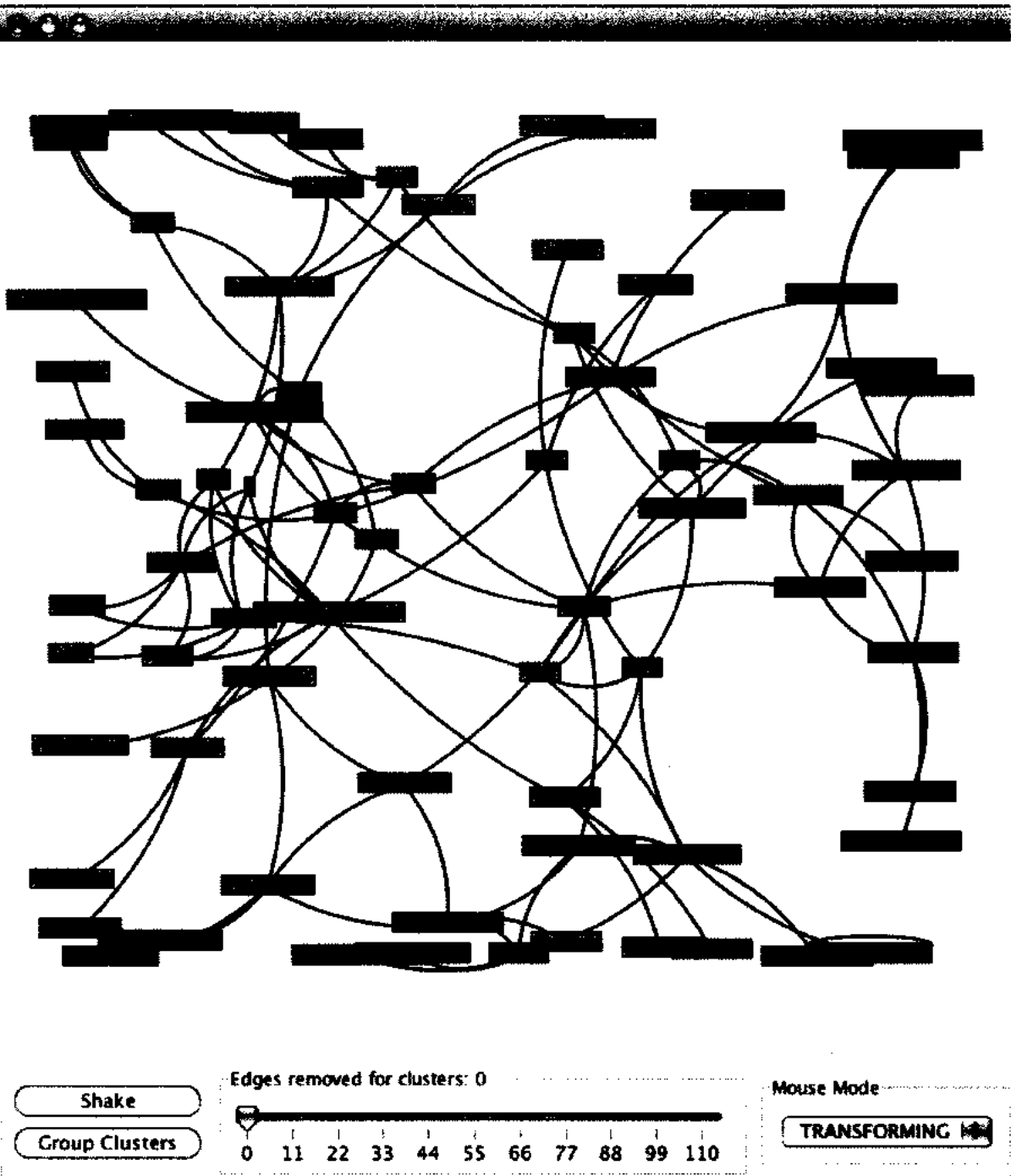
Although the first two extraction methods above provide a modeler with considerable control over the SemSim model decomposition process, they do not explicitly encourage the creation of models with a modular computational arrangement. That is, the extracted model may not have a low ratio of external inputs to internal dependencies. Furthermore, these two methods rely on the user having an idea about what they want their extracted model to contain, either semantically or computationally. Therefore, I implemented a third extraction method where the Extractor automatically discovers and extracts the highly interconnected components within a model's computational dependency network. In other words, this method identifies the "naturally occurring" computational modules within the code whether clustered intentionally or not.

Clicking the "Cluster" button below the list of model outputs (Figure 7) generates a new window for visualizing intra-model clusters. This interface is based on the Java Universal Network Graphing (JUNG) API, which implements the Girvan and Newman clustering algorithm [89]. This algorithm removes graph edges with a high level of "betweenness" in order to identify clustered sub-graphs. The algorithm calculates the betweenness metric by first computing the shortest paths between all vertex pairs in the graph, then identifying the most commonly traversed edges among these paths. Because the developers have not yet generalized their method to weighted or directed graphs, I used graphs with undirected edges as input to this clustering algorithm.

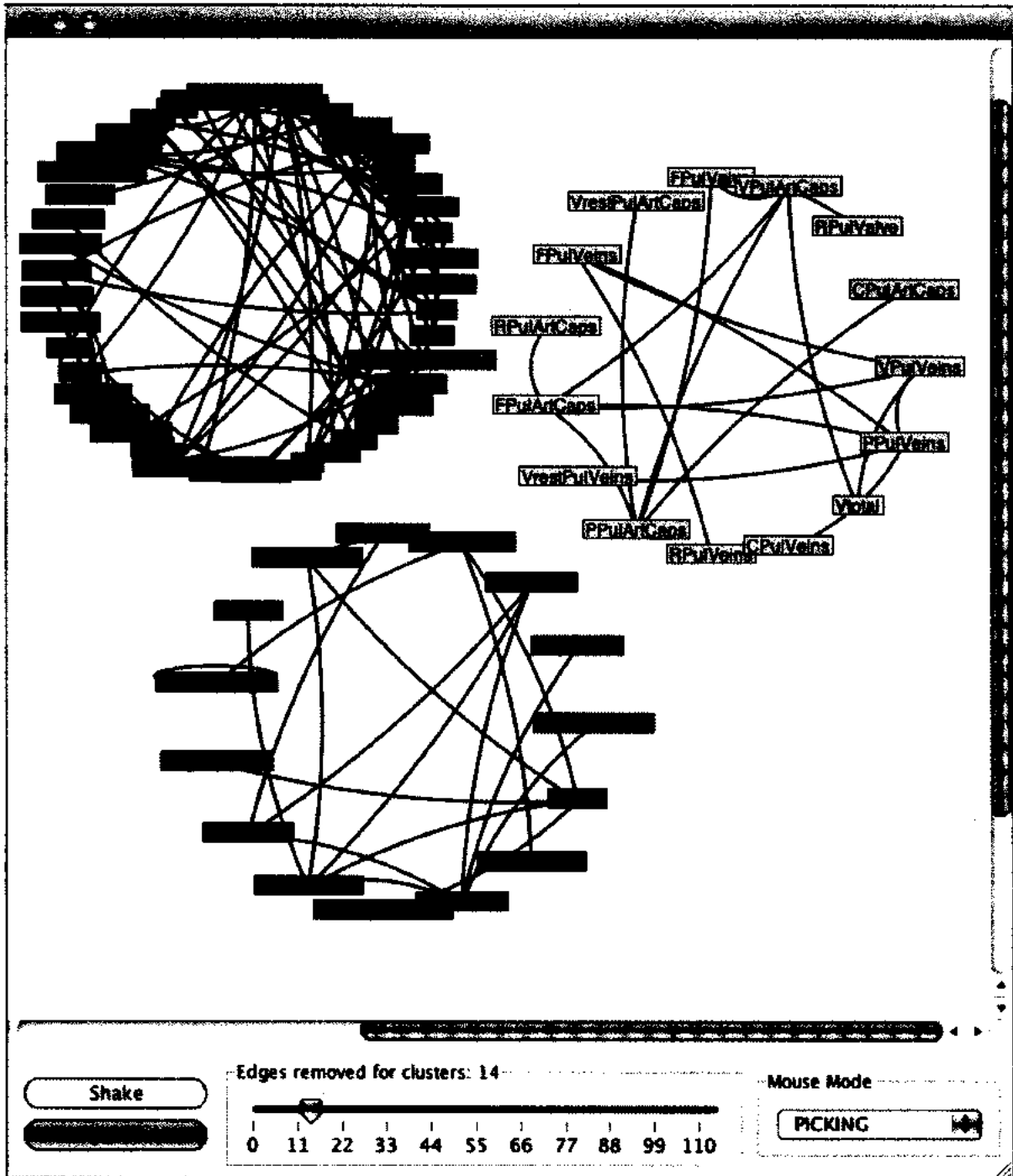
According to its developers, the algorithm runs in  $O(n^3)$  time in the worst case scenario, where  $n$  is the number of vertices in the graph [89]. Despite the algorithm's cubic dependency on the size of the network, the computational cost of the Girvan and Newman algorithm was not problematic when dealing with even the largest model (where  $n=234$ ) used in my research.

To identify clusters in their model, users adjust a slider bar to remove highly traversed edges and the JUNG API generates a new graph, separately coloring any clusters that result from

edge removal (Figure 9). As the user increases the number of edges removed using the slider bar, so does the number of distinct, clustered sub-graphs. Separate selectable checkboxes then appear in the main Extractor interface according to the clusters available for extraction, and these checkboxes are color coded to match the clusters in the clustering interface. By selecting one or more of these checkboxes users can extract the selected modules as a new SemSim model.



**Figure 8.** The interface for identifying computational clusters within a SemSim model. In this example the interface shows the unclustered computational dependency graph of the CV model described in Gennari et al. [55]



**Figure 9. Clustering analysis on a lumped parameter cardiovascular model. At an edge removal level of 14, the clustering algorithm identifies three distinct clusters, generally delineating the systemic circulation, pulmonary circulation and heart chambers.**

### 5.3 Merger

I created the Merger tool in order to help automate the recombination of individual SemSim models into larger, integrated systems. This tool compares the semantic content of two models to be merged and identifies semantically equivalent codewords. These equivalent codewords represent resolution (or coupling) points between the two models. They are the points at which computational dependencies are established between parts of the component models. Completely automating this resolution process requires knowledge about the user's purpose in constructing the merged system, but because there is presently no way for SemGen to understand this purpose, the current resolution process is semi-automated, requiring the user to manually specify how to couple the two models at each resolution point.

As mentioned in section 4.7, I had previously merged SemSim models in a semi-automated way using the Prompt ontology-merging plug-in for Protégé [78, 90]. After working with this tool, my colleagues and I decided that attempting to customize Prompt to automate SemSim merging would not be as desirable as building our own merging tool. Prompt identifies links between *single* classes or instances in two ontologies, and this limitation precludes its use as a SemSim merging tool. In order to identify semantic overlap between SemSim models, the *multiple*, linked instances involved in composite annotations must be compared in series. With our own tool integrated within SemGen, we offer a more automated, focused and powerful merging method specifically designed for merging SemSim models.

#### 5.3.1 Automatic detection of semantic overlap

Opening a new Merger tool prompts the user to select two existing SemSim models from a local directory. After loading these models, the Merger lists all the codewords from both models in two separate scroll panes at the bottom of the interface (Figure 10). The Prompt tool (see section 4.7), inspired several elements of the Merger interface design, especially the blue and red color-coding of the separate models.





After loading the component models into the interface, the Merger compares codeword annotations between the models and identifies semantic equivalencies. The Merger steps through each codeword in the first SemSim model and compares its semantic annotation with the annotations of all codewords in the second. If the codewords being compared both have composite annotations, the Merger compares each annotation component in series, starting with the physical property. If at each step the ontology source URI and the ontology term URI of both annotation components match, and the two codewords have the same relationships between their annotation components (e.g. *RO:contained in*, *RO:part of*, etc.) then the codewords are considered semantically equivalent and the Merger stores this resolution point. Alternatively, if the codewords being compared have non-composite annotations, the Merger identifies them as equivalent if both refer to the same ontology source URI and the same class URI. This method of identifying semantic equivalency is based on the MIREOT (Minimum Information to Reference an External Ontology Term [91]) standard, which recommends using both the source ontology URI and the ontology term URI when referencing an external knowledge source. Therefore, for any two annotation components, both the ontology URIs and the ontology term URIs must match in order for the Merger tool to recognize them as semantically equivalent.

In the worst case, the process of identifying semantic equivalencies takes  $O(mn)$  time, where  $m$  is the number of annotation components in the first model, and  $n$  is the number of annotation components in the second. However, despite the size of the models used for my end-to-end demonstration of modular modeling (Chapter 6), the Merger performs this model comparison task in a matter of seconds, even when thousands of codeword-codeword comparisons are required.

### 5.3.2 Manual mappings between codewords

Users may sometimes wish to assert a semantic equivalency between codewords that are not exact semantic matches as identified by the Merger. For example, as I will discuss in Chapter 6, when coupling the CV model with the vascular smooth muscle (VSM) model described by Gennari et al., I had to *manually* map the resistance of systemic arteries and capillaries in the

CV model to the resistance of the systemic arterioles in the VSM model.

The user can specify this type of manual mapping by selecting a single codeword from each codeword list at the bottom of the interface and pressing the “Add manual mapping” button. The Merger then adds this mapping to the list of resolution points in the center scroll pane and the user can choose their resolution method (see section 5.3.3 below).

In future versions, SemGen will be able to suggest this kind of mapping automatically, provided that custom annotations link to concepts in external reference ontologies. Using these links, the Merger tool will be able to identify codewords that may be closely related semantically, but not semantically equivalent. This feature will be implemented as a “Merging Wizard” that will employ a set of tools to more thoroughly examine semantic relationships between annotations. For example, provided a user links the custom annotation *Lumen of systemic arterioles* in the VSM model to *FMA:Lumen of systemic arterial tree* via an *RO:part of* relationship (see section 4.4), the Merger could recognize the similarity between the resistance codewords in the CV and VSM models and automatically suggest the mapping, thereby eliminating the need to map these codewords manually. I discuss this future work in more detail in Chapter 8.

### 5.3.3 Resolving points of semantic overlap

Once the Merger identifies and lists all points of semantic overlap in the interface, the user must choose the resolution method for each point. There are three available methods: preserve the codeword from the first model, preserve the codeword from the second model, or ignore the equivalency and keep the codewords disjoint. If the user chooses one of the first two options, the Merger uses the retained codeword in place of the discarded codeword in the equations of the merged system. This merging process follows the procedures previously detailed by Gennari et al. [55] In the resulting merged model the preserved codeword is solved according to its original equation and the equation solving for the discarded codeword is removed.

For example, when merging the CV and BARO models, the Merger recognizes that both models have codewords that simulate aortic blood pressure because both are annotated as *OPB:Fluid pressure <physical property of> FMA:Portion of blood <RO:contained in> FMA:Lumen of aorta*. The codeword used in the CV model is “Paorta” whereas “Paop” is used in the BARO model. In the CV model “Paorta” is calculated from the fluid analog of the law of capacitance and it depends on a state variable that simulates blood volume in the systemic arteries and capillaries compartment. Thus, “Paorta” is time varying. In the BARO model, however, aortic blood pressure is set as a constant, and “Paop” does not vary in time. In this merging demo I wanted to couple the baroreceptor to the circulatory model to produce a feedback loop that controls blood pressure. Therefore, I wanted to use the time varying aortic pressure as input to the baroreceptor equations, and so I chose to preserve “Paorta” from the CV model in the Merger resolution step. The Merger preserved the equation from the CV model that solves for aortic blood pressure and discarded the equation for aortic blood pressure in the baroreceptor model. If I had preserved aortic blood pressure from the CV model, the Merger would have kept aortic blood pressure as a user-defined external input parameter.

If the user chooses to ignore a point of semantic overlap and keep the codewords disjoint, then the Merger preserves both codewords and the equations that solve for them in the merged system. Two codewords with the exact same semantics may nonetheless have distinct mathematical roles. For example, I used this resolution option when merging the CV and BARO models in order to keep aortic blood pressure (CV) and an aortic blood pressure follower variable (BARO) distinct. While these two terms have the exact same semantic annotation, the follower variable is included as a mathematical device within the MML code in order to compute a time-derivative of aortic blood pressure, which is an input to the baroreceptor equations (see section 6.1.1).

In order to help the user decide which codeword they should preserve in the resulting merged model, I created a feature that describes the different computational configurations that will result from each possible resolution decision. Clicking the question mark button in one of the

resolution panels opens a new window with a text description of how the resolution decision will effect the computations in the merged model. In this case the user can see that if they choose to keep the “Paorta” codeword from the CV model rather than the “Paop” codeword from the BARO model, “Paorta” will become a time-varying input to the baroreceptor equations.

For the two semantic resolution points in the CV+BARO merging example that I did not ignore, the codeword from one model is a constant and the other varies in time. This distinction facilitates the user’s decision about which codeword to preserve, and requires a basic set of automated merging procedures. However, some resolution points require choosing between two variable codewords and the merging process can become more complex. For example, in my second Merging demonstration I merged the glycolysis model by Nielsen et al. [7] with the pentose phosphate pathway (PPP) reactions of the Chassagnole et al. metabolism model [8]. Both models contain codewords that represent the chemical concentration of D-fructose 6-phosphate. However, while there is only one reaction in the Nielsen et al. model that produces this chemical, there are three in the Chassagnole et al. model. The equations that solve for the concentrations of this species are therefore different in the two models, and in order to ensure that the merged model accounts for all the reactions that produce this species, the Merger must allow – to some degree – the merging of these two equations. I discuss the implications of this situation further when I detail this chemical network-merging task in section 6.2.

#### **5.3.4 Resolving points of syntactic overlap**

Once the user chooses which codewords to preserve in the merged model for each semantic resolution point, they press the “Merge” button, choose a name and location for the new merged model, and the Merger begins the process of composing the two individual models together. If both models contain a codeword with the same name (e.g. both CV and BARO models use “HR” for the heart rate codeword), the Merger prompts the user to rename the codeword in the second model. The Merger changes the codeword wherever it appears in the second model’s equations and renames the corresponding OWL individuals to reflect the new

name in the SemSim ontology. This procedure is important because, presently, each codeword in a SemSim model must have a unique name.

### **5.3.5 Resolving conflicts with units**

SemGen does not currently guarantee that SemSim models encoded for numerical simulation will be unit-balanced. This is because SemGen uses unit-free equations from the JSim compiling process that already contain unit correction factors. Thus, the merging of semantically identical codewords between models can introduce units-based mathematical inconsistencies in a model's equations. In order to avoid producing such inconsistencies, the Merger tool checks whether the codewords involved at a resolution point have the same units. If they do not, the Merger prompts the user for a conversion factor that is applied wherever the preserved codeword replaces the discarded codeword in the merged model. Although this functionality was enough to ensure the numerical accuracy of the models produced in my end-to-end demonstration of modular modeling, it may be insufficient for broader application. I discuss some of the limitations in unit consistency checking in section 8.1.3.

Once the Merging process is complete, the Merger saves the new merged SemSim model in the user-specified location. The Merger then asks the user if they want to encode the merged SemSim model as a new MML model so that it can be executed as a numerical simulation. If the user confirms, the Coder tool performs the SemSim-to-MML translation.

## **5.4 Coder**

The Coder automatically translates a SemSim model into simulation code that can be executed within a numerical solver. Presently, equations in SemSim models are encoded using MML syntax, since SemSim models must first exist as MML models. Therefore, the Coder currently translates SemSim models into MML files, which can then be loaded, compiled, and run within the JSim simulation environment. In the future, a more general equation encoding scheme, most likely MathML [92], will allow SemSim models to be translated into and from a wider variety of simulation languages.

When the Coder opens, it prompts the user for the name and location of the new MML file. Then, the Coder steps through all the data structures in the SemSim model and creates an appropriate MML declaration string depending on whether the data structure represents a solution domain, a user-defined input, or a model variable. The Coder writes out a new MML file to the user-specified location that consists of a standard MML file header, the solution domain declaration (if required), the list of codeword declarations, the initial conditions of model state variables, the list of equations used to solve the declared variables and a block comment reference table that lists all codewords and their semantic annotations. For reference, physical units are included in codeword declarations in the resulting MML model; however, the Coder keeps the MML unit conversion off because the original SemSim model equations are taken from the unit-independent form used in the MML-to-SemSim conversion process (see section 5.1). The Coder is a simple tool that is completely automated: no user input is required beyond specifying the name and location of the new MML model.

### ***5.5 Summary***

I designed the four tools detailed in this chapter to automate, as much as possible, the process of creating, decomposing, merging, and recoding SemSim models. Together, these tools provide a comprehensive, flexible system for building biosimulation models from modular, interoperable components. In order to demonstrate the utility of SemGen across modeling domains, physical scales, and modeling languages, I used SemGen to automate two model-building tasks that would otherwise require extensive hand coding. I discuss the results of these tasks in detail in the next chapter.

## **Chapter 6: Using SemGen for semantic composition of models: proof of concept**

The ultimate goal of my dissertation research is to demonstrate that SemGen provides a means for composing and decomposing multi-scale and multi-domain biosimulation models in a more automated, modular fashion. In order to provide this proof of concept, I used SemGen to perform two distinct model-merging tasks.

For my first model merging task I reproduced the hand-merged, multi-scale cardiovascular model described by Gennari et al. [55] discussed in section 4.7. This merged system combines a circulatory dynamics model (CV), a baroreceptor model (BARO) that controls heart rate based on aortic blood pressure, and a vascular smooth muscle model (VSM) that relates cell-level calcium dynamics to arteriolar diameter and resistance. For the Gennari et al. study, I combined MML versions of these three models *by hand* to produce a multi-scale cardiovascular model that simulated the influence of calcium dynamics on heart rate. For my dissertation research I merged these same three models by converting them into the SemSim format, annotating them with the Annotator, then using the Merger to compose them together. I was then able to validate the numerical results of the SemGen-merged system against those of the original hand-merged system that my colleagues and I developed previously. I describe my work on this task in section 6.1.

For my second model-merging task I wanted to demonstrate SemGen's ability to modularize models produced within another research domain, and, secondarily, to work with models originally coded in another language. For this task I combined aspects of two different SBML models of cellular metabolism. After browsing available SBML models in the Biomodels.net database I found two candidates: an *in vitro* glycolysis model published by Nielsen et al. [7], and an *E. coli* glycolysis model by Chassagnole et al. [8] that also simulates the pentose phosphate pathway (PPP), an alternative means of cellular energy



production. My modeling goal was to extract the PPP portion from the Chassagnole et al. model then merge it with the Nielsen et al. glycolysis model. In practice, this kind of merging task would be performed in order to replace the Chassagnole et al. version of glycolysis with the Nielsen et al. version. Or, the task might be performed if a user want to increase the detail of the Nielsen et al. model by adding a PPP shunt. I describe my work on this second merging task in section 6.2.

### ***6.1 Task 1: Merging the CV, BARO and VSM models***

After importing the MML versions of the CV, BARO and VSM models into the Annotator (and thus, creating SemSim versions of each), I annotated their codewords. This process required the use of standardized terms from the OPB, FMA, ChEBI, the NCI Thesaurus, and the ECG Ontology. The fluid flow equations in the CV model are mainly physics-based rather than constitutive (empirical), and so I was readily able to annotate most of the CV model's codewords using composite annotations. These annotations generally begin with an OPB property (*Fluid pressure, Fluid volume, etc.*) that links to composite physical entity (*FMA:Portion of blood <RO:contained in> FMA:Cavity of left ventricle*, for example). Additionally, because the FMA contains a robust set of cardiovascular concepts, I was able to thoroughly annotate the model with reference ontology terms. However, I did need to create custom physical entity annotations to describe two combinations of anatomical terms: *Lumen of systemic arteries and capillaries* and *Lumen of pulmonary arteries and capillaries*. I used the singular concept *Heart Rate* from the NCI Thesaurus for the both the CV and BARO models' heart rate codewords (because heart rate is a property of a physical *process* – see section 4.2), and I used the singular term *PR Duration* from the Electrocardiography Ontology to annotate the “PRint” codeword, which represents the time between atrial and ventricular contraction.

While I was able to thoroughly capture the semantics of the CV model, the BARO model proved more difficult to annotate. The BARO model is based heavily on constitutive, rather than physics-based, equations. Currently, the issue of how to annotate the parameters that shape these constitutive dependencies is an open question because the parameters' semantic

meaning lies more in the computational aspects of the model than in the physical or biological aspects. Therefore I left the annotation of these constitutive equation codewords to future work and focused on annotating those codewords in the BARO model with distinct semantics.

The VSM model contains mainly physics-based equations based on traditional electrophysiology and diffusion laws, and the codewords in these equations have distinct semantics. I was therefore able to annotate many of the codewords in the VSM model against OPB terms using composite annotations. However, the model required several custom physical entity annotations in order to represent the modelers' conceptual partitioning of the interior of the vascular smooth muscle cell into three concentric volumes. While the electrophysiological laws in the VSM model are physics-based, the equations in the other major aspects of the model, which relate intracellular calcium concentration to muscle activation and arteriolar diameter, are constitutive. Therefore, as in the BARO model, I did not annotate the codewords that constrain these constitutive relationships.

### 6.1.1 Merging the CV and BARO models

I then began the merging process by merging the CV and BARO models. When loaded into the Merger, the Merger identified three points of semantic equivalence between the models (Figure 10). The first resolution point between the two models involves aortic blood pressure from the CV model and an aortic blood pressure follower variable from the BARO model. The follower variable in the BARO model is used as a mathematical device to compute the time-derivative of aortic blood pressure, which is another input required by the baroreceptor equations. "PaopFOL" is computed in MML with this equation:

$$d\text{PaopFOL}/dt = (\text{Paop} - \text{PaopFOL})/\text{tauFOL};$$

In engineering terms, the "Paop" term is passed through a low-pass filter. This filter has a small enough time constant ("tauFOL") that the "PaopFOL" curve almost exactly matches the "Paop" curve. By computing "PaopFOL" this way, its derivative can be used as an input

in other MML equations. Presently the SemSim framework does not offer a means of capturing the semantics of “PaopFOL’s” use as a mathematical device. Therefore, I annotated this codeword with the same semantics as “Paorta” in the CV model and “Paop” in the BARO model. Semantically, all three codewords are equivalent, and that is why the Merger identified this second resolution point. I wanted to preserve this follower variable in the merged system without having it replace the “Paorta” term in the CV model. Therefore I chose to ignore this first semantic equivalency (see section 5.3.3 for background on resolving semantic equivalencies).

The second resolution point identified by the Merger was between the heart rate codewords in the CV and BARO models. Both of these codewords are annotated using the non-composite annotation term *Heart Rate* from the NCI Thesaurus. Whereas “HR” in the CV model is a static parameter, it varies in the BARO model according to aortic blood pressure and the pressure’s time-derivative. I wanted to preserve the latter dependencies in the merged model and make the baroreceptor-computed heart rate an input to the CV equations. This resolution step, along with the first, creates a feedback loop between the baroreceptor equations and the circulatory system: “HR” is computed from a set of initial conditions, then used to compute “Paorta”, and “Paorta” is used to compute the next “HR” value. With the models coupled this way, the system simulates baroreceptor feedback control of blood pressure. The baroreceptor equations reduce heart rate when aortic blood pressure is elevated and increase heart rate when the pressure rises. I therefore chose to preserve the “HR” codeword from the BARO model in this resolution step.

The third resolution point represents the equivalence between the codewords that simulate aortic blood pressure in the two models. Because aortic blood pressure varies according to volumes and flows in the CV model and is constant in the BARO model, I chose to preserve the “Paorta” term from the CV model for this resolution step. With this assertion, the time-varying aortic blood pressure from the CV model becomes an input to the baroreceptor equations, coupling the models.

Once I had chosen my resolution options for this first merging task, I merged the models. The Merger first prompted me to resolve a unit inconsistency between the two HR terms. The units for HR in the CV model are “1/min” whereas the units are “min<sup>-1</sup>” in the BARO model. Although the units for these terms are mathematically identical, they are captured as strings within the SemSim framework and not identical syntactically. After entering a conversion factor of 1, the merging process completed and the Merger saved the new CV+BARO SemSim model. As I discuss later (section 8.1.3), it may be possible to automate this kind of resolution step in the future using unit conversion algorithms within the JSim engine.

### **6.1.2 Merging the CV+BARO and VSM models**

I next merged the CV+BARO model with the VSM model. My modeling goal for this stage of the merging process was to use the arteriolar resistance from the VSM model as an input to the CV model so that cellular calcium dynamics in the VSM model would influence systemic arterial blood flow in the CV+BARO model. After loading these models, the Merger did not automatically recognize any points of semantic overlap between them because these models do not share any codewords that are annotated with identical semantics. Therefore, in order to couple these models, I created a manual mapping between the VSM codeword representing systemic arteriolar resistance (“Rsa”) and the CV codeword representing the resistance of the systemic arteries and capillaries compartment (“Rartcap”). In other words, because the systemic arterioles account for the major portion of resistance to blood flow in the systemic arterial/capillary network, I made the assumption that arteriolar resistance was equivalent to total arterial/capillary resistance when merging these models. These kinds of simplifying assumptions are commonplace in computational modeling, because they minimize model complexity while preserving the validity of results. This is why model merging can never be a fully automatic process, as these assumptions require manual mappings between semantically distinct concepts.

I then merged the models and the Merger prompted me for another units conversion factor: Arteriolar resistance in the original VSM model is computed using units of mmHg\*s/L

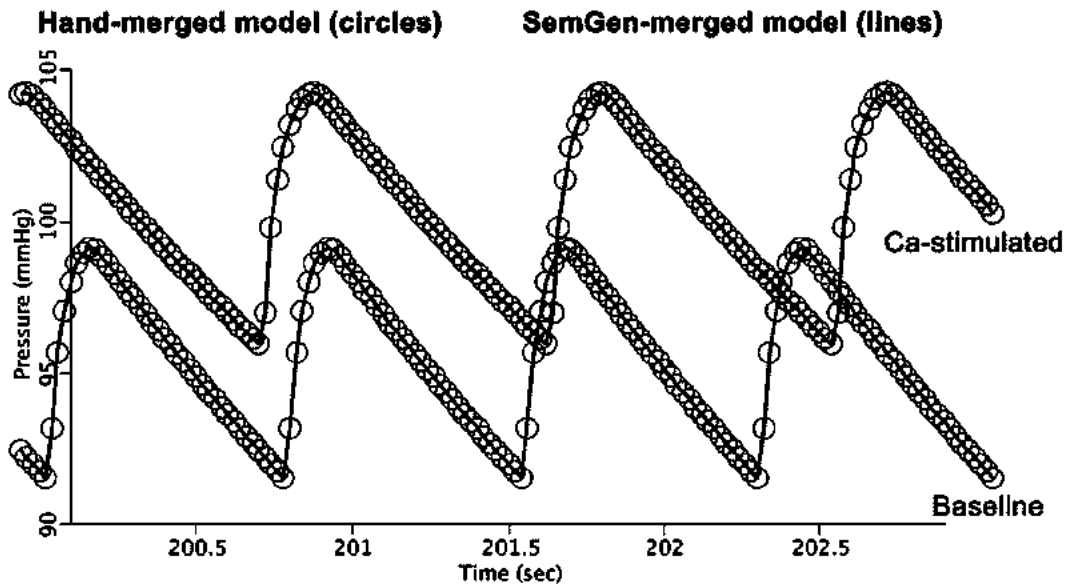
whereas systemic arterial/capillary resistance in the original CV model is in mmHg\*sec/ml. Therefore I used a conversion factor of 0.001 L/ml to unit-balance the resulting equations in the merged model. The Merger also prompted me to resolve a point of *syntactic* overlap between the models. Both models include a codeword named “Rsa”. In the CV+BARO model “Rsa” represents the resistance of the systemic arteries and in the VSM model “Rsa” represents the resistance of the systemic arterioles. Based on the assumptions mentioned above about systemic resistances, I elected to use the “Rsa” term from the VSM model and discard the “Rsa” term from the CV+BARO model.

Once the merging process completed I used the Coder to translate the CV+BARO+VSM SemSim model into executable MML code. This code compiled successfully, and can be executed within JSim. However, in order for this model to completely reproduce the numerical results of the hand-coded version of the CV+BARO+VSM model, I had to add several manual edits. Originally, after creating the hand-merged version of the CV+BARO+VSM model, I edited it slightly to improve its internal stability and verisimilitude. Therefore, I included these same edits in the SemGen-merged version once I had it encoded as an MML file. First, in the original hand-merged version, I changed the VSM model’s arteriolar pressure (“Partl”) from a static input parameter to a variable that is equal to the average between the aortic and systemic venous blood pressures from the CV model. Instead of using a static constant arteriolar pressure, the CV model provides the VSM equations with a more realistic, time-varying estimate of arteriolar pressure. This change also serves as a second coupling point between the models, wherein the VSM model becomes dependent on the CV model’s circulatory dynamics. (SemGen did not identify this point of coupling because the CV and BARO models simulate aortic blood pressure, not arteriolar blood pressure.) I added this same manual edit to the SemGen-generated CV+BARO+VSM model. Next, I reproduced another set of hand-coded changes made to the original merged model. During longer simulations the original hand-coded model would become unstable because the heart rate changes within the heart cycle. In physiological terms, the model was not simulating the myocyte refractory period, and the heart was free to initiate another beat before the cardiac cycle completed. Therefore, I introduced a discrete heart rate term into the

hand-coded model that only updates after the completion of a heartbeat. With this edit the heart rate is still variable, as it is computed from the aortic pressure-dependent baroreceptor equations, but it does not update continuously. The model's stability improved with the addition of this new discrete heart rate term, and I saw the same effect when I manually coded the same feature into the SemGen-generated CV+BARO+VSM model.

Once I had incorporated these manual changes made to the original model I compared the numerical results of the SemGen-coded CV+BARO+VSM model to the results of the hand-coded version. When I examined the time curves for "Paorta" during baseline and calcium-stimulated simulations I saw no visible difference between the hand-coded and SemGen-coded model results (Figure 11). However, on closer inspection I found that, while the numerical results for "Paorta" were extremely close, the SemGen-coded results deviated from those of the original model by an average of 0.00002% throughout the simulation. This small error may be due to the fact that the SemGen-coded version includes unit correction factors in its equations whereas the hand-coded version relies on JSim's automated unit correction features to introduce these factors during compilation.

## Aortic blood pressure

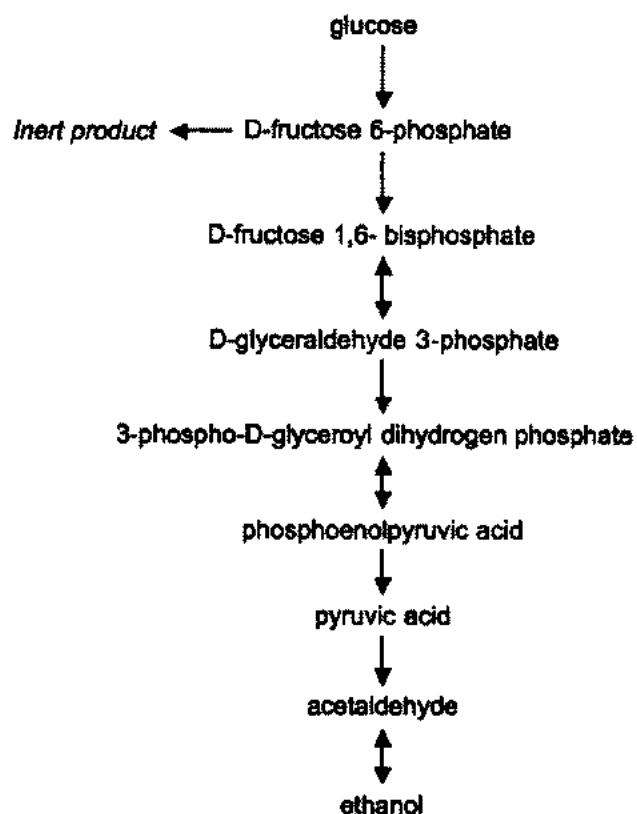


**Figure 11. Comparison between the simulation results of the hand-coded and SemGen-merged CV+BARO+VSM models.**

### ***6.2 Task 2: Merging the Nielsen et al. glycolysis model with the Chassagnole et al. pentose phosphate pathway model***

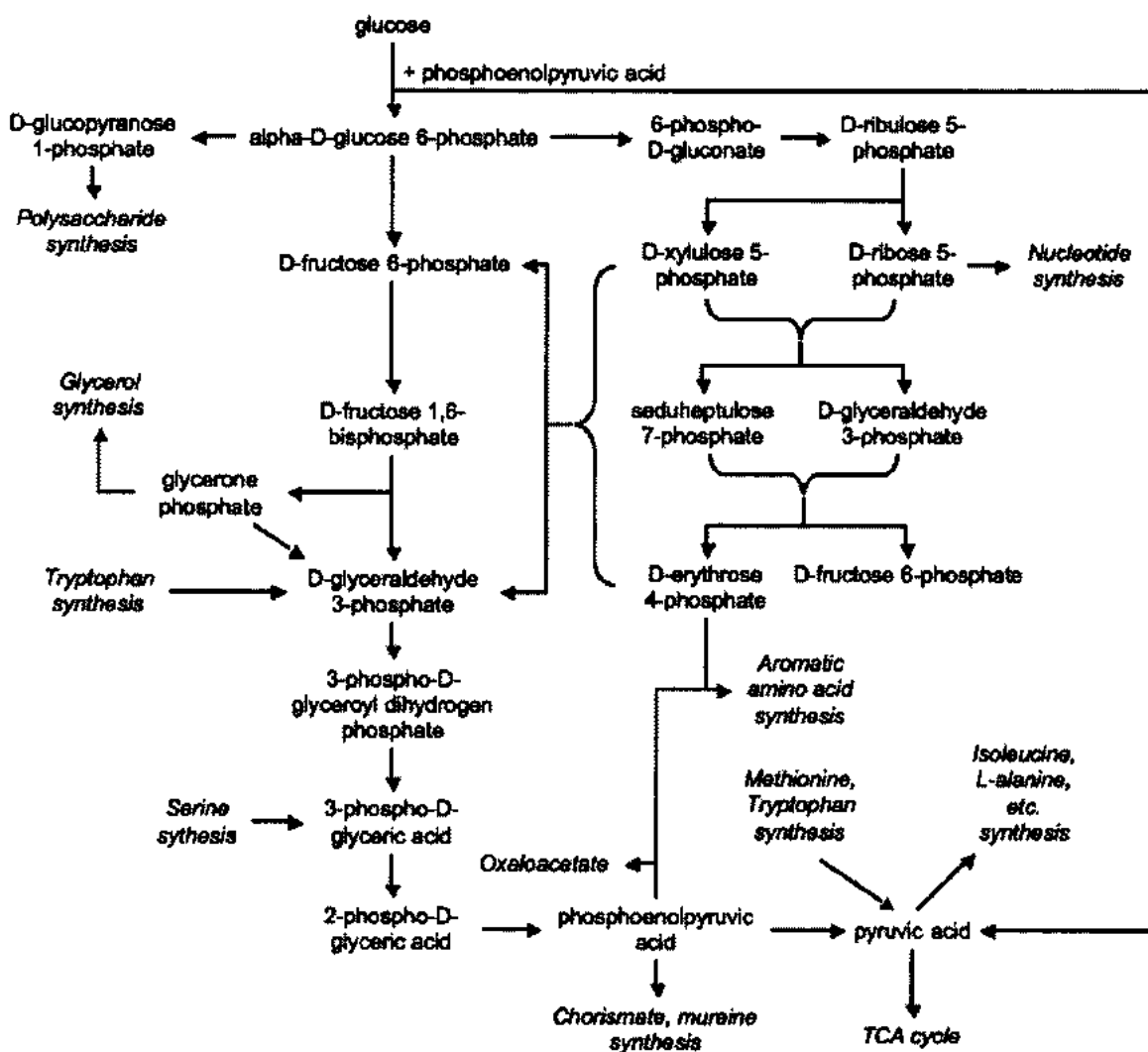
For my second merging task I wanted to merge models from a different research domain than the CV, BARO and VSM models. I also wanted to use models coded in a language other than MML in order to demonstrate SemGen's utility across modeling languages. I therefore chose to work with two SBML models that simulate cellular metabolism. In consultation with Dr. Herbert Sauro from the University of Washington Bioengineering department, I identified two candidate SBML models to merge. The first, published by Nielsen et al. [7], simulates the reactions of glycolysis as performed in an *in vitro* experiment carried out in an artificial reaction chamber (Figure 12). The second model by Chassagnole et al. [8] simulates carbon metabolism in more detail – it contains a glycolysis model that connects to the reactions of the pentose phosphate pathway (PPP) shunt, an alternative energy-production pathway (Figure 13). For this second merging task I wanted to extract out the PPP portion of the

Chassagnole et al. model and merge it with the Nielsen representation of glycolysis. This process might be performed by a modeler interested in either augmenting the original Nielsen et al. model, or replacing the glycolysis pathway of the Chassagnole et al. model with a different representation.



**Figure 12. Structure of the Nielsen et al. glycolysis model. In addition to the species shown, some of the reactions involve one or more of the metabolites AMP, ADP, ATP, NAD and NADH.**





**Figure 13. Structure of the Chassagnole et al. carbon metabolism model. Adapted from Chassagnole et al. [8] In addition to the chemical species shown, some of the reactions involve one or more of the metabolites AMP, ADP, ATP, NAD, NADH, NADP and NADPH.**

### 6.2.1 Creating SemSim versions of the metabolism models

In order to create SemSim versions of the Nielsen et al. and Chassagnole et al. models, I first downloaded the SBML versions of these models from the Biomedel.net database. Next, as discussed in section 5.1, I used the MML language as in interlocutor to convert the SBML models into the SemSim format. I first imported the SBML models into JSim, which

automatically generated MML versions of the models, including all the equations necessary to solve the system of reactions. I then saved the MML versions as two new files and imported them into SemGen using the Annotator tool. The Annotator generated new SemSim versions of the metabolism models and automatically added physical property annotations to those codewords with units corresponding to a single *OPB:Physical property* subclass.

Next, I manually annotated both metabolism models using the Annotator in order to make them as semantically interoperable as possible. Based on the variables' units, the Annotator annotated most of the physical properties of the variables in these models automatically during the MML-to-SemSim conversion. Therefore, my primary task was to complete the composite annotations for the model codewords using physical entity references. Nearly all the physical entity annotations used in these two models were taken from the ChEBI ontology. Two annotations for the Chassagnole et al. model were taken from the GO cellular component ontology and I created a custom physical entity annotation for the "Continuous flow stirred-tank reactor" container in the Nielsen et al. model. Although I was able to capture the semantics for many of the codewords in the metabolism models, some codewords, such as the reaction rate constants, were part of constitutive dependencies, and some required physical process annotations. Both of these types of annotations were outside the scope of my research and so I left some codewords in the metabolism models unannotated. I discuss these annotation limitations further in section 6.2.1.2.

#### **6.2.1.1 *Errors and inconsistencies in curation of the metabolism models***

The original SBML code for both metabolism models contains some useful semantic information about the physical entities present in the model. Each chemical species is annotated against the ChEBI ontology; therefore, I used these annotations as a guide in creating my own annotations in SemGen. However, I discovered several issues associated with SBML model curation as I examined each species annotation.

First, while both models simulate reactions involving fructose 6-phosphate, the model curators annotated this chemical species differently in the two models. In the Nielsen et al.

model the annotation refers to *ChEBI:20935* (“D-fructose 6-phosphate”) whereas the annotation in the Chassagnole et al. refers to *ChEBI:16084* (“beta-D-fructofuranose 6-phosphate”). I submitted an annotation error report to the Biomodels database and they acknowledged that the species in the Chassagnole et al. model had been annotated improperly. Therefore I used the ChEBI term from the Nielsen et al. model for the Chassagnole et al. model’s annotations.

Secondly, in anticipating the process of merging the models together, I noticed that the models used different annotations for two other shared chemical species: pyruvate and phosphoenolpyruvate. The Nielsen et al. annotations referred to the conjugate acids of these chemical species (“pyruvic acid” and “phosphoenolpyruvic acid”) whereas the Chassagnole et al. model referred to the conjugate bases. I again contacted the Biomodels team about the discrepancy, they acknowledged it, and said they would change the annotations in the Nielsen et al. model to the conjugate bases. I therefore used the conjugate base annotations in the SemSim versions of both metabolism models.

#### **6.2.1.2 Limitations in annotating the metabolism models**

Many of the codewords in the metabolic models represent the rates of chemical reactions. For example, “reaction\_2.rate” in the Nielsen et al. model represents the rate of the reaction [fructose 6-phosphate + ATP → fructose 1,6-bisphosphate + ADP] in units of millimoles per minute. The SemSim composite annotation scheme is inadequate to fully capture the semantics of this codeword because the codeword represents the chemical flows of *several different* chemical species. In discussions with Dr. Cook, we decided that the appropriate annotation procedure here would be to link the codeword’s *physical dependency* annotation to its physical entity role players (if not already asserted via the computational dependencies in the model). In other words, the user would use the Annotator to assert that the physical dependency of “reaction\_2.rate” involves fructose 6-phosphate, ATP, fructose 1,6-bisphosphate and ADP as role players. I chose not to include physical dependency annotations as part of my dissertation work, and ultimately I was able to perform the extraction and merging tasks required for my end-to-end demonstration without relying on

such manual dependency annotations; however, these kinds of annotations will be critical in future applications of SemGen, as they will greatly improve the precision of the model extraction and merging process. I discuss the importance of dependency annotations further in Chapter 8.

### **6.2.2 Extracting the pentose phosphate pathway from the Chassagnole et al. model**

After annotating the metabolism models, my next task was to extract out the portions of the Chassagnole et al. model that simulate the pentose phosphate pathway (PPP) shunt. My goal was to merge this extracted model with the Nielsen et al. glycolysis model. I opened the Chassagnole et al. SemSim model in the Extractor and selected those physical chemical species involved with the PPP shunt for extraction. This selection was based on the chemical species involved in the PPP pathway as indicated in the network diagram in the Chassagnole et al. source publication ([8], Figure 4). The ChEBI names of the preserved species are:

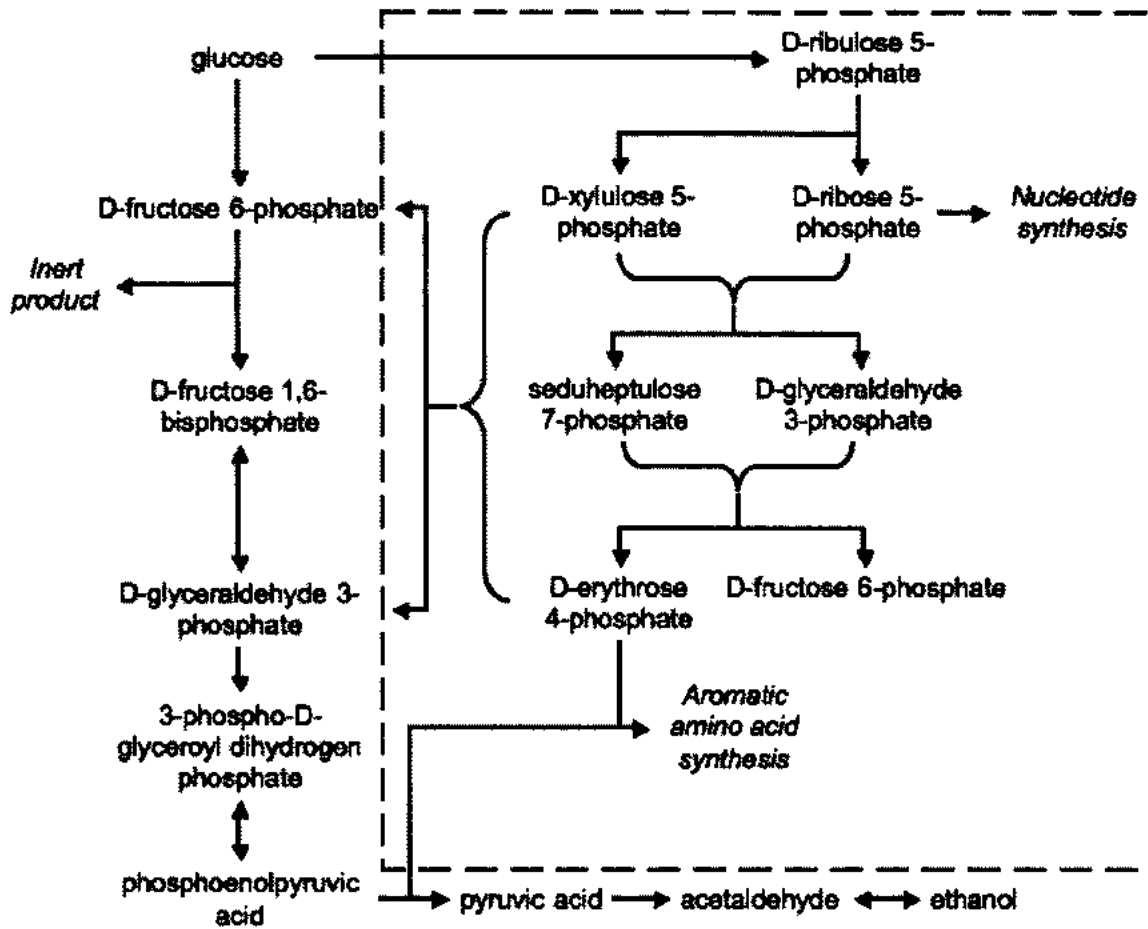
- 6-phospho-D-gluconate
- ATP
- D-erythrose 4-phosphate(2-)
- D-fructose 6-phosphate
- D-glyceraldehyde 3-phosphate
- D-ribose 5-phosphate
- D-ribulose 5-phosphate
- D-xyulose 5-phosphate
- NADPH
- sedoheptulose 7-phosphate

I used the “More inclusive” option for this physical entity extraction (see section 5.2.2) in order to avoid turning many of the reaction rates in the extracted model into user-defined inputs. Because of the limitations in annotating reaction rates discussed in section 6.2.1, the “Less inclusive” extraction option does not preserve the dependencies associated with

reaction rates. Instead, these rates change into user-defined inputs, removing important interdependencies between chemical species concentrations and the rates of the reactions that affect these concentrations. Using the “More inclusive” type of extraction allowed me to bypass these limitations. The future challenge here lies in associating reaction rate codewords with the chemical species that participate in the reaction. Theoretically, if I had established semantic links between each reaction rate in the parent model and all the chemical species involved in its reaction, I could have used the “Less inclusive” extraction option and extracted a similar PPP model.

### **6.2.3 Merging the PPP shunt with the Nielsen et al. glycolysis model**

My second merging goal was to produce a new metabolic model by coupling the PPP model with the Nielsen et al. representation of glycolysis to produce the Glycolysis+PPP model as illustrated in Figure 14. After I extracted the PPP shunt from the Chassagnole et al. model, I loaded the new PPP SemSim model and the Nielsen et al. SemSim model into the Merger tool. Upon loading, the Merger tool performed its automated, pair-wise comparison between the codewords in the two models and identified 14 points of semantic equivalence. Table 1 presents these resolution points in detail.



**Figure 14. Structure of the target Glycolysis+PPP model. Chemical species and reactions inside the dashed box belong to the PPP model, those outside the box belong to the Nielsen et al. glycolysis model.**

**Table 1. Semantic equivalencies automatically identified by the Merger tool when merging the pentose phosphate pathway (PPP) model with the Nielsen et al. glycolysis model. For each equivalency, the common annotation is listed along with the corresponding codewords from the PPP and Nielsen et al. models in parentheses.**

<i>OPB:Discrete chemical concentration &lt;physical property of&gt; CHEBI:D-fructose 6-phosphate (cf6p, F6P)</i>
<i>OPB:Discrete chemical concentration &lt;physical property of&gt; CHEBI:NAD (cnad, NAD)</i>
<i>OPB:Discrete chemical concentration &lt;physical property of&gt; CHEBI:phosphoenolpyruvic acid (cpep, ADP)</i>
<i>OPB:Discrete chemical concentration &lt;physical property of&gt; CHEBI:ADP (cadp, ADP)</i>
<i>OPB:Discrete chemical amount flow rate &lt;physical property of&gt; CHEBI:D-glyceraldehyde 3-phosphate (vGAP.rate, GAPflow.rate)</i>
<i>OPB:Discrete chemical concentration &lt;physical property of&gt; CHEBI:AMP (camp, AMP)</i>
<i>OPB:Extensive discrete chemical amount &lt;physical property of&gt; CHEBI:D-fructose 6-phosphate (cf6p.amt, F6P.amt)</i>
<i>OPB:Discrete chemical concentration &lt;physical property of&gt; CHEBI:D-glyceraldehyde 3-phosphate (cgap, GAP)</i>
<i>OPB:Discrete chemical concentration &lt;physical property of&gt; CHEBI:3-phospho-D-glyceroyl dihydrogen phosphate (cpgp, DPG)</i>
<i>OPB:Discrete chemical concentration &lt;physical property of&gt; CHEBI:D-fructose 1,6-bisphosphate (cfdp, FBP)</i>
<i>OPB:Extensive discrete chemical amount &lt;physical property of&gt; CHEBI:D-glyceraldehyde 3-phosphate (cgap.amt, GAP.amt)</i>
<i>OPB:Discrete chemical concentration &lt;physical property of&gt; CHEBI:NADH (cnadh, NADH)</i>
<i>OPB:Discrete chemical concentration &lt;physical property of&gt; CHEBI:ATP (catp, ATP)</i>
<i>OPB:Discrete chemical amount flow rate &lt;physical property of&gt; CHEBI:D-fructose 6-phosphate (vf6p.rate, F6Pflow.rate)</i>

The high degree of semantic overlap between these two models is due to the fact that they both simulate glycolysis, and thus, include many of the same chemical species. I elected to preserve the codewords from the Nielsen et al. model for all 14 points of resolution because I wanted to keep the Nielsen et al. representation of glycolysis and simply graft the PPP shunt onto its existing computational structure.

In addition to the 14 automatically identified resolution points, I also created three manual semantic mappings between codewords in the two models. First, all chemical species in SBML models must be associated with a fluid compartment in which they reside and this compartment's volume is required to calculate chemical concentrations during numerical simulation. For my merging task I wanted to create a new system where the chemical species of the PPP shunt were in the same fluid compartment as the species of the Nielsen et al. glycolysis model. Therefore, I manually equated the codeword representing the fluid volume of the "cytosol" compartment in the PPP model with the fluid volume of the "continuous flow stirred-tank reactor" compartment of the Nielsen et al. model. By equating these compartments, then electing to preserve the Nielsen et al. codeword, I ensured that chemical concentrations of all species in the merged model would be calculated as if they existed within the same fluid compartment.

Second, in order to drive the PPP shunt with an input chemical flow from the Nielsen et al. glycolysis reactions, I manually equated the Nielsen et al. representation of glucose with the PPP representation of 6-phospho-D-gluconate. I made this mapping because the Nielsen et al. model condenses the first few reactions of glycolysis into a single reaction whereas these reactions are represented in more detail in the Chassagnole et al. model (compare Figure 12 and Figure 13). Consequently, the Nielsen et al. model does not include alpha-D-glucose 6-phosphate, the chemical species in the glycolysis chain that is required to initiate the first reaction of the PPP shunt, nor 6-phospho-D-gluconate, the product of that first reaction. I therefore made the assumption that the chemical concentration and chemical amount of 6-phospho-D-gluconate in the PPP model would be the same as the concentration and amount of glucose at the head of the Nielsen et al. model. After making these mappings I elected to preserve the Nielsen et al. glucose codewords, therefore replacing 6-phospho-D-gluconate as the initial chemical substrate of the PPP shunt with glucose (see top of Figure 14). In the merged model, glucose is consumed by two reactions, the first producing D-fructose 6-phosphate, the second producing D-ribulose 5-phosphate, the upstream reactant of the PPP component.

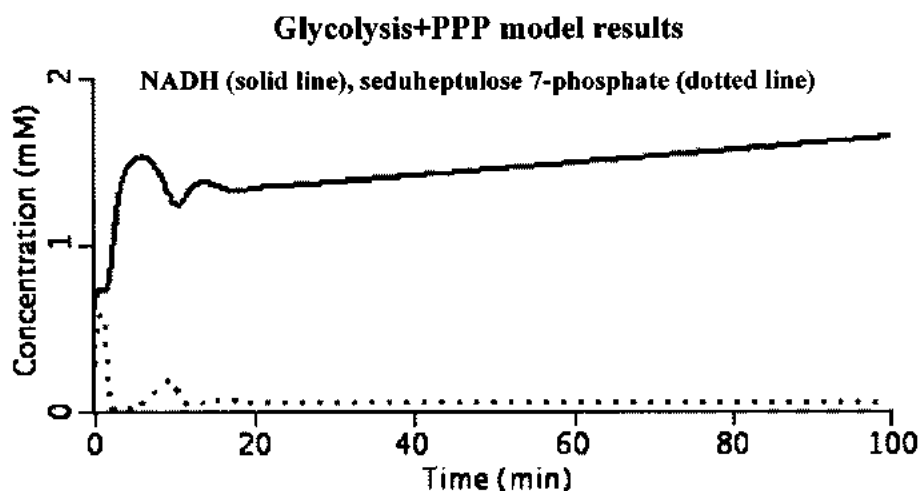


When I merged the PPP and Nielsen et al. models, the Merger tool prompted me for several conversion factors to ensure that the merged model's equations included appropriate unit balance. Both models use the same unit name "time," but time represents minutes in the Nielsen et al. model and seconds in the PPP model. Therefore, the chemical flows in the Nielsen et al. model are all computed in units of millimoles per minute whereas they are computed in units of millimoles per second in the PPP model. When the Merger tool prompted me to reconcile this unit conflict, I included a conversion factor of 60 sec/min in order to ensure unit balance between the models.

In my early experiments with this merging task I recognized that in order to couple the PPP and Nielsen et al. models so that the chemical reactions of the PPP model influence the chemical species in the Nielsen et al. model, I would need to recode several conservation equations controlling the formation and consumption of species in the merged model. In grafting the PPP shunt onto the glycolysis model, I introduced new reactions that affect the chemical flows of the species in the glycolysis model. For example, the PPP model includes a reaction that produces D-fructose 6-phosphate, a chemical also represented in the Nielsen et al. glycolysis pathway. Therefore, I needed to adjust the equations controlling the chemical formation and consumption of D-fructose 6-phosphate when merging the two models together. Simply replacing the PPP codeword for D-fructose 6-phosphate with the Nielsen et al. codeword does not provide this important level of coupling. In order to help automate this equation editing, I created a subroutine in the merging program that identifies candidate conservation equations that the user may want to adjust given the semantic overlap in the model. Specifically, for each semantic resolution step, the Merger examines whether the discarded codeword is computationally dependent on codewords annotated as *OPB:Flow rates* (such as *Discrete chemical amount flow rate* or *Fluid flow rate*). The Merger then collects these flow codewords and prompts the user to direct their influence on codewords that are 1) dependent on one or more of these flows, 2) annotated as an *OPB:Displacement* (state variables), and 3) annotated with the same physical entity as the codewords in the resolution step. This coupling via flow conservation equations is very crucial for merging the

reactions of chemical network models, and analogous situations can arise in other modeling domains. For example, suppose I merged the CV model described by Gennari et al. with a model that simulates the infusion of blood into the systemic veins. Both models might use the same annotation for pressure in the systemic veins, and the Merger would recognize this overlap. However, if I wanted to preserve the infusion flow into the circulation so that it increases blood volume, I would need to include the infusion model blood flow in the conservation equation that solves for the state variable representing blood volume in the systemic veins compartment - just as I needed to include the new chemical flows of D-fructose 6-phosphate in the conservation equations that solve for the chemical amount of that species in the Glycolysis+PPP merged model.

Once the merging process completed, I encoded the new Glycolysis+PPP SemSim model as an MML model using the Coder tool. This model compiled successfully and produced the results shown in Figure 15. As shown in the figure, the time curve of NADH concentration in the merged system does not oscillate continuously, even though it does so in the original Nielsen et al. model. Instead, NADH concentration only oscillates for a transient period at the beginning of the simulation and eventually plateaus in longer simulations. Given this change in the model's behavior, and the similar shape of transient fluctuations among codewords from either component model, it was evident that the component models of the merged system were coupled.



**Figure 15. Numerical results from the Glycolysis+PPP merged model. Solid curve: concentration of NADH, a species involved in the glycolysis reactions. Dotted curve: concentration of sedoheptulose 7-phosphate, a species in the PPP shunt. Both time curves show an initial transient oscillation that eventually dampens.**

In order to further check this coupling, I set all the chemical flows from the PPP component that affect species in the Nielsen et al. component to zero. Since this edit removes all influence of the PPP component on the merged system, I expected the revised model to reproduce the original results of the Nielsen et al. model, specifically the NADH concentration curve. I found that this version did exactly reproduce the NADH curve from the original Nielsen et al. results, confirming that the original model had not been altered outside of its coupling with the PPP shunt, and that the chemical flows between the shunt and the glycolysis network accorded with the target chemical network.

### **6.3 Summary**

In this chapter I have demonstrated how a user can use SemGen and the SemSim modeling approach to translate existing biosimulation models into the SemSim format, decompose them into re-useable sub-models, and recombine them in a modular way to produce runnable composite models. I used all four main components of SemGen in concert to perform this demonstration: the Annotator, Extractor, Merger and Coder. The demonstration's success

validates the functionality of these tools. I have also shown how SemGen can accommodate models coded in other simulation languages like SBML using MML as an interlocutor. **Most importantly, because the merged models I produced for this demonstration simulate biological phenomena across multiple physical scales and multiple physical domains, I have demonstrated SemGen's utility as a semantic composition tool that is multi-scale and multi-domain.**

## **Chapter 7: Results from applying the cluster identification method to test models**

As described in Chapter 5, users can identify and extract clusters of highly interconnected codewords in a model's computational network using the third type of decomposition available in SemGen's Extractor tool. This method relies on the cluster identification algorithm of Girvan and Newman [89] as implemented in the JUNG network-graphing package [82] to locate modular components within SemSim models. Given that biosimulations represent biological function according to a modeler's conception of the relationships between semantically distinct concepts, one might expect that the clustering algorithm would identify semantically related clusters within a model's computational network. That is, the computational modules within a model might reflect its semantic components. To explore this issue further, I applied the clustering method on two test models, the lumped parameter cardiovascular model described by Gennari et al. (CV-[55]) and the Chassagnole et al. [8] carbon metabolism model. Primarily, I was concerned with determining whether this clustering method might provide a general means of separating a model's components into semantically related modules at various levels of biological organization. I found that while this method does delineate semantically related model features in many cases, there is no guarantee that these clusters will match a user's conception about the model's semantic architecture.

### ***7.1 Results of clustering analysis on the CV model***

First, I applied the clustering method to the CV model. This model consists of a closed-loop, lumped parameter circulatory system with of a four-chambered heart, a two-compartment systemic circulation and a two-compartment pulmonary circulation. In section 5.2 I discussed a real world model decomposition task that required an extraction of the systemic and pulmonary circulations of this model. Thus, I applied the clustering method to this model to see if, once it identified three distinct modules, these modules would represent the heart, pulmonary and systemic circulatory components (thus facilitating the decomposition task).

Additionally, I investigated whether a clustering of eight distinct modules would delineate the eight distinct circulatory segments. These three-module and eight-module decompositions represent two levels of biological organization, and both could conceivably be useful to modelers wishing to separate out the CV model's components into reusable pieces.

After loading the SemSim version of the CV model into the clustering dialog I incrementally increased the number of highly traversed edges to remove from the graph. At each step in this process I recorded the list of physical entities included in each identified cluster. The results of this analysis on the CV model are presented in Figure 9 (section 5.2.4) and Table 2. For the most part, when the method identified three distinct modules, it grouped the heart, pulmonary circulatory, and systemic circulatory components separately. Notably, the clustering algorithm grouped the pulmonary and aortic valves with the circulatory loops they feed rather than with the heart components. This shows that, while these valves are anatomically part of the heart, in the CV model they are more closely associated with the computational dependencies of the circulatory loops than with those of the heart.

**Table 2. Two levels of network clustering performed on the CV model. The table lists the physical entities present in each identified computational module.**

Edges removed	Module	Physical entities in module
14-19	Module 1	Portion_of_blood_contained_in_Cavity_of_left_atrium
		Portion_of_blood_contained_in_Cavity_of_left_ventricle
		Cardiac_ventricle
		Portion_of_blood_contained_in_Cavity_of_right_atrium
		Portion_of_blood_contained_in_Cavity_of_right_ventricle
	Module 2	Portion_of_blood_contained_in_Tricuspid_valve
		Portion_of_blood_contained_in_Mitral_valve
		Cardiac_atrium
		Portion_of_blood_contained_in_Lumen_of_systemic_venous_tree_organ
		Portion_of_blood_contained_in_Lumen_of_aorta
Module 3	Portion_of_blood_contained_in_Aortic_valve	
	Portion_of_blood_contained_in_Lumen_of_systemic_arteries_and_capillaries	
	Portion_of_blood_contained_in_Lumen_of_pulmonary_venous_tree_organ	
	Portion_of_blood_contained_in_Lumen_of_cardiovascular_system	
30-31	Module 1	Portion_of_blood_contained_in_Lumen_of_pulmonary_arteries_and_capillaries
		Portion_of_blood_contained_in_Pulmonary_valve
		Portion_of_blood_contained_in_Aortic_valve
	Module 2	Portion_of_blood_contained_in_Lumen_of_aorta
		Portion_of_blood_contained_in_Lumen_of_systemic_arteries_and_capillaries
	Module 3	Portion_of_blood_contained_in_Cavity_of_left_ventricle
		Portion_of_blood_contained_in_Mitral_valve
	Module 4	Portion_of_blood_contained_in_Cavity_of_left_atrium
		Cardiac_atrium
	Module 5	Portion_of_blood_contained_in_Lumen_of_pulmonary_venous_tree_organ
		Portion_of_blood_contained_in_Lumen_of_pulmonary_arteries_and_capillaries
	Module 6	Portion_of_blood_contained_in_Lumen_of_cardiovascular_system
		Portion_of_blood_contained_in_Lumen_of_pulmonary_arteries_and_capillaries
	Module 7	Portion_of_blood_contained_in_Pulmonary_valve
		Portion_of_blood_contained_in_Lumen_of_systemic_arteries_and_capillaries
	Module 8	Portion_of_blood_contained_in_Lumen_of_systemic_venous_tree_organ
Portion_of_blood_contained_in_Cavity_of_right_ventricle		
Module 9	Cardiac_ventricle	
	Portion_of_blood_contained_in_Tricuspid_valve	
	Portion_of_blood_contained_in_Cavity_of_left_ventricle	
Module 10	Portion_of_blood_contained_in_Cavity_of_left_atrium	
	Portion_of_blood_contained_in_Cavity_of_right_atrium	

When the clustering method identified eight distinct modules, the separation between the eight segments of the circulation was not completely distinct. As shown in Table 2, modules 4 and 6 included physical entities from two different circulatory compartments. In general, however, the terms in each module are intimately related anatomically, even though this decomposition did not separate out the eight circulatory containers distinctly.

## ***7.2 Results of clustering analysis on the Chassagnole et al. carbon metabolism model***

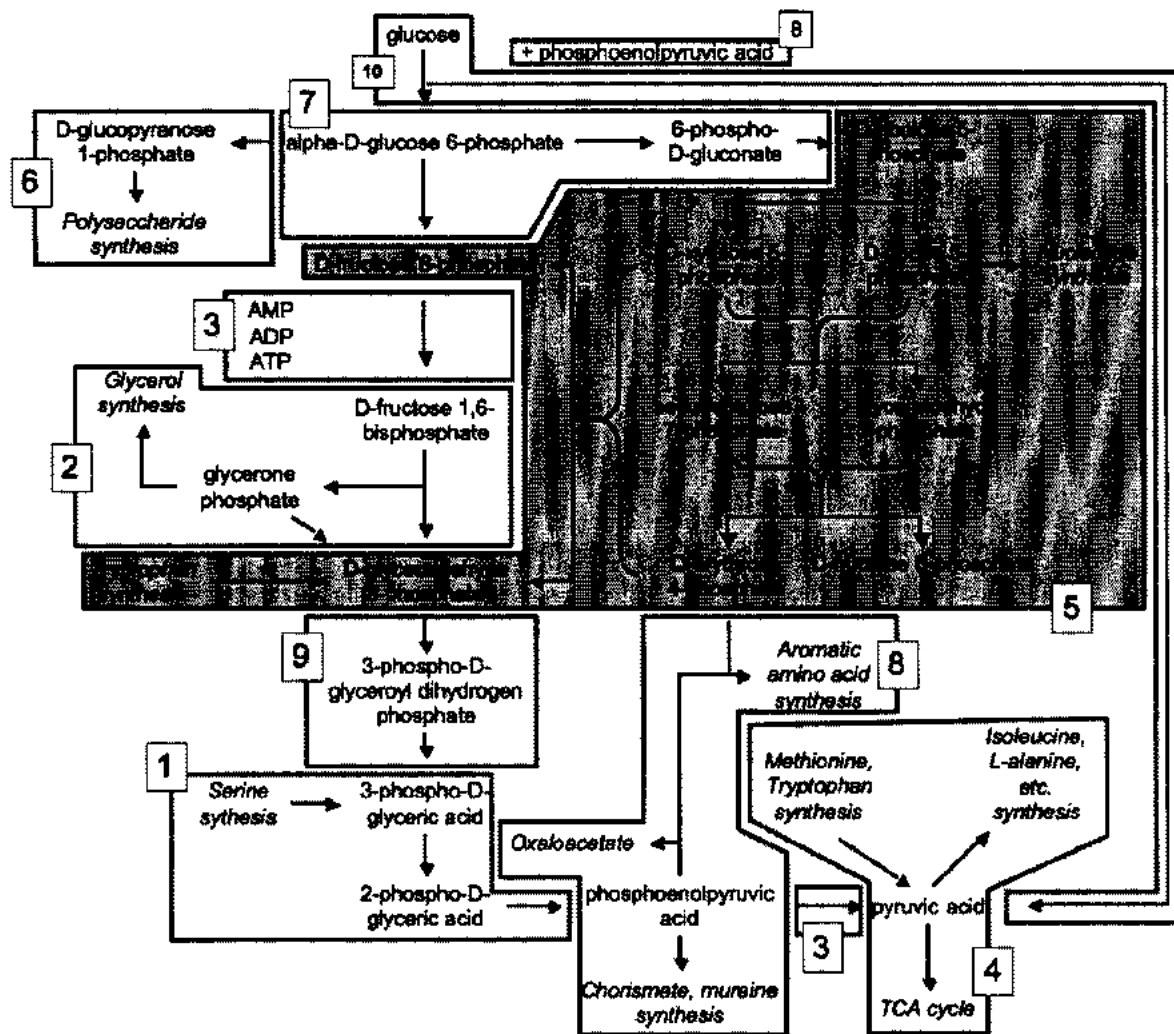
To further explore the capabilities of the automatic clustering method, I performed a clustering analysis on the Chassagnole et al. carbon metabolism model used in my second end-to-end demonstration of modular modeling. I was interested in whether the clustering algorithm would identify a computational module that represented only the chemical species involved in the PPP shunt. A modeler may want to extract the PPP shunt from its parent model for reuse, just as I did in my second end-to-end demonstration of model merging (section 6.2.2). I hypothesized that this module would come into relief at a gross level of clustering – i.e. after the algorithm had identified two or three distinct modules.

As with the CV model, I loaded the Chassagnole et al. model into the Extractor tool, and started the clustering analysis, which incrementally increased the number of edges removed from the graph. I then recorded the number of clusters identified by the algorithm along with the physical entities associated with each cluster. It took significantly longer to perform the clustering analysis on the Chassagnole et al. model than the CV model. This is because the CV model has 114 edges connecting 76 codewords and the Chassagnole et al. model has 429 edges connecting 234 codewords. As mentioned in section 5.2.4, the Girvan and Newman algorithm performs clustering in  $O(n^3)$  time in the worst case scenario, where  $n$  is the number of vertices in the graph. Despite the Chassagnole et al. model's higher complexity, however, a single clustering analysis on the model only takes a matter of seconds on my MacBook Pro Intel Duo Core laptop, and the entire analysis still completed in a manner of minutes.

I iteratively removed the edges in the Chassagnole et al. model using the clustering algorithm



until it identified a total of 15 modules. Then, I reviewed the grouping of chemical species throughout the different clustering levels and found that the algorithm most precisely delineated the chemical species of the PPP shunt when it decomposed the model into ten separate clusters (110-119 edges removed). Thus, my initial hypothesis about the clustering level required to delineate the PPP module was incorrect. Figure 16 shows the grouping of chemicals and reactions in each of these ten modules (note that clusters 3 and 8 are discontinuous using this representation of the network). Cluster 5 in the figure (shaded) includes all the chemical species involved in the PPP shunt except 6-phospho-D-gluconate. Table 3 presents the ten distinct clusters identified at this clustering level.



**Figure 16. Structure of the Chassagnole et al. metabolic model showing the groupings of chemical species when the clustering analysis identified ten distinct clusters. Among all the clusters automatically identified during the analysis, module 5 (shaded) most precisely delineated the chemical species of the PPP shunt, although it excludes 6-phospho-D-gluconate.**

**Table 3. Grouping of chemical entities in the Chassagnole et al. model when the clustering algorithm identified ten distinct clusters.**

Edges removed	Modules	Physical entities in module
110-119	Module 1	2-phospho-D-glyceric acid 3-phospho-D-glyceric acid
	Module 2	D-fructose 1,6-bisphosphate glycerone phosphate
	Module 3	AMP ADP ATP
	Module 4	pyruvic acid
	Module 5	D-erythrose 4-phosphate (2-) D-glyceraldehyde 3-phosphate D-ribose 5-phosphate D-fructose 6-phosphate D-xylulose 5-phosphate sedoheptulose 7-phosphate D-ribulose 5-phosphate cytosol
	Module 6	D-glucopyranose 1-phosphate
	Module 7	6-phospho-D-gluconate alpha-D-glucose 6-phosphate NADP NADPH
	Module 8	phosphoenolpyruvic acid
	Module 9	3-phospho-D-glyceroyl dihydrogen phosphate NAD NADH
	Module 10	glucose extracellular region

The fact that module 5, containing the majority of the PPP shunt's chemical species, persists as a relatively large module at a high level of edge-removal testifies to its high level of interconnectedness. Although the clustering algorithm did not separate out the PPP module at more gross levels of decomposition (where it identified only two or three modules), the finer levels of decomposition began to reveal the PPP component. Thus, while the PPP shunt remains tightly coupled with, and semantically indistinct from, the glycolysis portion of the Chassagnole et al. model, the clustering algorithm eventually identified most of the shunt as a

unique, tightly interconnected portion of the metabolic network.

### **7.3 Discussion**

Whereas the clustering analysis on the CV model successfully delineated semantically related clusters at a gross level (it separated out the heart, pulmonary and systemic circulatory components), the analysis on the Chassagnole et al. model only separated out the PPP shunt at finer levels of decomposition. This is due to the fact that the gross level components of the Chassagnole et al. model, the glycolysis pathway and the PPP shunt, are more highly interconnected computationally than the gross level components of the CV model. In the Chassagnole et al. model the glycolysis pathway and the PPP shunt share six points of coupling, whereas there is less overlap between the three main circulatory components of the CV model. The clustering algorithm did eventually delineate a cluster representing nearly all of the PPP shunt; however, it never completely delineated all of the shunt's chemical components and required a high level of edge removal before doing so.

My results show that while the clustering method provides a means for modelers to discover candidate modules for extraction, the strength of the semantic relationships within these modules can vary. Because computational connectedness does not always reflect semantic connectedness, there is no guarantee that the clustering algorithm will decompose a model according to a user's conception of its semantic architecture. However, as shown here, the clustering algorithm can sometimes delineate useful, semantically related modules that are loosely coupled to the rest of the model's computational network.

Throughout these tests with the clustering algorithm I have used a network graph where the vertices represent model codewords and the edges represent an undirected computational dependency between two codewords. This is only one of several ways of representing the computational structure of a model graphically. For example, one could use directed edges, or introduce computational dependencies as vertices. These alternative graphs may give very different clustering results, possibly improving the delineation of semantically related clusters.

## **Chapter 8: Limitations, future directions, and summary**

In this chapter I address the major limitations of my research and outline the future work needed to make semantic compositions of models within SemGen a more robust and automated process. I also provide a summary of my dissertation research and discuss its implications for the fields of biosimulation, informatics, and beyond.

### ***8.1 Research limitations***

While my research has a potentially broad impact on modeling and simulation in general, this dissertation represents a more preliminary proof-of-concept demonstration of SemGen's capabilities. Although I was successful in applying SemGen for my modular modeling tasks, several important opportunities for automating model composition remain, and SemGen will require further improvements and testing before it will be ready for widespread use.

#### **8.1.1 Limitations in semantic annotation**

As discussed in sections 4.2, I did not annotate the equations in the models used for my project. I made this decision in part because I wanted to focus on building the tools needed for codeword annotations, but also because reference sources for annotating equations across physical scales and domains are not currently available. While the Systems Biology Ontology (SBO) contains concepts that could be used to annotate the chemical network models used in my project, no resources exist for thoroughly annotating the equations in the other test models. Annotating model equations and connecting them with role player information would make the semantic information in SemSim models more complete and would better inform the resolution process during merging tasks. For example, if two models contain semantically identical fluid flow codewords, but one is solved using Ohm's Law for Fluids and the other using Poiseuille's Law, this information could help the modeler choose which codeword to preserve at the point of coupling.

As with equation annotations, I chose to scope my project so that I did not apply physical

*process* annotations in my research. For example, I did not annotate codewords like “N\_hrv” from the BARO model that represent the rate of a process – in this case the firing frequency of the vagus nerve. This limitation is important to consider because without process annotations, the Merger tool has no way to identify semantic equivalencies between models that contain the same process properties. If I had merged the BARO model with a more detailed model of vagal nerve dynamics, for example, the Merger would have likely missed some important semantic equivalencies when coupling the models. Consider also that when I merged the CV and BARO models, I coupled them through their representations of heart rate - the frequency of the cardiac contraction process. Fortunately, in this instance, I was able to use a non-composite annotation for the heart rate codewords in both models, and the merger identified this semantic equivalency. However, annotators will not always be able to use non-composite annotations as surrogates for process properties in this way. Had a non-composite annotation been unavailable in this case, this merging task may have been less successful, given SemGen’s current capabilities.

Along with equations and process properties, I did not apply annotations to many of the properties of constitutive dependencies for my project. These dependencies are based on empirical relationships between two or more physical properties rather than principles of physics. For example, the codewords for the maximum and minimum elastances of the left ventricle in the CV model are properties of a constitutive dependency. They shape the dependency that determines left ventricular elastance as a function of time. Therefore, while I was able to associate some physical entity information with these codewords (*FMA:Cavity of left ventricle*, for example), I left their physical property annotations unspecified.

Some constitutive dependency properties have more canonical usage than others. Linear fluid resistance, for example, is a constitutive property of the dependency between a pressure potential and fluid flow and is represented as a distinct class in the OPB. However, less canonical dependencies have less clear semantics, and my colleagues and I are currently tackling the issue of how best to annotate the codewords that shape these dependencies.

As part of the European Commission's Virtual Physiological Human (VPH) Project, my colleagues and I are addressing the above limitations in SemGen's annotation capabilities (see also section 8.2.1.2). We are currently researching how best to incorporate physical dependency, physical process, and constitutive dependency property annotations into the SemSim framework. As we make these architectural improvements, we will be able to more completely capture the biophysical meaning of SemSim model components, increasing their interoperability, and allowing more opportunities to automate model composition and decomposition tasks.

### **8.1.2 Limitations in SemGen's interoperability with other coding languages**

As discussed in section 5.1, SemGen can currently only import and annotate models that compile in the MML format. Although models in several other languages can be translated into MML, such as SBML and CellML, they sometimes require manual edits to compile successfully, and some of the semantic information encoded in these models is lost during the translation process. To preserve this information, and automate the annotation process as much as possible, SemGen must include parsing tools that convert SBML and CellML models directly into SemSim files. Furthermore, widespread adoption of SemGen and the SemSim framework will require parsing tools for many of the other simulation languages in existence. In particular, my colleagues and I will need to research whether it will be feasible to translate models written in procedural languages like MATLAB or FORTRAN into the SemSim format. While we were disheartened from our preliminary research on this issue [37] (see section 4.7), these procedural languages are some of the most commonly used within the biosimulation community. Even tools that only offer a partial solution to this translation problem might significantly promote SemGen's adoption as a modeling standard.

Currently, to convert a SemSim model into executable simulation code, users can only output to MML. Because researchers use simulation languages and environments tailored to their modeling needs, SemGen must also eventually include encoding algorithms that convert SemSim models into various other languages for numerical simulation. These features will also encourage SemGen's adoption as a modeling standard among the biosimulation

community.

### 8.1.3 Limitations in merging models

Several important limitations exist within the Merger tool that, when addressed, will improve its robustness and increase the automation of the model merging process. First, as discussed in section 5.1.1.2, SemGen allows the use of custom annotations, and annotators can link these annotations to reference ontology concepts via standardized relations. However, the Merger tool does not currently take these relationships into account when identifying potential points of coupling between models, it only identifies codewords that are semantically identical, not semantically *similar*. My colleagues and I envision the development of Merging Wizard tool that computes the semantic distance between ontology terms, and helps the modeler gauge the semantic similarity between components in different models. I discuss our approach to the Merging Wizard in more detail in section 8.2.2.

Second, the Merger does not currently make semantic comparisons between composite and non-composite annotations. Thus, there is no way for the Merger tool to recognize that a heart rate codeword annotated singularly as *NCIThesaurus:Heart Rate* is semantically identical to another heart rate codeword annotated compositely as *OPB:Event rate <physical property of> GO:Heart contraction*. In order to establish this functionality, model curators will need to create a repository of composite annotations and map its contents to non-composite annotations in external reference ontologies. This way the Merger tool will have a means of discovering semantic equivalencies between composite and non-composite annotations. As part of the VPH project, my colleagues and I are exploring the issues surrounding the creation of a composite annotation repository, including the mapping of repository terms to external sources.

Finally, in order to ensure numerical accuracy in a merged model, the Merger must be able to add in unit-correction factors as needed. For example, if a modeler replaces a codeword possessing units of millimeters of mercury with a codeword possessing units of kilopascals during the merging process, the Merger should introduce appropriate conversion factors into



the relevant equations. The Merger tool presently handles this issue in a limited, semi-automated capacity. During the model merging process, the Merger uses the MML string representations of units to ensure unit equivalencies. So, for example, if two semantically equivalent codewords have different unit strings, the Merger prompts the user for a conversion factor. However, the tool cannot yet recognize some unit equivalencies, such as “1/mmHg” and “mmHg<sup>(-1)</sup>.” Also, MML allows the use of custom units, which can be defined in terms of MML’s standardized unit set. If two models use custom unit declarations with the same name, but actually represent different units, then the Merger must be able to process this information as well, and insert conversion factors appropriately during the merging process. In order to automate these kinds of resolutions, the Merger needs access to a more structured representation of physical units. SemGen may be able to provide this functionality in the future by accessing the unit-conversion tools within the JSim simulation environment, or by using structured unit representations in the Unified Code for Units of Measure [93] or in other units ontologies [94, 95].

#### **8.1.4 Limitations in the types of models used for this project**

The models I used for my end to end-demonstrations of automated, modular modeling all used algebraic and/or ODE equations. My colleagues and I have not yet tested the SemSim framework on multi-dimensional (distributed) models that use partial differential equations (PDEs). Therefore, while my successful demonstrations apply to the many ODE-based, lumped-parameter models in existence, further research and development is needed to validate the SemSim approach for PDE-based models.

Additionally, all the models used for my research were deterministic – they contained no random, or stochastic, aspects. Because this is a distinction at the computational, as opposed to the semantic, level of a mathematical model, I believe the SemSim approach to modularity would also be valid for stochastic systems. The presence of stochastic features in a model would not impact the model composition or decomposition process within SemGen, because these processes do not depend on whether a model codeword is solved deterministically or randomly. Only the semantics of model components and the computational dependencies

between them must be considered during the composition and decomposition processes.

Lastly, while the models used for my project cover a wide range of the kinetic physical domains investigated within biological research, there are several domains they do not address. In terms of the OPB's seven physical domain categories, I used models that represent physical properties within the chemical, diffusion, electrochemical, and fluid domains. Further testing with models in the heat, potential field and solid kinetic domains will bolster my arguments about the generality of my modular modeling approach.

## ***8.2 Future directions***

In addition to testing the SemSim approach with different kinds of models and making the merging process more automated, other challenges and opportunities exist for improving the semantic composition and decomposition of SemSim models.

### **8.2.1 The annotation bottleneck**

In order to achieve the long-term goal of populating a repository of semantically interoperable SemSim models (see section 1.2), annotators must first convert existing models into the SemSim format and annotate them as thoroughly as possible. Although annotators can use SemGen to access a wide range of ontology concepts and automatically apply them within SemSim models, the annotation process still requires many manual steps. Particularly, when annotators search for annotations, they must take time to ensure they use the exact term they need.

#### ***8.2.1.1 Finding the right annotation component***

Currently SemGen allows annotators to perform string search queries on online OWL ontologies and other SemSim models. Although this functionality provides a useful start to the annotation process, annotators should be able to browse ontology trees and view all the relevant information about a particular ontology class within SemGen. This will help annotators decide between competing annotation components, and increase their familiarity with the content of existing ontologies. Given that the NCBO's BioPortal website already

provides much of this functionality through their browser interface and their web services, my colleagues and I envision programmatically linking SemGen to the BioPortal web services in the future so that SemGen can provide more informative annotation search tools.

My colleagues and I also envision developing a tool that will filter ontology search results based on existing semantic information in a SemSim model. For example, suppose an annotator is annotating a codeword that already has its physical property specified as an *OPB:Fluid pressure*. The annotator then sends a query to the FMA for completing the physical entity component of the annotation. In this case we want to provide a method for filtering the annotator's search results so they only include structural entities that are fluids (i.e. only those entities that can possess an *OPB:Fluid pressure*). By filtering search results this way, we will help the annotator more efficiently locate target annotation terms.

#### **8.2.1.2 Reusing composite annotations**

Reusing composite annotations is another means of accelerating the annotation process. As discussed in section 5.1.1.1, annotators can readily reuse *singular* annotation components from other SemSim models, but cannot yet copy a full composite annotation from one SemSim model to another. Although this functionality could be implemented relatively easily for the purposes of annotating SemSim models, my colleagues and I also want to store composite annotations as reusable concepts within their own repository. This way, the annotations become more readily searchable and available for annotating other kinds of biological information, such as experimental data. Using this approach annotators could search a single knowledge base for reusable composite annotations instead of searching over an entire set of SemSim models. My colleagues and I are currently working on this challenge as part of the VPH Project and are collaborating with a team from the University of Cambridge on how best to store composite model annotations in order to create semantic links between physiological models and other biomedical data [96].

#### **8.2.2 Merging Wizard**

As mentioned above, currently the Merger tool only recognizes semantic equivalencies, not

similarities when merging models. Therefore, we plan to create a Merging Wizard tool that computes the semantic similarity between non-equivalent codewords and suggests new potential coupling points between models. This tool will help automate the resolution process when merging models, and will allow semantic comparisons involving custom annotations. For example, I used the custom physical entity annotation *Lumen of systemic arterioles* for a resistance codeword in the VSM model and related it to *FMA:Lumen of systemic arterial tree* via a *part of* relationship. I also related this same FMA term to the custom term *Lumen of systemic arteries and capillaries* used in another resistance codeword annotation in the CV+BARO model. In this latter case I asserted that *Lumen of systemic arteries and capillaries* *<has part>* *FMA:Lumen of systemic arterial tree*. When I merged the VSM and CV+BARO models, the Merger tool was not equipped to discover that these two custom annotations were related semantically via the FMA term, and I had to manually map the two resistance codewords during the merging process. If the Merger tool been able to discover the connection between these codewords, it could have suggested the mapping automatically. This type of functionality, embedded in a Merging Wizard, will be extremely helpful during the merging process if a user does not already have intimate, *a priori* knowledge about the semantic overlap between merged models.

### 8.2.3 User testing and evaluation

Although SemGen currently provides many of the necessary functions for modular model composition, user testing will be required prior to public deployment within the biosimulation community. Since its creation, I have been SemGen's primary user, and the program has yet to be tested outside my research group. My colleagues and I would like to eventually conduct formal user testing on SemGen in order to improve the tool's utility and design and to anticipate the needs of biosimulation modelers. Because it is difficult to quantitatively assess the effectiveness of design tools like SemGen without a large user base, a more focused, *qualitative* study that explores users' experiences with the software would be most appropriate as a first step. My colleagues and I eventually plan to conduct a study wherein participants perform assigned modeling tasks with and without the aid of SemGen. Interviews, direct observations, and recordings of users' activity collected during this study

will inform future generations of SemGen and the design of quantitative evaluation studies.

Following qualitative user testing, I hope to release a SemGen version for public download, and then make iterative improvements as I obtain feedback from the user community. My hope is that as SemGen improves and evolves, it will become a valuable tool within the field of biosimulation, and that SemSim models will proliferate across research domains, accelerating the development of new, useful models in a variety of biomedical disciplines.

### **8.3 Summary**

Based on the modeling demonstrations provided in Chapter 6, I have established SemGen's utility as a tool for the modular composition and decomposition of biosimulation models. Because SemGen is based on the SemSim framework, its capabilities extend to biosimulation models across physical scales and physical domains. SemGen's utility is based on the semantic interoperability of SemSim models, which is enabled by the composite annotation scheme my colleagues and I developed (section 4.4). This scheme represents a major contribution of my research to the field of informatics, and its application extends beyond biosimulation modeling. Using our composite annotation approach, annotators in various other fields can capture the semantics of complex concepts by post-coordinating terms built from existing ontology concepts. My colleagues and I are planning to apply this approach to the annotation of various sorts of biomedical data as part of the VPH Project.

For my dissertation research I have also demonstrated the utility of three decomposition tools within SemGen, and shown, in Chapter 7, how the clustering identification method can automatically extract out modular portions of model code. While this extraction method allows the user to crop out computationally related modules within a model, there is no guarantee that decompositions of this kind will reflect the user's preconception about the model's *semantic* architecture.

The products of my dissertation research offer a modular approach to biosimulation modeling that meets the challenges of increasing model complexity and model interoperability

introduced in Chapter 2. My hope is that this approach will help evolve the field of biosimulation into a more productive, integrated, and collaborative industry. In particular, I hope my work will benefit researchers in the field of patient-specific modeling, as I believe this field has vast potential to improve medical-decision making. Additionally, many disciplines outside of biology and medicine use physics-based modeling and simulation for product development. Both the SemSim framework and SemGen scale beyond the field of biosimulation, and other industries could conceivably make use of the SemSim modeling approach, provided they have a rich set of ontologies for annotating the semantics of their models.

## Bibliography

1. Neal, M.L. and R.C. Kerckhoffs, *Current progress in patient-specific modeling*. Briefings in Bioinformatics, 2010. **11**(1): p. 111-126.
2. Bassingthwaite, J., P. Hunter, and D. Noble, *The Cardiac Physiome: perspectives for the future*. Experimental Physiology, 2009. **94**(5): p. 597.
3. Hucka, M., et al., *The systems biology markup language (SBML): a medium for representation and exchange of biochemical network models*. Bioinformatics, 2003. **19**(4): p. 524-31.
4. Krieger, K., *COMPUTER SCIENCE: Life in Silico: A Different Kind of Intelligent Design*. Science, 2006. **312**(5771): p. 189.
5. Lloyd, C.M., M.D. Halstead, and P.F. Nielsen, *CellML: its future, present and past*. Progress in Biophysics and Molecular Biology, 2004. **85**(2-3): p. 433-50.
6. *Working Group 10: Model Sharing and the Defining of Modeling Standards*. [http://www.imagwiki.org/mediawiki/index.php?title=Working\\_Group\\_10](http://www.imagwiki.org/mediawiki/index.php?title=Working_Group_10). Accessed 2010.
7. Nielsen, K., P.G. Sørensen, F. Hynne, and H.G. Busse, *Sustained oscillations in glycolysis: an experimental and theoretical study of chaotic and complex periodic behavior and of quenching of simple oscillations*. Biophysical Chemistry, 1998. **72**(1-2): p. 49-62.
8. Chassagnole, C., N. Noisommit-Rizzi, J.W. Schmid, K. Mauch, and M. Reuss, *Dynamic modeling of the central carbon metabolism of Escherichia coli*. Biotechnology and Bioengineering, 2002. **79**(1): p. 53-73.
9. *NIH Roadmap - Overview*. <http://nihroadmap.nih.gov/overview.asp>. Accessed 2010.
10. Frank, O., *Die grundform des arteriellen pulses*. Zeitschrift für Biologie, 1899. **37**: p. 483-526.
11. Ideker, T., T. Galitski, and L. Hood, *A New Approach to Decoding Life: Systems Biology*. Annual Review of Genomics and Human Genetics, 2001. **2**(1): p. 343-372.
12. Hodgkin, A.L. and A.F. Huxley, *A quantitative description of ion currents and its applications to conduction and excitation in nerve membranes*. Journal of Physiology, 1952. **117**(4): p. 500-544.
13. Ekins, S., J. Mestres, and B. Testa, *In silico pharmacology for drug discovery*:

- methods for virtual ligand screening and profiling*. British Journal of Pharmacology, 2007. **152**(1): p. 9.
14. Guyton, A.C. and J.E. Hall, *Textbook of Medical Physiology*. 11th ed. 2006, Philadelphia: Elsevier Saunders.
  15. Rideout, V.C., *Mathematical and computer modeling of physiological systems*. 1991.
  16. Sagawa, K., R.K. Lie, and J. Schaefer, *Translation of Otto Frank's paper "Die Grundform des Arteriellen Pulses" Zeitschrift fur Biologie 37: 483-526 (1899)*. J Mol Cell Cardiol, 1990. **22**(3): p. 253-77.
  17. Wesseling, K.H., J.R. Jansen, J.J. Settels, and J.J. Schreuder, *Computation of aortic flow from pressure in humans using a nonlinear, three-element model*. Journal of Applied Physiology, 1993. **74**(5): p. 2566-73.
  18. *Anesoft Continuing Medical Education and Simulation Software*. <http://www.anesoft.com/>. Accessed 2010.
  19. *Advanced Simulation - Aviation and Body Simulation Technologies*. <http://www.advsim.com/>. Accessed 2010.
  20. Bergman, R.N., Y.Z. Ider, C.R. Bowden, and C. Cobelli, *Quantitative estimation of insulin sensitivity*. American Journal of Physiology- Gastrointestinal and Liver Physiology, 1979. **236**(6): p. 667.
  21. Neal, M.L. and J.B. Bassingthwaighe, *Subject-specific model estimation of cardiac output and blood volume during hemorrhage*. Cardiovascular Engineering, 2007. **7**(3): p. 97-120.
  22. Zenker, S., J. Rubin, and G. Clermont, *From inverse problems in mathematical physiology to quantitative differential diagnoses*. PLoS Computational Biology, 2007. **3**(11): p. e204.
  23. *Integrative Physiology: Highly-integrated\_human:model\_index [Model Wiki]*. [http://bioeng.washington.edu/model/doku.php?id=Integrative\\_Physiology:Highly-integrated\\_human:model\\_index](http://bioeng.washington.edu/model/doku.php?id=Integrative_Physiology:Highly-integrated_human:model_index). Accessed 2010.
  24. *The Virtual Soldier Project*. <http://www.virtualsoldier.us/>. Accessed 2010.
  25. Yngve, G., *Visualization for Biological Models, Simulation, and Ontologies*, Ph.D. Dissertation. University of Washington. 2007.
  26. Bassingthwaighe, J.B., *Strategies for the physiome project*. Annals of Biomedical Engineering, 2000. **28**(8): p. 1043-58.



27. Hunter, P.J. and T.K. Borg, *Integration from proteins to organs: the Physiome Project*. Nature Reviews Molecular Cell Biology, 2003. 4(3): p. 237-43.
28. Arts, T., T. Delhaas, P. Bovendeerd, X. Verbeek, and F.W. Prinzen, *Adaptation to mechanical load determines shape and properties of heart and circulation: the CircAdapt model*. American Journal of Physiology - Heart and Circulatory Physiology, 2005. 288(4): p. H1943-54.
29. Beard, D.A., *A biophysical model of the mitochondrial respiratory system and oxidative phosphorylation*. PLoS Computational Biology, 2005. 1(4).
30. Kerckhoffs, R.C., M.L. Neal, Q. Gu, J.B. Bassingthwaite, J.H. Omens, and A.D. McCulloch, *Coupling of a 3D finite element model of cardiac ventricular mechanics to lumped systems models of the systemic and pulmonary circulation*. Annals of Biomedical Engineering, 2007. 35(1): p. 1-18.
31. Lu, K., J.W. Clark, F. Ghorbel, D. Ware, and A. Bidani, *A human cardiopulmonary system model applied to the analysis of the Valsalva maneuver*. American Journal of Physiology - Heart and Circulatory Physiology, 2001. 281(6): p. H2661-H2679.
32. Winslow, R.L., J. Rice, S. Jafri, E. Marban, and B. O'Rourke, *Mechanisms of altered excitation-contraction coupling in canine tachycardia-induced heart failure, II: model studies*. Circulation Research, 1999. 84(5): p. 571-86.
33. *BioModels Database*. <http://www.ebi.ac.uk/biomodels-main/>. Accessed 2010.
34. *NSR Physiome Models*. <http://www.physiome.org/Models/>. Accessed 2010.
35. *CellML Model Repository*. <http://www.cellml.org/models>. Accessed 2010.
36. *MATLAB Central File Exchange*.  
<http://www.mathworks.com/matlabcentral/fileexchange/loadCategory.do>. Accessed 2010.
37. Neal, M.L., J.H. Gennari, T. Arts, and D.L. Cook, *Advances in semantic representation for multiscale biosimulation: A case study in merging models*. Proceedings of the Pacific Symposium on Biocomputing, 2009: p. 304-315.
38. *Community - SBML.org*. <http://sbml.org/Community>. Accessed 2010.
39. *MATLAB Central*. <http://www.mathworks.com/matlabcentral/>. Accessed 2010.
40. *BioModels Database: A Database of Annotated Published Models*.  
<http://www.ebi.ac.uk/biomodels/>. Accessed 2010.

41. Bornstein, B.J., S.M. Keating, A. Jouraku, and M. Hucka, *LibSBML: an API Library for SBML*. *Bioinformatics*, 2008. **24**(6): p. 880.
42. *CellML API - CellML*. <http://www.cellml.org/tools/api>. Accessed 2010.
43. *JSim Home Page*. <http://physiome.org/jsim/index.html>. Accessed 2010.
44. Sanchez, R. and J.T. Mahoney, *Modularity, flexibility, and knowledge management in product and organization design*. *Strategic Management Journal* 1996. **17**(Winter 1996): p. 63-76.
45. Krause, F., J. Uhlenndorf, T. Lubitz, M. Schulz, E. Klipp, and W. Liebermeister, *Annotation and merging of SBML models with semanticSBML*. *Bioinformatics*, 2010. **26**(3): p. 421.
46. Baldwin, C.Y. and K.B. Clark, *Design Rules Volume I: The Power of Modularity*. 2000, Cambridge: The MIT Press.
47. Bartholet, R.G., P.F. Reynolds, D.C. Brogan, and J.C. Carnahan, *In search of the philosopher's stone: Simulation composability versus component-based software design*. *Proceedings of The Fall Simulation Interoperability Workshop*, 2004.
48. Szyperski, C., *Component Software: Beyond Object-oriented Programming*. 2nd ed. 2002: Addison Wesley.
49. Carnahan, J., P. Reynolds, and D. Brogan, *Simulation-specific characteristics and software reuse*. *Proceedings of The Winter Simulation Conference*, 2005: p. 2492-2499.
50. Dahmann, J.S., R.M. Fujimoto, and R.M. Weatherly, *The Department of Defense High Level Architecture*. *Proceedings of the Winter Simulation Conference*, 1997: p. 142-149.
51. *B2 Spice Software V5.0 | Electronics Design Software | RD Research*. <http://www.spice-software.com/>. Accessed 2010.
52. Vachharajani, M., N. Vachharajani, and D.I. August, *The Liberty Structural Specification Language: A high-level modeling language for component reuse*. *Proceedings of the ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI)*, 2004: p. 195-206.
53. Diaz-Calderon, A., C.J.J. Paredis, and P.K. Khosla, *A modular composable software architecture for the simulation of mechatronic systems*. *Proceedings of the ASME Computers in Engineering Conference*, 1998.

54. Maxwell, T. and R. Costanza, *An Open Geographic Modeling Environment*. SIMULATION, 1997. **68**(3): p. 175-185.
55. Gennari, J.H., M.L. Neal, B.E. Carlson, and D.L. Cook, *Integration of multi-scale biosimulation models via light-weight semantics*. Proceedings of the Pacific Symposium on Biocomputing, 2008. **13**: p. 414-25.
56. Schulz, M., J. Uhlenhof, E. Klipp, and W. Liebermeister, *SBMLmerge, a system for combining biochemical network models*. Genome Informatics, 2006. **17**(1): p. 62-71.
57. Kasputis, S. and H.C. Ng, *Composable simulations*. Proceedings of the Winter Simulation Conference, 2000. **2**: p. 1577-1584.
58. Davis, P.K. and R.H. Anderson, *Improving the Composability of Department of Defense Models and Simulations*. 2003, Rand Corporation: Santa Monica, CA.
59. Tolk, A., S.Y. Diallo, and C.D. Tunista, *Applying the Levels of Conceptual Interoperability Model in Support of Integratability, Interoperability, and Composability for System-of-Systems Engineering*. Journal of Systemics, Cybernetics and Informatics, 2008. **5**(5): p. 65-74.
60. Petty, M.D. and E.W. Weisel, *A Composability Lexicon*, in *The Spring 2003 Simulation Interoperability Workshop*. 2003: Orlando, FL.
61. Brogan, D., P. Reynolds, R. Bartholet, J. Carnahan, and Y. Loitiere, *Semi-automated Simulation Transformation for DDDAS*, in *Computational Science - ICCS 2005*. 2005, Springer: Heidelberg. p. 721-728.
62. Rosse, C. and J.L.V. Mejino, *A Reference Ontology for Bioinformatics: The Foundational Model of Anatomy*. Journal of Biomedical Informatics, 2003. **36**: p. 478-500.
63. Degtyarenko, K., P. Matos, M. Ennis, J. Hastings, M. Zbinden, A. McNaught, R. Alcantara, M. Darsow, M. Guedj, and M. Ashburner, *ChEBI: a database and ontology for chemical entities of biological interest*. Nucleic Acids Research, 2007.
64. *Virtual Physiological Human - Latest news*. <http://www.vph-noe.eu/home>. Accessed 2010.
65. Whitmore, I., *Terminologia anatomica: new terminology for the new anatomist*. The Anatomical Record Part B: The New Anatomist, 1999. **257**(2): p. 50-53.
66. Haendel, M., F. Neuhaus, D. Sutherland, J.L.E. Mejino, Jr., C. Mungall, and B. Smith, *The Common Anatomy Reference Ontology*, in *Anatomy Ontologies for*

- Bioinformatics: Principles and Practice*, A. Burger, D. Davidson, and R. Baldock, Editors. 2007, Springer: New York.
67. Osborne, J., J. Flatow, M. Holko, S. Lin, W. Kibbe, L. Zhu, M. Danila, G. Feng, and R. Chisholm, *Annotating the human genome with Disease Ontology*. BMC Genomics, 2009. **10**(Suppl 1): p. S6.
  68. Robinson, P.N., S. Kohler, S. Bauer, D. Seelow, D. Horn, and S. Mundlos, *The Human Phenotype Ontology: a tool for annotating and analyzing human hereditary disease*. The American Journal of Human Genetics, 2008. **83**(5): p. 610-615.
  69. Bard, J., S. Rhee, and M. Ashburner, *An ontology for cell types*. Genome Biology, 2005. **6**(2): p. R21.
  70. Sioutos, N., S. Coronado, M.W. Haber, F.W. Hartel, W.L. Shaiu, and L.W. Wright, *NCI Thesaurus: a semantic model integrating cancer-related clinical and molecular information*. Journal of Biomedical Informatics, 2007. **40**(1): p. 30-43.
  71. Cook, D.L., J.V. Mejino Jr, M.L. Neal, and J.H. Gennari, *Bridging Biological Ontologies and Biosimulation: The Ontology of Physics for Biology*. Proceedings of the American Medical Informatics Association Annual Symposium, 2008: p. 136-140.
  72. Smith, B., M. Ashburner, C. Rosse, J. Bard, W. Bug, W. Ceusters, L.J. Goldberg, K. Eilbeck, A. Ireland, and C.J. Mungall, *The OBO Foundry: coordinated evolution of ontologies to support biomedical data integration*. Nature Biotechnology, 2007. **25**(11): p. 1251-1255.
  73. Smith, B., W. Ceusters, B. Klagges, J. Köhler, A. Kumar, J. Lomax, C. Mungall, F. Neuhaus, A. Rector, and C. Rosse, *Relations in biomedical ontologies*. Genome Biology, 2005. **6**(5): p. R46.
  74. Gkoutos, G., E. Green, A.M. Mallon, J. Hancock, and D. Davidson, *Using ontologies to describe mouse phenotypes*. Genome Biology, 2004. **6**(1): p. R8.
  75. Mungall, C., G. Gkoutos, C. Smith, M. Haendel, S. Lewis, and M. Ashburner, *Integrating phenotype ontologies across multiple species*. Genome Biology, 2010. **11**(1): p. R2.
  76. McGuinness, D.L. and F. Van Harmelen, *OWL web ontology language overview*. W3C recommendation, 2004. **10**: p. 2004-03.
  77. Detwiler, L.T., J.L.V. Mejino Jr, C. Rosse, and J.F. Brinkley, *Efficient Web-based navigation of the Foundational Model of Anatomy*. Proceedings of the American Medical Informatics Association Annual Symposium, 2003: p. 829.

78. Noy, N.F. and M.A. Musen, *The PROMPT suite: Interactive tools for ontology merging and mapping*. International Journal of Human-Computer Studies, 2003. **59**(6): p. 983-1024.
79. *OWL API*. <http://owlapi.sourceforge.net/>. Accessed 2010.
80. *JDOM*. <http://www.jdom.org/>. Accessed 2010.
81. Heer, J., S.K. Card, and J.A. Landay, *Prefuse: a toolkit for interactive information visualization*. Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, 2005: p. 421-430.
82. *JUNG - Java Universal Network/Graph Framework*. <http://jung.sourceforge.net/index.html>. Accessed 2010.
83. Shaw, M., L.T. Detwiler, J.F. Brinkley, and D. Suciu, *Generating application ontologies from reference ontologies*. Proceedings of the American Medical Informatics Association Annual Symposium, 2008: p. 672-676.
84. *SPARQL Query Language for RDF*. <http://www.w3.org/TR/rdf-sparql-query/>. Accessed 2010.
85. *NCBO BioPortal: Welcome to the NCBO BioPortal*. <http://bioportal.bioontology.org/>. Accessed 2010.
86. Noy, N.F., N.H. Shah, P.L. Whetzel, B. Dai, M. Dorf, N. Griffith, C. Jonquet, D.L. Rubin, M.A. Storey, and C.G. Chute, *BioPortal: ontologies and integrated data resources at the click of a mouse*. Nucleic Acids Research, 2009. **37**(Web Server issue): p. W170.
87. *NCBO BioPortal: Electrocardiography (ECG) Ontology*. <http://bioportal.bioontology.org/ontologies/39893>. Accessed 2010.
88. Guyton, A.C., T.G. Coleman, and H.J. Granger, *Circulation: overall regulation*. Annual Review of Physiology, 1972. **34**(1): p. 13-44.
89. Girvan, M. and M.E.J. Newman, *Community structure in social and biological networks*. Proceedings of the National Academy of Sciences, 2002. **99**(12): p. 7821-7826.
90. Noy, N.F. and M.A. Musen, *PROMPT: Algorithm and tool for automated ontology merging and alignment*. Proceedings of the Seventeenth National Conference on Artificial Intelligence (AAAI-2000), 2000: p. 450--455.

91. Courtot, M., F. Gibson, A.L. Lister, J. Malone, D. Schober, R.R. Brinkman, and A. Ruttenberg, *MIREOT: the Minimum Information to Reference an External Ontology Term*. Available from Nature Precedings: <http://hdl.handle.net/10101/npre.2009.3574.1>, 2009.
92. *W3C Math Home*. <http://www.w3.org/Math/>. Accessed 2010.
93. Schadow, G., C.J. McDonald, J.G. Suico, U. Fähring, and T. Tolxdorff, *Units of measure in clinical information systems*. *Journal of the American Medical Informatics Association*, 1999. 6(2): p. 151.
94. *Ontology Detail: Units of measurement*. <http://www.obofoundry.org/cgi-bin/detail.cgi?id=unit>. Accessed 2010.
95. *Measurement Units Ontology - Morfeo Wiki*. [http://forge.morfeo-project.org/wiki\\_en/index.php/Units\\_of\\_measurement\\_ontology](http://forge.morfeo-project.org/wiki_en/index.php/Units_of_measurement_ontology). Accessed 2010.
96. *Ricordo*. <http://www.ricordo.eu/>. Accessed 2010.

## **VITA**

Maxwell Lewis Neal was born in Cleveland, Ohio and studied Biology at Case Western Reserve University. He began his biological research career by exploring the ecology of fossil tubeworms, before migrating into the field of biomedicine. In 2002 he moved to Seattle and learned about physiological modeling while working on the Virtual Soldier Project. His experience on this project inspired him to pursue research that would advance the field of biosimulation. In 2010 he earned his Doctor of Philosophy from the University of Washington's Biomedical and Health Informatics program.