UC Irvine
UC Irvine Previously Published Works

Title
Pycellerator: an arrow-based reaction-like modelling language for biological simulations.

Permalink
https://escholarship.org/uc/item/5ng839zj

Journal
Bioinformatics (Oxford, England), 32(4)

ISSN
1367-4803

Authors
Shapiro, Bruce E
Mjolsness, Eric

Publication Date
2016-02-01

DOI
10.1093/bioinformatics/btv596

Peer reviewed

eScholarship.org Powered by the California Digital Library
University of California

Bioinformatics, 32(4), 2016, 629–631
doi: 10.1093/bioinformatics/btv596
Advance Access Publication Date: 26 October 2015
Applications Note

OXFORD

Systems biology

Pycellerator: an arrow-based reaction-like modelling language for biological simulations

Bruce E. Shapiro[1,*] and Eric Mjolsness[2]

[1]Department of Mathematics, California State University, Northridge, CA 91330, USA and [2]Department of Computer Science, University of California, Irvine, CA 92697, USA

*To whom correspondence should be addressed.

Associate Editor: Jonathan Wren

Received on July 12, 2015; revised on September 19, 2015; accepted on October 12, 2015

Abstract

Motivation: We introduce Pycellerator, a Python library for reading Cellerator arrow notation from standard text files, conversion to differential equations, generating stand-alone Python solvers, and optionally running and plotting the solutions. All of the original Cellerator arrows, which represent reactions ranging from mass action, Michales–Menten–Henri (MMH) and Gene-Regulation (GRN) to Monod–Wyman–Changeaux (MWC), user defined reactions and enzymatic expansions (KMech), were previously represented with the Mathematica extended character set. These are now typed as reaction-like commands in ASCII text files that are read by Pycellerator, which includes a Python command line interface (CLI), a Python application programming interface (API) and an iPython notebook interface.

Results: Cellerator reaction arrows are now input in text files. The arrows are parsed by Pycellerator and translated into differential equations in Python, and Python code is automatically generated to solve the system. Time courses are produced by executing the auto-generated Python code. Users have full freedom to modify the solver and utilize the complete set of standard Python tools. The new libraries are completely independent of the old Cellerator software and do not require Mathematica.

Availability and implementation: All software is available (GPL) from the github repository at https://github.com/biomathman/pycellerator/releases. Details, including installation instructions and a glossary of acronyms and terms, are given in the Supplementary information.

Contact: bruce.e.shapiro@csun.edu

Supplementary information: Supplementary data are available at *Bioinformatics* online.

1 Introduction

Cellerator describes biological interactions with a reaction-like arrow-based input language. Input strings are converted into differential equations and integrated to produce numerical time-course predictions using Mathematica (Shapiro *et al.*, 2003). Extensions include Cellzilla for two-dimensional tissue simulation (Shapiro *et al.*, 2013) and KMech for exact enzymatic expansion (Yang *et al.*, 2005).

There are many tools that convert reactions to differential equations and solve them, particularly in Python, but they do not use an arrow-based language in the same manner as Cellerator. Examples include PySCeS (Olivier *et al.*, 2004), which has its own text modeling language; PyDSTool (Clewey, 2012) (for hybrid systems); and pybrn (`pybrn.sf.net`, for SBML-like structures). Perhaps the closest conceptually to Pycellerator are PySB (Lopez *et al*, 2013) and SBML shorthand (Wilkinson, 2011). PySB is a rule-based system with a collection of text language rules that are merged into Python commands; models are built as Python programs. SBML Shorthand is not a simulator; it is a pre-processor for converting models into SBML, in which each reaction is represented by a single line of text. In addition, a number of popular stochastic simulation tools are also implemented in Python.

© The Author 2015. Published by Oxford University Press. All rights reserved. For Permissions, please e-mail: journals.permissions@oup.com

Pycellerator also provides a simulation, modelling, programming and analysis interface. In Pycellerator the modelling language corresponds to (and extends) the arrow language introduced in Cellerator. Each arrow corresponds to a Cellerator arrow, but it can now by reprsented with ASCII arrow-like characters. The Pycellerator library provides a command line interface (CLI), an application programming interface (API) and an iPython notebook interface (Perez and Granger, 2007), so that users have a choice of Python programming environments. Mathematica is not required.

2 Approach

Simulations require a model file (Supplementary Fig. S1), which is a text file divided up into sections representing reactions, rate constants, initial conditions and functions. Alternatively, models may be generated from (or saved as) SBML (Hucka *et al.*, 2003) or as legacy Mathematica files. The model is parsed, converted to differential equations and Python code is generated to perform the simulation. This auto-generated code may either be run and plotted within the notebook (Fig. 1) or from the command line, and is a completely stand-alone program (Supplementary Fig. S4).

Within a model file arrows are enclosed in square brackets; the canonical Cellerator form becomes `[X- >Y,k]` where the typewriter symbols – > replace the fancier →. When several species are involved, expressions like $\mathtt{e_1}\ \mathtt{X_1} + \mathtt{e_2}\ \mathtt{X2} + \cdots$ and $\mathtt{f_1}\ \mathtt{Y_1} + \mathtt{f_2}\ \mathtt{Y2} + \cdots$ can be used on the left and right hand side of the arrow, where $\mathtt{e_i}$ and $\mathtt{f_j}$ are the before and after stoichiometries, The interpreter converts each reactant $\mathtt{U_j}$ (with its stoichiometry) to a differential equation term using mass action kinetics, e.g. $U_j' = \sum_{\text{reactions}} [(f_j - e_j) k X_1^{e_1} X_2^{e_2} \cdots]$. Additional text forms exist for all Cellerator arrows, e.g. catalyzed reactions, MMH, Hill, S-systems, GRN, user-defined arrows and all KMech reactions (see Supplementary Tables S1–S3).

Built-in functions allow the user to inspect the differential equations, generate simulation code and run a simulation or parameter scan. In addition, standard Python packages (e.g. `pyplot`, `numpy`, `scipy`, `sympy`) can be used to analyze the results of the simulation.

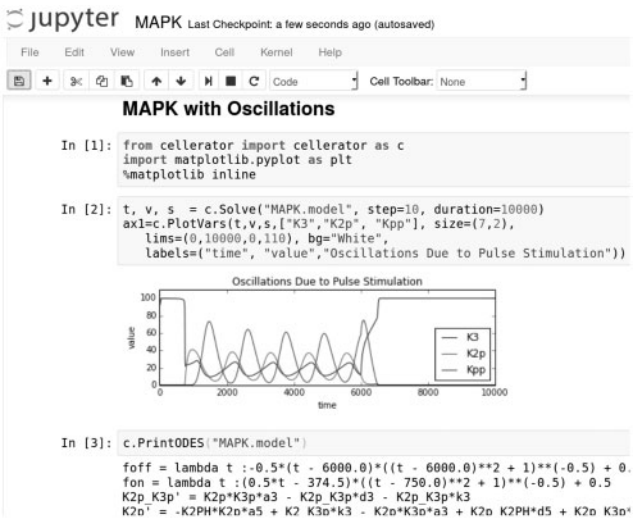

Fig. 1. Sample output (cutoff) for simulating oscillations in a MAP kinase cascade. This model is structurally identical to (Huang and Ferrell, 1996) with two modifications: time-dependent input and a competitive feedback reaction. A similar model was proposed by (Kholodenko, 2000), and oscillations have been observed in yeast by (Hilioti *et al.*, 2008). The model file and output code are shown in Supplementary Figures S1 and S4

3 Methods

The work flow is summarized in Supplementary Figure S2. Input files are parsed using the `pyparsing` package, which allows the grammar to be specified in BNF. Reactions are converted into a Python `reaction` class that is used for all subsequent data processing. Conversion into differential equations utilizes the `sympy` symbolic processing package. Flux models use `pulp`, the Python linear programming toolkit. Simulation code is generated and saved as a Python program using the `odeint` solver in `scipy`. This solver is a wrapper for LSODA (Hindmarsh, 1983) and automatically switches between stiff (BDF) and non-stiff (Adams) methods, depending on the nature of the problem. The user has full access to the code and can change any desired parameter of the solver. This stand-alone code can be run as a separate program from the command line, wrapped within another program, or run automatically using `eval`. Default is to output a `numpy` array.

4 Discussion

Pycellerator variables may be specified using an array index notation to represent multiple compartments or multi-stage cascades. A possible future extension to the software would be two and three dimensional tissue based implementations similar to Cellzilla (Shapiro *et al.*, 2013). Dynamical grammars can also be implemented using a more expressive notation that makes this modelling paradigm significantly more powerful (Mjolsness, 2013). Rule-based models, graph grammars and hybrid systems, for example, could be automaticallygenerated using the techniques we have described here.

5 Conclusion

The Cellerator arrow notation for specifying biological interactions has been implemented in a human-readable text-based language. Parsers could be written in any language. Extensible open source Python libraries are provided for CLI, API and notebook support. The iPython notebook provides Pycellerator with a very convenient front end for modelers who want to combine code, text, markup and figures together in single documents. This is particularly useful for training researchers. The libraries can be used either as a front-end to produce or interact with other solvers or as end-user solvers in of themselves.

Funding

E.M. was supported by NIH grants R01 GM086883 and R01 HD073179, and by the United States Air Force under Contract No. FA8750-14-C-0011 under the DARPA PPAML program.

Conflict of Interest: none declared.

References

Clewley,R. (2012) Hybrid Models and Biological Model Reduction with PyDSTool. *PLoS Comp. Biol.*, **8**, e1002628.

Hilioti,Z. *et al.* (2008) Oscillatory phosphorylation of yeast Fus3 MAP kinase controls periodic gene expression and morphogenesis. *Curr. Biol.*, **18**, 1700–1706.

Hindmarsh,A.C. (1983) ODEPACK: a systematized Collection of ODE Solvers. *IMACS Trans. Sci. Comp.*, **1**, 55–64.

Huang,C.F. and Ferrel, J.E. (1996) Ultrasensitivity in the mitogen-activated protein kinase cascade. *Proc. Natl Acad. Sci. USA*, **93**, 10078–10083.

Hucka,M. *et al*. (2003) The systems biology markup language (SBML): a medium for representation and exchange of biochemical network models. *Bioinformatics*, **19**, 513–523.

Kholodenko,B.N. (2000) Negative feedback and ultrasensitivity can bring about oscillations in the mitogen-activated protein kinase cascades. *Eur. J. Biochem*., **267**, 1583–1588.

Lopez,C.F. *et al*. (2013) Programming biological models in Python using PySB. *Mol. Syst. Biol*., **9**. doi:10.1038/msb.2013.1

Mjolsness,E. (2013) Time-ordered product expansions for computational stochastic systems biology. *Phys. Biol*., **10**, 035009.

Olivier,B.G. *et al*. (2004) Modelling cellular systems with PySCeS. *Bioinformatics*, **21**, d560–d561.

Pérez,F. and Granger,B. (2007) IPython: a system for interactive scientific computing. *Comput. Sci. Eng*., **9**, 21–29. doi:10.1109/MCSE.2007.53.

Shapiro,B.E. *et al*. (2003) Cellerator: extending a computer algebra system to include biochemical arrows for signal transduction simulations. *Bioinformatics*, **19**, 677–678. doi: 10.1093/bioinformatics/btg042.

Shapiro,B.E. *et al*. (2013) Using cellzilla for plant growth simulations at the cellular level. *Front. Plant Sci*., **4**. doi: 10.3389/fpls.2013.00408

Wilkinson,D. (2011) *Stochastic Modeling for Systems Biology*. CRC Press, Boca Raton.

Yang,C.-R. *et al*. (2005) An enzyme mechanism language for the mathematical modeling of metabolic pathways. *Bioinformatics*, **21**, 774–780.