CENTER FOR REPRODUCIBLE BIOMEDICAL MODELING

Biannual Newsletter

VOL.2 JAN 2023

Welcome, new PIs!

EDITED BY JANIS SHIN

The Center has hired three new PIs: Eran Agmon, Elebeoba "Chi-Chi" May, and Joseph Hellerstein. We've asked them a few questions and we hope you have as much fun getting to know them as much as we did!

Our newest PIs at the Center for Reproducible Biological Models. Eran Agmon (left), Elebeoba May (center), and Joseph Hellerstein (right).

Eran Agmon

Eran Agmon is a new Assistant Professor for Cellular Analysis and Modeling at the University of Connecticut Health Center. He does research on computational methods for multi-scale modeling of whole cells – from their molecular underpinnings, to their integrated functions, to the ecological level with many heterogeneous cells interacting in a shared environment. The next generation of computational biology models will need to combine multi-source and multi-level data with multi-algorithmic representations of diverse biological mechanisms. These simulations will be used to interpret datasets, make experimental predictions, identify medical solutions, and help us understand fundamental principles of biological organization.

To meet this need, Eran led the development of Vivarium – an open-source model integration tool that supports flexible model development and makes it easier for scientists to define any imaginable mechanistic model, combine it with existing models, and execute them together as an integrated simulation. It provides an interface that makes individual simulation tools into modules that can be wired together, parallelized across multiple CPUs, and simulated across many spatial and temporal scales.

Elebeoba "Chi-Chi" May

Dr. Elebeoba E. May is an Associate Professor in the Department of Medical Microbiology and Immunology and a Wisconsin Institute of Discovery (WID) faculty. As director of the Multi-scale Immunobiology Design, Algorithms, and Simulation (MIDAS) Lab, Dr. May's research

Welcome, new PIs! (continued)

focuses on the design of integrated quantitative and empirical platforms for the development of multi-scale, predictive models of biological systems. The emphasis of her work is on addressing challenges in the areas of chronic infection and disease, biodefense, and biomanufacturing. Additional research thrust areas draw on coding theory, information and communication theory to model and investigate fundamental biological processes such as gene expression and microbial communication-enabled adaptation. The lab's experimental work focuses on quantifying host response to infection using various agents and characterizing microbial interactions and response to combinatorial stress using *E. coli* as a model system. The May lab also works on expanding models of TB granuloma formation and macrophage infection response to understand the impact of comorbidities, such as micronutrient (Vitamin D) deficiency, alcohol use, and COPD on infection response, conditions that disproportionately impact minority communities. Dr. May is a recipient of an NIH/NHLBI K25 Quantitative Research Career Development Award, the NSF Director's Award for Superior Accomplishment, and Women of Color Research Sciences and Technology award.

Joseph Hellerstein

Joseph Hellerstein is a Data Scientist, Affiliate Professor of Computer Science and Engineering, and Senior Data Science Fellow at the eScience Institute within the University of Washington. His research and teaching focus on control analysis and design of computational and biological systems, and model engineering. This involves applying software engineering techniques to the biomedical model development to improve their credibility and reuse. His work on control engineering of computing systems has been used in products at IBM, Microsoft, and Google, and has had thousands of research citations. Since joining the University of Washington, he has focused on incorporating technologies used in software engineering into the development of biomedical models. Some projects are: SBMLLint, a system for detecting and isolating mass balance errors in biomedical models; VSCode-Antimony, a smart editor for developing biomedical models; and AMAS, an automated model annotation system. He has also developed/co-developed and taught several courses: "Molecular Biology for Computer Scientists," "Software Development for Data Scientists", "Computational Systems Biology for BioMedical Applications", and "Advanced Controls for Biological Systems."

How did you get involved in systems biology/computational modeling research? What piqued your interest?

EA: As a student I found there is a tendency for biological research to focus narrowly on a specific organismic sub-system, or to take a population-level view that oversimplifies the organism as a list of genes or list of behaviors. Computational systems biology looks at organisms from multiple different perspectives, using the most appropriate mathematical representations. I was drawn to this field to synthesize these different perspectives into a true integrative framework for understanding the role of the organism in biology, starting off with the most minimal organisms – whole bacterial cells.

EM: I was always interested in biology since high school but more from the theoretical perspective. My training (BSc, MS and PhD) is in Computer Engineering and I began applying methods from Electrical and Computer Engineering to modeling molecular biology in graduate school. So the opportunity to merge computation and biology in graduate school was

Welcome, new PIs! (continued)

very much in line with my interest. My entry into systems biology was during my first professional position at Sandia National Laboratories (Albuquerque, NM), where I co-developed BioXyce—a large scale biological circuit simulation tool.

JH: Towards the end of my time at Google, I was working on Google Cloud. A part of this was validating some scheduling ideas with customers who had significant computing requirements. The only customers willing to work with our early software were academics, mostly in biology. Our major partner was at the Institute for Systems Biology (ISB), a group doing cancer research. Getting their applications to run on Google Cloud required that I understand something about biology (I had not taken a biology class since high school). After a few months of working with ISB, I decided I was more interested in the biology problems than the computing problems I worked on at Google.

This change in direction required that I acquire considerable background. I started reading up on molecular biology for a while, but I needed something that motivated me more. So, I asked a colleague in UW Computer Science, Ed Lazowska, if I could teach a course on Molecular Biology to CS students, thinking that the fear of "looking like an idiot" would be very motivating. He said "yes" (a testimony to our friendship, not to my knowledge). Soon after this, Ed asked if I was interested in joining the then new eScience Institute. I left Google, and joined eScience.

After I began at UW, I started hunting for interesting research I could do in biology. Someone mentioned that I should talk with a fellow in Bioengineering—Herbert Sauro. We met at the Allen School, and it was clear that we spoke the same language. In particular, we both had a strong interest in mathematical modeling and in control engineering.

It took a few more months, but we eventually found interests to pursue jointly in teaching and research. As our joint interests grew, I had more opportunities to work with Bioengineering students, and eventually, I started supervising their research.

How are you implementing reproducibility in your lab? If you are building a tool or method, what are some key features that you're most excited about or proud about?

EA: Collaborating on large integrative models forces everyone involved to consider reproducibility, since they need to provide components that others can use and trust. In all my collaborations, I promote scalable software engineering practices such as continuous integration, unit testing, semantic versioning, and documentation. I believe that if computational biologists adopted some of these well-established practices, we would be able to build more extensible and robust systems.

How do reproducible models (or lack thereof) affect the work done in your lab?

EM: My lab builds molecular to cellular scale models. So building reproducible models is important to our work, both as model developers and as model users. Many times we use published models or components of published models as starting points, so the reproducibility of the model directly impacts how difficult it will be to reuse or extend models. Also for our multi-scale models that incorporate molecular/biochemical network models, reproducibility at all scales is important.

Welcome, new PIs! (continued)

Outside of work, what do you do for fun?

EA: I enjoy reading nonfiction books about the history of science, getting lost on a walk either in a city or in nature, and the rest of my time is spent with my family.

EM: My joy is enjoying time with my family watching Marvel movies, playing Minecraft or just being in the same time zone. I also enjoy reading various types of novels.

JH: My wife & I love the Pacific Northwest (PNW), especially during the summer. We're big on the usual PNW activities: hiking, biking, kayaking. We also have a very demanding cat (although "demanding cat" may be redundant).

Quick Tutorials

IRIDIUM

BY HERBERT SAURO

Iridium Modeling Tool: Many modeling tools fall into two camps: those that embrace a purely scripting solution, such as Python, and those that provide a graphical user interface experience, such as COPASI or VCell. The objective of writing Iridium was to see if we could create a hybrid between these two extremes that uses the antimony language to describe models but uses a GUI to control simulations. We called this new application Iridium. The application is available for Windows and Mac users (Intel or Arm) and can be downloaded as binaries in a zip file from the GitHub repository. All the source code is of course open source and available at the GitHub repo (sys-bio/IridiumSimulator). The latest release is August 11st, 2022 (A new windows version that fixes two issues is available in a December 2022 release). To try it out, navigate to the releases section and download the appropriate platform binary zip file. Unpack the zip file to any convenient place on your hard drive. To launch Iridium, click on the executable file called roadrunnerUI.exe.

Iridium supports time-course simulation, as well as the ability to compute the steady-state. In both cases, users have the ability to do simple one or 2D parameter scans either in linear or log space. In addition, users can select a variety of outputs to observe during a simulation including concentrations, fluxes, elasticities, and eigenvalues.

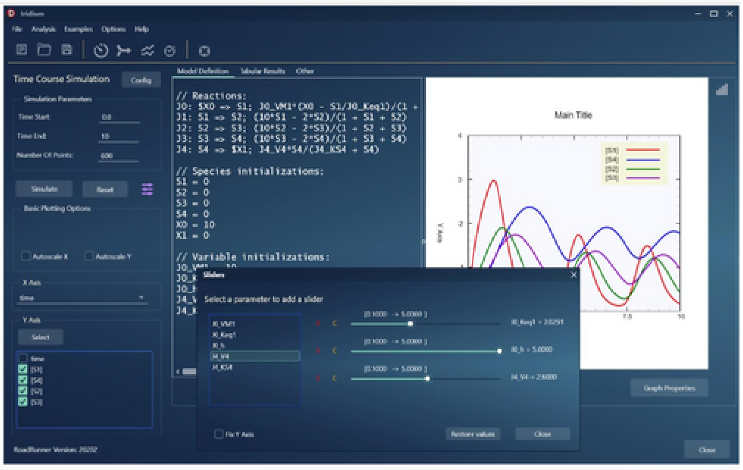
Under the steady-state option, the users can compute control coefficients and elasticities from metabolic control analysis as well as other measures such as the Jacobian matrix and eigenvalues. Calculations such as these are obtained by Iridium running libroadrunner in the background. One unique feature of Iridium is the ability to do real-time simulation. This means that when a user runs a model in time, the user can interact with the model by changing sliders and observe the immediate effect on the model dynamics.

A separate control is provided to allow the user to obtain more specialist outputs such as the stoichiometric matrix, as well as the link and conservation matrix.

Iridium comes with a number of built-in examples, allowing users to quickly try out models. However, users can also import standard SBML (for example, models from BioModels) or standard antimony text files. Iridium also has a fairly flexible graphing capability where users can control many aspects of the visual look of a plot as well as export the plot to PDF so that it can be used in publications. As well as graphical plots, Iridium can also produce raw simulation data, which can be exported for use in other applications such as Excel or Python. There are also some more trivial features, such as the ability to have a dark-style look to the user interface.

Possible enhancements for 2023 include 3D bar graph viewing of 2D data especially control coefficients which find use in metabolic engineering studies.

Quick Tutorials (continued)



```
// Reactions:
J0: $X0 => S1; J0_VM1*(X0 - S1/J0_Keq1)/(1 +
J1: S1 => S2; (10*S1 - 2*S2)/(1 + S1 + S2)
J2: S2 => S3; (10*S2 - 2*S3)/(1 + S2 + S3)
J3: S3 => S4; (10*S3 - 2*S4)/(1 + S3 + S4)
J4: S4 => $X1; J4_V4*S4/(J4_KS4 + S4)

// Species initializations:
S1 = 0
S2 = 0
S3 = 0
S4 = 0
X0 = 10
X1 = 0

// Variable initializations:
```

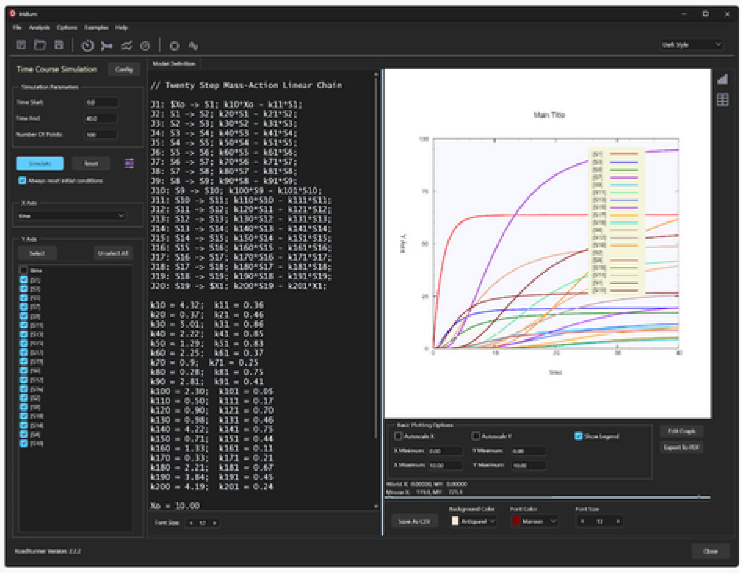
Sliders

Select a parameter to add a slider

J0_Keq1 = 2.0291

J0_h = 5.0000

J4_V4 = 2.6000



```
// Twenty Step Mass-Action Linear Chain

J1: $Xo -> S1; k10*Xo - k11*S1;
J2: S1 -> S2; k20*S1 - k21*S2;
J3: S2 -> S3; k30*S2 - k31*S3;
J4: S3 -> S4; k40*S3 - k41*S4;
J5: S4 -> S5; k50*S4 - k51*S5;
J6: S5 -> S6; k60*S5 - k61*S6;
J7: S6 -> S7; k70*S6 - k71*S7;
J8: S7 -> S8; k80*S7 - k81*S8;
J9: S8 -> S9; k90*S8 - k91*S9;
J10: S9 -> S10; k100*S9 - k101*S10;
J11: S10 -> S11; k110*S10 - k111*S11;
J12: S11 -> S12; k120*S11 - k121*S12;
J13: S12 -> S13; k130*S12 - k131*S13;
J14: S13 -> S14; k140*S13 - k141*S14;
J15: S14 -> S15; k150*S14 - k151*S15;
J16: S15 -> S16; k160*S15 - k161*S16;
J17: S16 -> S17; k170*S16 - k171*S17;
J18: S17 -> S18; k180*S17 - k181*S18;
J19: S18 -> S19; k190*S18 - k191*S19;
J20: S19 -> $X1; k200*S19 - k201*X1;

k10 = 4.32;  k11 = 0.36
k20 = 0.37;  k21 = 0.46
k30 = 5.01;  k31 = 0.86
k40 = 2.22;  k41 = 0.85
k50 = 1.29;  k51 = 0.83
k60 = 2.25;  k61 = 0.37
k70 = 0.9;  k71 = 0.25
k80 = 0.28;  k81 = 0.75
k90 = 2.81;  k91 = 0.41
k100 = 2.30;  k101 = 0.05
k110 = 0.50;  k111 = 0.37
k120 = 0.90;  k121 = 0.70
k130 = 0.98;  k131 = 0.46
k140 = 4.22;  k141 = 0.75
k150 = 0.71;  k151 = 0.44
k160 = 1.33;  k161 = 0.11
k170 = 0.33;  k171 = 0.21
k180 = 2.21;  k181 = 0.67
k190 = 3.84;  k191 = 0.45
k200 = 4.19;  k201 = 0.24

Xo = 10.00
```

Modeling a feedback oscillator with a slider panel for interaction modeling (top), Dark mode showing simulation of a larger model described using the Antimony format (bottom).

Moreover, if time permits, we hope to add a plugin interface that allows other users to add new functionality via Python plugins. Possible extensions using the python plugin system include the ability of a user to automatically export a reproducible model package in the form of a COMBINE archive.

The tool is currently being tested by one of our collaborators.

SBMLDIAGRAMS

BY LILLIAN TATKA

The Sauro Lab at the University of Washington has recently released the python package SBMLDiagrams for visualizing reaction networks [1]. Developed by postdoctoral scholar Jin Xu, SBMLDiagrams simplifies the process of creating visualizations of networks encoded in the Systems Biology Markup Language (SBML) [2].

SBML level 3 currently supports layout and render information, with layout describing the size and positioning of elements such as compartments, species, and reactions and render information describing color and shape information [3]. However, it can be difficult for new users to acquire the extensive knowledge needed to successfully use these tools

SBMLDiagrams allows users to easily read, write, create, and visualize layout and render specifications for biochemical models without an extensive understanding of the underlying model object. A user can simply load an SBML model and use the SBMLDiagrams autolayout feature to easily create a visualization of the network. The user can also customize the layout and render by specifying text, size, positioning and color and shape information of nodes. The visualizations can be exported as PNG, JPG, or PDF files.

SBMLDiagrams is available as a pip package (pip install SBMLDiagrams). Source code and documentation can be found at github.com/sys-bio/SBMLDiagrams and sys-bio.github.io/SBMLDiagrams.

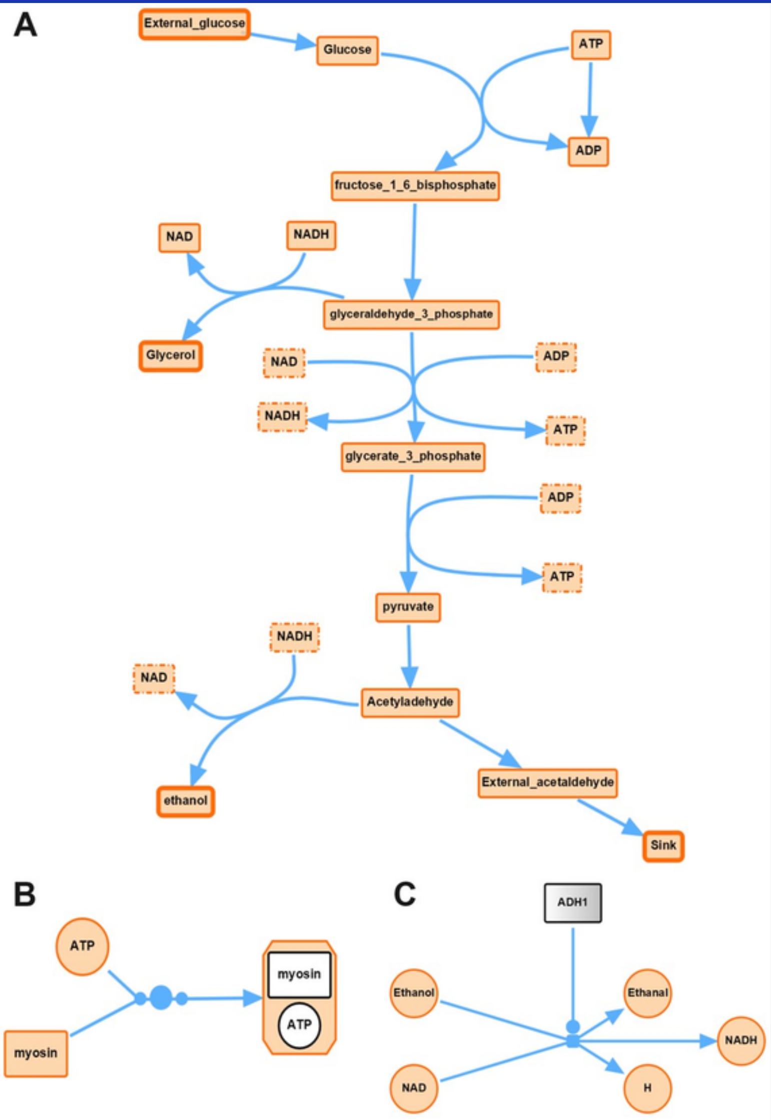
[1] Xu, J., Jiang, J., Sauro H. SBMLDiagrams: a python package to process and visualize SBML layout and render. *Bioinformatics.* (2022)

[2] Hucka, M., Finney, A., Sauro, H., Bolouri, H., Doyle, J., Kitano, H., Arkin, A., Bornstein, B., Bray, D., CornishBowden, A. & Others The systems biology markup language (SBML): a medium for representation and exchange of biochemical network models. *Bioinformatics.* 19, 524-531 (2003)

[3] Deckard, A., Bergmann, F. & Sauro, H. Supporting the SBML layout extension. Bioinformatics. 22, 2966-2967 (2006)

[4] Wolf, J., Sohn, H., Heinrich, R. & Kuriyama, H. Mathematical analysis of a mechanism for autonomous metabolic oscillations in continuous culture of Saccharomyces cerevisiae. FEBS Letters. 499, 230-234 (2001)

Quick Tutorials (continued)



Some visualization examples by SBMLDiagrams. (A) Using SBMLDiagrams to visualize a model of glycolysis (Wolf et al., 2001). Alias nodes are indicated with dashed border lines and boundary nodes have a thicker border width than floating nodes. An animation is also available on GitHub (https://github.com/sys-bio/SBMLDiagrams). (B) Interface to SBGN with a complex species. (C) Interface to SBGN with a gradient node [4]

ANTIMONY: MAKING EXCHANGEABLE MODELS USING SBML

BY LUCIAN SMITH AND HERBERT SAURO

SBML[1] is the de facto language for exchanging models of biochemical systems. However, SBML is expressed using XML which is easier for computers to read and write. The XML however is not meant to be read and certainly not written by humans. As a result, developers have created tools to allow users to read and write SBML using easier formats. These can include spreadsheet-like inputs such as provided by COPASI or visual network editors such as CellDesigner or PathwayDesigner. The center, through support from the NSF, has continued to enhance its own inhouse format called Antimony [2]. This is a text based language that makes it very easy to create, understand, and edit SBML models. We provide a converter library (including a Python package) to convert Antimony models to and from SBML, making creating and editing SBML models simpler and more comprehensible.

In this article we will provide some examples of Antimony to show you how to use it to read and write SBML. Let's start with the simple case, the conversion of citrate to isocitrate catalyzed by aconitase in the Krebs cycle. In a textbook this would be usually be written as a graphic such as:

Citrate → Isocitrate

If we want to express this using Antimony we could write something very similar:

```
Citrate -> Isocitrate
```

We can't use a special graphic symbol for a reaction so instead we just use a dash and a greater than symbol next to each other, '->'. Because SBML is used to build kinetic models we also have the option to specify a kinetic rate law. To do that we just write the kinetic rate equation immediately after the reaction. For example:

```
Citrate -> Isocitrate; Vm*Citrate/(Km + Citrate)
```

Note the semicolon that separates the reaction and the rate equation. The rate equation we used is the classic Briggs-Haldane rate law but we could use any equation we wanted.

If you have more than one reaction you can just list them one after the other to build up a complete pathway. For example, the next reaction in the Krebs cycle is isocitrate dehydrogenase. This involves multiple reactants (NAD) and products (NADH, CO2). In Antimony we would write such as reaction in a way that shouldn't be too surprising:

```
Isocitrate + NAD -> Ketoglutarate + NADH + CO2; k1*Isocitrate*NAD
```

For the rate law we used a simple mass-action law to save some space. One other useful piece to include in a reaction it's its name. Often this can be the name of the enzyme but it doesn't have to be. For example:

Quick Tutorials (continued)

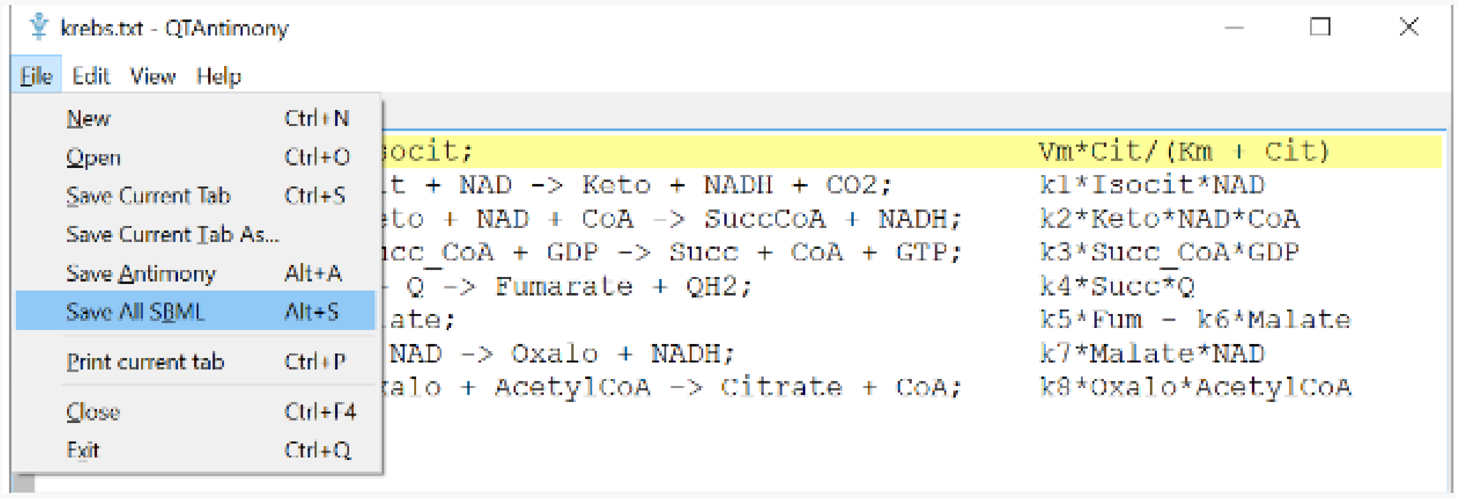
```
Aconitase: Citrate -> Isocitrate; Vm*Citrate/(Km + Citrate)
```

To add the name we just prefix the reaction with the name separated by a colon symbol. Let's complete the Krebs cycle using the Anthony format as shown below. Note that, as with all names, we can't use spaces in a name. Below we've used the underscore symbol instead:

```
Aconitase: Cit -> Isocit;       Vm*Cit/(Km + Cit)
IsoCitrate_Dh: Isocit + NAD -> Keto + NADH + CO2;  k1*Isocit*NAD
Ketoglutarate_Dh: Keto + NAD + CoA -> SuccCoA + NADH; k2*Keto*NAD*CoA
SuccCoA_Synthase: Succ_CoA + GDP -> Succ + CoA + GTP; k3*Succ_CoA*GDP
Succinate_Dh: Succ + Q -> Fumarate + QH2;        k4*Succ*Q
Fumarase: Fum -> Malate;             k5*Fum - k6*Malate
Malate_Dh: Malate + NAD -> Oxalo + NADH;       k7*Malate*NAD
Citrate_synthase: Oxalo + AcetylCoA -> Citrate + CoA; k8*Oxalo*AcetylCoA
```

Once you have your model written out like this, how can you convert it into SBML? There are a few ways you can do this:

1. Use QTAntimony (available as an executable for all platforms at https://github.com/sysbio/antimony/releases)
2. Use Tellurium (available in Python with pip install tellurium)
3. VSCode plugin (to be released by end of January/February 2023)
4. Online converter (to be released in spring of 2023)



```
ocit;                                        Vm*Cit/(Km + Cit)
t + NAD -> Keto + NADH + CO2;                k1*Isocit*NAD
eto + NAD + CoA -> SuccCoA + NADH;           k2*Keto*NAD*CoA
cc_CoA + GDP -> Succ + CoA + GTP;            k3*Succ_CoA*GDP
 Q -> Fumarate + QH2;                        k4*Succ*Q
ate;                                         k5*Fum - k6*Malate
NAD -> Oxalo + NADH;                         k7*Malate*NAD
alo + AcetylCoA -> Citrate + CoA;            k8*Oxalo*AcetylCoA
```

The screenshot above is from QTAntimony with the Kreb's cycle model shown. The dropdown File Menu also shows the Save SBML option. Once in SBML format the model can be loaded into many other different tools for simulation, analysis or visualization.

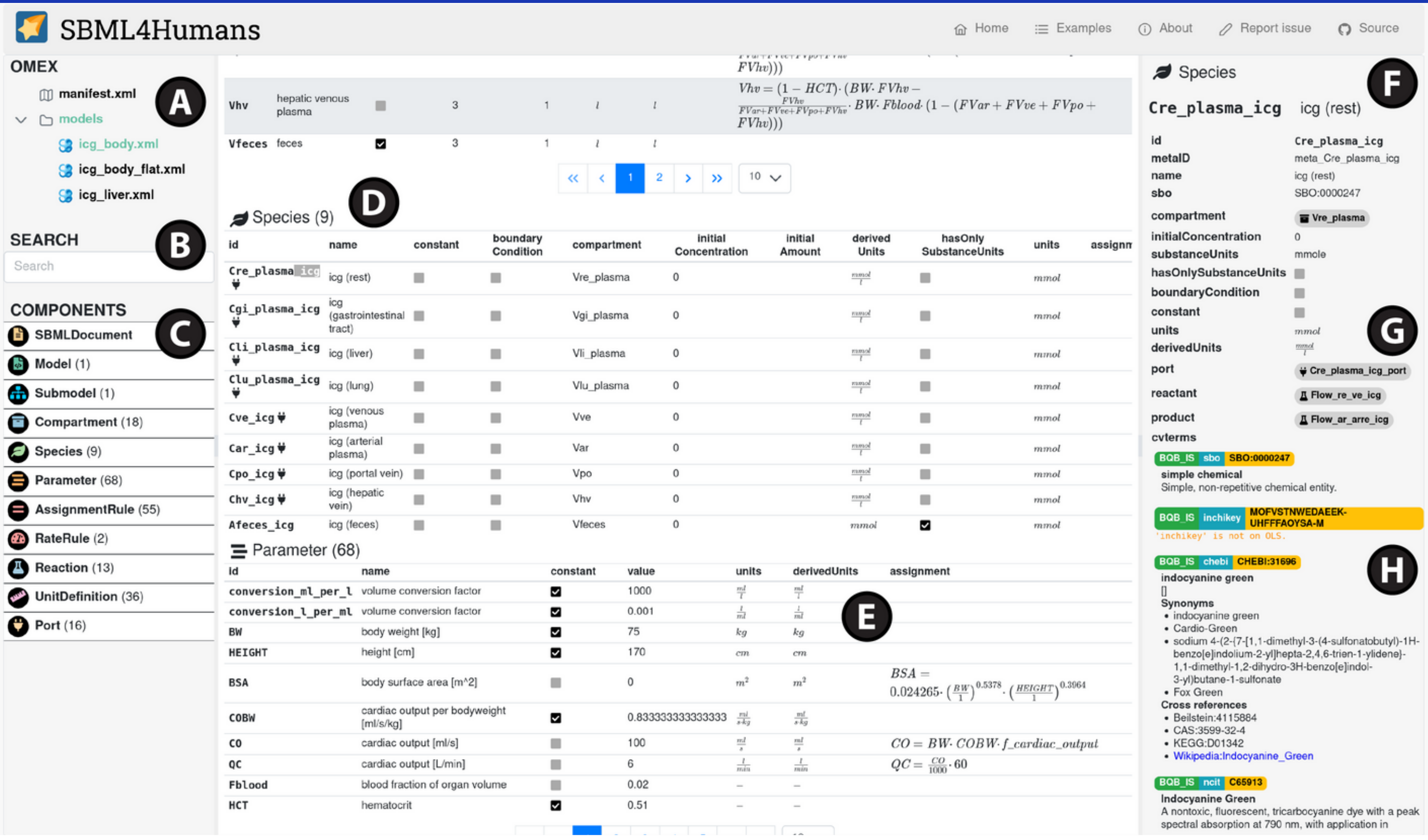
Although SBML is specific to biochemical models, the data model it uses can be interpreted as basic flows and forces. This means that SBML can be used to represent a variety of other model domains, most notably COVID infectious models.

More about Antimony can be found at https://tellurium.readthedocs.io/en/latest/antimony.html, and an online tutorial is available through the Center's YouTube channel, at https://www.youtube.com/watch?v=deVW9HqxkDY.

[1] Hucka M, Finney A, Sauro HM, Bolouri H, Doyle JC, Kitano H, Arkin AP, Bornstein BJ, Bray D, Cornish-Bowden A, Cuellar AA, Dronov S, Gilles ED, Ginkel M, Gor V, Goryanin II, Hedley WJ, Hodgman TC, Hofmeyr JH, Hunter PJ, Juty NS, Kasberger JL, Kremling A, Kummer U, Le Novère N, Loew LM, Lucio D, Mendes P, Minch E, Mjolsness ED, Nakayama Y, Nelson MR, Nielsen PF, Sakurada T, Schaff JC, Shapiro BE, Shimizu TS, Spence HD, Stelling J, Takahashi K, Tomita M, Wagner J, Wang J; SBML Forum. The systems biology markup language (SBML): a medium for representation and exchange of biochemical network models. Bioinformatics. 2003 Mar 1;19(4):524-31. doi: 10.1093/bioinformatics/btg015. PMID: 12611808.

[2] Smith LP, Bergmann FT, Chandran D, Sauro HM. Antimony: a modular model definition language. Bioinformatics. 2009 Sep 15;25(18):2452-4. doi: 10.1093/bioinformatics/btp401. Epub 2009 Jul 3. PMID: 19578039; PMCID: PMC2735663.

Quick Tutorials (continued)



Vhv	hepatic venous plasma	▨	3	1	l	l	$Vhv = (1 - HCT) \cdot (BW \cdot FVhv - \frac{FVhv}{FVar+FVve+FVpo+FVhv} \cdot BW \cdot Fblood \cdot (1 - (FVar + FVve + FVpo + FVhv)))$
Vfeces	feces	✓	3	1	l	l	

Species (9)

id	name	constant	boundary Condition	compartment	initial Concentration	initial Amount	derived Units	hasOnly SubstanceUnits	units	assignm
Cre_plasma_icg	icg (rest)	▨	▨	Vre_plasma	0		$\frac{mmol}{l}$	▨	$mmol$	
Cgi_plasma_icg	icg (gastrointestinal tract)	▨	▨	Vgi_plasma	0		$\frac{mmol}{l}$	▨	$mmol$	
Cli_plasma_icg	icg (liver)	▨	▨	Vli_plasma	0		$\frac{mmol}{l}$	▨	$mmol$	
Clu_plasma_icg	icg (lung)	▨	▨	Vlu_plasma	0		$\frac{mmol}{l}$	▨	$mmol$	
Cve_icg	icg (venous plasma)	▨	▨	Vve	0		$\frac{mmol}{l}$	▨	$mmol$	
Car_icg	icg (arterial plasma)	▨	▨	Var	0		$\frac{mmol}{l}$	▨	$mmol$	
Cpo_icg	icg (portal vein)	▨	▨	Vpo	0		$\frac{mmol}{l}$	▨	$mmol$	
Chv_icg	icg (hepatic vein)	▨	▨	Vhv	0		$\frac{mmol}{l}$	▨	$mmol$	
Afeces_icg	icg (feces)	▨	▨	Vfeces	0		$mmol$	✓	$mmol$	

Parameter (68)

id	name	constant	value	units	derivedUnits	assignment
conversion_ml_per_l	volume conversion factor	✓	1000	$\frac{ml}{l}$	$\frac{ml}{l}$	
conversion_l_per_ml	volume conversion factor	✓	0.001	$\frac{l}{ml}$	$\frac{l}{ml}$	
BW	body weight [kg]	✓	75	kg	kg	
HEIGHT	height [cm]	✓	170	cm	cm	
BSA	body surface area [m^2]	▨	0	m^2	m^2	$BSA = 0.024265 \cdot (\frac{BW}{1})^{0.5378} \cdot (\frac{HEIGHT}{1})^{0.3964}$
COBW	cardiac output per bodyweight [ml/s/kg]	✓	0.833333333333333	$\frac{ml}{s \cdot kg}$	$\frac{ml}{s \cdot kg}$	
CO	cardiac output [ml/s]	▨	100	$\frac{ml}{s}$	$\frac{ml}{s}$	$CO = BW \cdot COBW \cdot f\_cardiac\_output$
QC	cardiac output [L/min]	▨	6	$\frac{l}{min}$	$\frac{l}{min}$	$QC = \frac{CO}{1000} \cdot 60$
Fblood	blood fraction of organ volume	▨	0.02	–	–	
HCT	hematocrit	✓	0.51	–	–	

Species

Cre_plasma_icg icg (rest)

id	Cre_plasma_icg
metaID	meta_Cre_plasma_icg
name	icg (rest)
sbo	SBO:0000247
compartment	Vre_plasma
initialConcentration	0
substanceUnits	mmole
hasOnlySubstanceUnits	▨
boundaryCondition	▨
constant	▨
units	$mmol$
derivedUnits	$\frac{mmol}{l}$
port	Cre_plasma_icg_port
reactant	Flow_re_ve_icg
product	Flow_ar_arre_icg
cvterms	

BQB_IS sbo SBO:0000247
simple chemical
Simple, non-repetitive chemical entity.

BQB_IS inchikey MOFVSTNWEDAEEK-UHFFFAOYSA-M
'inchikey' is not on OLS.

BQB_IS chebi CHEBI:31696
indocyanine green
[]
Synonyms
- indocyanine green
- Cardio-Green
- sodium 4-(2-{7-[1,1-dimethyl-3-(4-sulfonatobutyl)-1H-benzo[e]indolium-2-yl]hepta-2,4,6-trien-1-ylidene}-1,1-dimethyl-1,2-dihydro-3H-benzo[e]indol-3-yl)butane-1-sulfonate
- Fox Green

Cross references
- Beilstein:4115884
- CAS:3599-32-4
- KEGG:D01342
- Wikipedia:Indocyanine_Green

BQB_IS ncit C65913
Indocyanine Green
A nontoxic, fluorescent, tricarbocyanine dye with a peak spectral absorption at 790 nm, with application in

A screenshot of an SBML model parsed with SBML4Humans, showing some of the core functionality provided to help users understand the model and its components. Reproduced with permission from Das and König, 2023

SBML4HUMANS

BY VERONICA PORUBSKY

SBML4Humans is an HTML format designed to provide an interactive and reactive report for SBML models so that humans can easily comprehend the content of a model. This removes the need for a modeler to programmatically parse the model with custom code, and provides an easy to understand overview of the model components. SBML4Humans provides a number of useful features to aid in model comprehension. It allows users to render the model information in an interactive web interface and navigate between the model components, which may include biochemical species, rate laws, and parameters for the underlying mathematical model description. Users can also readily access annotations describing the model components, and even use search functionality to filter the information displayed. In addition to supporting standard SBML model descriptions, SBML4Humans is compatible with the COMBINE archive format, and will generate a human-readable report for models contained in these archives.

While the package supports the most relevant and most frequently used information contained in the SBML core data structure, it also provides support for hierarchical models (comp package), for constraint-based models (fbc package), and for uncertainties and distributions (disturb package). Expanding the functionality of SBML4Humans to include these packages greatly increases the utility of the package for modelers in these specialized domains.

Altogether, SBML4Humans provides modelers at all stages an accessible method to investigate SBML models without requiring significant knowledge of the standard format or significant effort to parse the model manually. It can be used to readily review the model components and to understand the biology represented. With the ability to filter and search for specific terms and elements, and to investigate not only the explicit model components and parameters but also the metadata and annotations describing them, SBML4Humans allows users to gain great insight into how each model was constructed. Ultimately, this insight is essential to ensure that models can be reproduced and reused to further the scientific exploration.

Reference: S. Das and M. König. SBML4Humans: Interactive SBML report for humans. preprints.org. January, 2023.