BIOINFORMATICS

Vol. 00 no. 00 2012
Pages 1–2

Simulation Core Library: a Java library for numerical computation in systems biology

Andreas Dräger [1,*], Roland Keller [1,†], Alexander Dörr [1,†], Akito Tabira [2], Akira Funahashi [2], Michael J. Ziller [3], Richard Adams [4], Nicolas Rodriguez [5], Nicolas Le Novère [5], Hannes Planatscher [6], Andreas Zell [1,*]

[1]Center for Bioinformatics Tuebingen (ZBIT), University of Tuebingen, Tübingen, Germany, [2]Keio University, Graduate School of Science and Technology, Yokohama, Japan, [3]Department of Stem Cell and Regenerative Biology, Harvard University, Cambridge, MA, USA, [4]SynthSys Edinburgh, CH Waddington Building, University of Edinburgh, Edinburgh EH9 3JD, UK, [5]European Bioinformatics Institute, Wellcome Trust Genome Campus, Hinxton, Cambridge, UK, [6]Natural and Medical Sciences Institute at the University of Tuebingen, Reutlingen, Germany

Received on XXXXX; revised on XXXXX; accepted on XXXXX

Associate Editor: XXXXXXX

ABSTRACT

Motivation: Dynamic simulation of biological phenomena is a the key aspect of research in systems biology. However, it is often difficult to use available implementations of numerical methods as a backend for custom-made programs.

Results: The Simulation Core Library is a community-driven project that provides a large collection of numerical solvers and a sophisticated interface hierarchy for the definition of custom differential equation systems. It is entirely implemented in Java™ without the necessity to include any platform-dependent wrappers or libraries, and does not depend on any commercial licenses. It already includes an efficient and exhaustive implementation of methods to interpret the content of models encoded in SBML using the JSBML project. To demonstrate its capabilities, it has been tested with the entire SBML Test Suite and all models of BioModels Database.

Availability: Source code, binaries, and documentation can be freely obtained under the terms of the LGPL version 3 from the website http://sourceforge.net/projects/simulation-core/.

Contact: simulation-core-development@lists.sourceforge.net

1 INTRODUCTION

As part of the movement towards quantitative biology, modeling, simulation, and computer analysis of biological networks have become integral parts of modern biological research. XML-based standard description formats such as the Systems Biology Markup Language (SBML, Hucka *et al.* 2004) or CellML (Lloyd *et al.*, 2004) enable encoding of biological network models, and interpret them in terms of a differential equation system, with additional structures such as discrete events and algebraic equations. Software libraries for reading and manipulating the content of these formats are available. However, a prerequisite for model analysis, simulation, and calibration (e.g., the estimation of parameter values), is a multiple-purpose and efficient numerical solver library that has been designed with the requirements of biological network models in mind.

Many stand-alone programs providing these features come with graphical user interfaces. However, the vast majority of the internal solvers for these systems are part of larger software suites and can therefore not be easily integrated into custom programs. Some are implemented in programming languages that are either platform-dependent (e.g., C or C++) and/or require a commercial license (e.g., MATLAB™) for their execution.

The Simulation Core Library presented here is a platform-independent, well-tested alternative. This generic library is completely decoupled from any graphical user interface and can therefore easily be integrated into third-party programs. It comprises several Ordinary Differential Equation (ODE) solvers and an interpreter for SBML models. It is the first simulation library based on JSBML (Dräger *et al.*, 2011). Furthermore, the Simulation Core Library contains classes to both export simulation configurations to SED-ML (Simulation Experiment Description Markup Language, Waltemath *et al.*, 2011), and facilitate the re-use and reproduction of these experiments by executing SED-ML files.

2 IMPLEMENTATION

All the solver classes are derived from the abstract class `AbstractDESSolver` (Fig. 1). Several solvers of the Apache Commons Math library (version 3.0) are integrated with the help of wrapper classes. Numerical methods and the actual differential equation systems are strictly separated. The class `MultiTable` stores the results of a simulation within its `Block` data structures. The abstract description of differential equation systems, with the help of several distinct interfaces, makes possible to decouple them from a particular type of biological network. It is therefore possible to pass an instance of an interpreter for a respective model description format to any available solver. This interpretation is the most time consuming step of the integration procedure. This is why efficient and clearly organized data structures are required to ensure a high performance of the overall library. The interpretation of SBML models is split between evaluation of

*to whom correspondence should be addressed

†authors with equal contribution

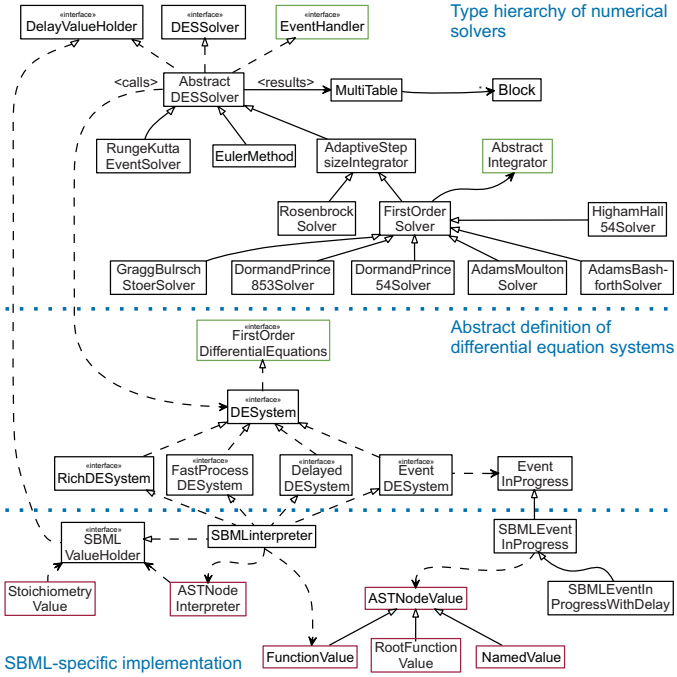

Figure 1: Architecture of the Simulation Core Library (simplified). Numerical methods are strictly separated from differential equation systems. The upper part displays the unified type hierarchy of all currently included numerical integration methods. The middle part shows the interfaces defining several special types of the differential equations to be solved by the numerical methods. The class `SBMLinterpreter` (bottom part) implements all of these interfaces with respect to the information content of a given SBML model. Similarly, an implementation of further data formats can be included into the library.

events and rules, computation of stoichiometric information, and computation of the current values for all model components (such as species and compartments). For a given state of the ODE system, the class `SBMLinterpreter`, responsible for the evaluation of models encoded in SBML returns the current set of time-derivatives of the variables. It is connected to an efficient MathML interpreter of the expressions contained in kinetic laws, rules and events. The nodes of the syntax tree for those expressions depend on the current state of the ODE system. If the state has changed, the values of the nodes have to be recalculated. At the beginning of the simulation the syntax trees of all kinetic laws, rules and events are restructured and merged to contain equivalent nodes only once. This significantly decreases the computation time during the simulation. An important aspect in the interpretation of SBML models is the determination of the exact time at which an event occurs, as this influences the precision of the system's variables. We therefore adjusted the Rosenbrock solver (Kotcon *et al.*, 2011), an integrator with an adaptive step size, to a very precise timing of the events. Algebraic rules are turned into assignment rules before the simulation: If the system is not overdetermined, exactly one of the variables contained in an algebraic rule can be chosen as the target variable of the new assignment rule with the help of a bipartite matching. The simulation algorithm then proceeds as follows: For each time step, the ODE solver gets the current variable values and calculates the system's state for the next time point. After that, events and rules are processed, that can change the values. The modified values then become the initial values for the next time step. The event processing of the Rosenbrock solver is directly integrated in the solver class and influences the step size. The time-accurate handling of events and rules leads to very precise results of the simulation. SED-ML support is enabled by inclusion of the `jlibsedml` library (http://www.jlibsedml.org) in the binary download. Clients of the the Simulation Core Library can choose to use the `jlibsedml` API directly, or access SED-ML support via facade classes in the `org.simulator.sedml` package that do not require direct dependencies on `jlibsedml` in their code.

3 RESULTS AND CONCLUSION

The SBML implementation has successfully passed the SBML Test Suite (version 2.0.2). Furthermore, it solved 99.27 % of the models from the BioModels.net database (release 21, Le Novère *et al.*, 2006). Therefore, the Simulation Core Library is an efficient Java tool for the simulation of differential equation systems used in systems biology. It can be easily integrated in larger applications. For instance, CellDesigner version 4.2 (Funahashi *et al.*, 2003) already uses it as one of its simulation libraries. The stand-alone application SBMLsimulator (available at http://www.cogsys.cs.uni-tuebingen.de/software/SBMLsimulator) provides a convenient graphical user interface for the simulation of SBML models and uses it as a computational backend. The abstract class structure of the library supports the integration of additional model formats, such as CellML, besides its SBML implementation. To this end, it is only necessary to implement a suitable interpreter class.

By including support for the emerging standard SED-ML, we hope to facilitate the exchange, archival and reproduction of simulation experiments performed using the Simulation Core Library.

ACKNOWLEDGEMENT

The authors are grateful to B. Kotcon, S. Mesuro, D. Rozenfeld, A. Yodpinyanee, A. Perez, E. Doi, R. Mehlinger, S. Ehrlich, M. Hunt, G. Tucker, P. Scherpelz, A. Becker, E. Harley, and C. Moore, Harvey Mudd College, USA, for providing a Java implementation of Rosenbrock's method, and to Michael T. Cooling, University of Auckland, New Zealand, for fruitful discussion.

Funding: The Federal Ministry of Education and Research (BMBF, Germany) in the project Virtual Liver (grant number 0315756).

Conflict of Interest: none declared.

REFERENCES

Dräger, A. *et al.* (2011). JSBML: a flexible Java library for working with SBML. *Bioinformatics*, **27**(15), 2167–2168.

Funahashi, A. *et al.* (2003). CellDesigner: a process diagram editor for gene-regulatory and biochemical networks. *BioSilico*, **1**(5), 159–162.

Hucka, M., *et al.* (2004). Evolving a lingua franca and associated software infrastructure for computational systems biology: the Systems Biology Markup Language (SBML) project. *Systems Biology, IEE*, **1**(1), 41–53.

Kotcon, B. *et al.* (2011). *Final Report for Community of Ordinary Differential Equations Educators*. Harvey Mudd College Joint Computer Science and Mathematics Clinic, 301 Platt Boulevard, Claremont, CA 91711.

Le Novère, N. *et al.* (2006). BioModels Database: a free, centralized database of curated, published, quantitative kinetic models of biochemical and cellular systems. *Nucleic Acids Res*, **34**, D689–D691.

Lloyd, C. M. *et al.* (2004). CellML: its future, present and past. *Prog Biophys Mol Biol*, **85**(2-3), 433–450.

Waltemath, D. *et al.* (2011). Reproducible computational biology experiments with SED-ML–the Simulation Experiment Description Markup Language. *BMC Syst Biol*, **5**, 198.