*Proceedings of the 2009 Winter Simulation Conference*
*M. D. Rossetti, R. R. Hill, B. Johansson, A. Dunkin, and R. G. Ingalls, eds.*

# DESIGN AND DEVELOPMENT OF SOFTWARE TOOLS FOR BIO-PEPA

Adam Duguid
Stephen Gilmore
Maria Luisa Guerriero

School of Informatics
The University of Edinburgh
Edinburgh, EH8 9AB, U.K.

Jane Hillston
Laurence Loewe

Centre for Systems Biology at Edinburgh
The University of Edinburgh
Edinburgh, EH9 3JH, U.K.

## ABSTRACT

This paper surveys the design of software tools for the Bio-PEPA process algebra. Bio-PEPA is a high-level language for modelling biological systems such as metabolic pathways and other biochemical reaction networks. Through providing tools for this modelling language we hope to allow easier use of a range of simulators and model-checkers thereby freeing the modeller from the responsibility of developing a custom simulator for the problem of interest. Further, by providing mappings to a range of different analysis tools the Bio-PEPA language allows modellers to compare analysis results which have been computed using independent numerical analysers, which enhances the reliability and robustness of the results computed.

## 1 INTRODUCTION

Systems biology has recently inspired much work on simulation tools and techniques (Kitano 2002). It was born with the realisation that computational models are pivotal for understanding the increasingly complex molecular biological systems under investigation (Regev and Shapiro 2002). Such models can make quantitative predictions, which improve our capacity to construct falsifiable hypotheses that are at the heart of scientific progress. The need for quantitative models in biology is enormous and will further increase as the quantitative approach of systems biology slowly works its way into various other biological disciplines. For example, evolutionary systems biology depends on the ability to automatically analyse variations in many realistic models for answering some of the most difficult questions in evolutionary biology (Loewe 2009). To meet this need for high quality simulations, it is desirable to have an easy way of building concise models of biochemical reaction networks. It is also desirable that these models are amenable to as many automated model analysis techniques as possible. This contributes to the quality of results and to the level of understanding without requiring potentially large amounts of time for tedious and error-prone low-level programming tasks.

The goal of Bio-PEPA tools is to deliver a modelling environment that is easy to use and supports a wide range of analyses for biochemical systems. Such systems consist of biochemical species which are affected by reactions that change the numbers of these species. Bio-PEPA (Ciocchetta and Hillston 2008, Ciocchetta and Hillston 2009) is a stochastic process algebra designed specifically for investigating these systems. It represents species of molecules as "sequential components" and reactions as "actions" (for simplicity, we will continue to call them "species" and "reactions" here). Bio-PEPA supports various qualitative analysis methods in addition to quantitative methods such as discrete stochastic simulation, continuous deterministic simulation, and probabilistic model-checking. The Bio-PEPA language promotes *integrated* model analysis where a high-level language (in our case a process algebra) is mapped to a set of different model representations so that a range of different formal analyses can be applied. This has been demonstrated in a number of examples (Calder et al. 2006, Ciocchetta et al. 2009).

The Bio-PEPA tools are under constant development to support language extensions and new analysis techniques. For instance, following the recent formal definitions of locations (Ciocchetta and Guerriero 2009) and events (Ciocchetta 2009), their support within the Bio-PEPA tools is currently under development. Furthermore, novel analysis techniques and abstractions for Bio-PEPA (for instance the application of behavioural equivalences (Galpin and Hillston 2009)), can be easily integrated into the Bio-PEPA tools.

This work describes how we organise the development of model analysis tools for Bio-PEPA and the software architecture of some of these tools. As will be seen below, we organise our tools in very minimalistic "tool chains", which currently pass through one of two main tools for processing Bio-PEPA files. The resulting very loose coupling allows for flexibility in choosing how to implement additional functionalities. This structure results in minimal dependencies, as it only requires the unambiguous definition of interface file-formats between modules. Thus a high degree of parallelism is enabled in tool development and code reuse is simple in the form of reusing the functionality of an existing tool in a new context.

The remainder of this paper is organised as follows. Section 2 provides some background information on Bio-PEPA and its quantitative and qualitative mathematical analysis techniques before comparing it to related tools. Section 3 outlines the design goals for Bio-PEPA tool development, before discussing the benefits of our minimalistic tool chains approach. Section 4 provides an overview of the current structure of tools in the Bio-PEPA world. Finally, Section 5 demonstrates the use of our tools on various case studies.

## 2 BACKGROUND

### 2.1 Properties of Bio-PEPA

There are many high-level modelling languages in existence that can be used to formally describe a network of biochemical reactions. Such formalisms like process algebras, Petri nets and notations for model-exchange (e.g. SMBL) vary in properties such as the degree to which they are domain specific. Here we briefly review key aspects of the design of the process algebra Bio-PEPA (Ciocchetta and Hillston 2008, Ciocchetta and Hillston 2009). Bio-PEPA expands the stochastic process algebra PEPA (Hillston 1996) to include functional rates, stoichiometry and other features that facilitate the modelling of biochemical reaction networks. It consists of a small number of operators with clearly defined rules that dictate possible behaviours. Each reaction a species participates in must state the behaviour/role of the species in this reaction (e.g. reactant, product, modifier etc) while other operators and rules specify how these species should interact. Here we briefly review key properties of the process algebra Bio-PEPA.

**Clarity:** Clarity can be pivotal for constructing models. For example, in the Ordinary Differential Equation (ODE) model of the MAP kinase cascade (Schoeberl et al. 2002) inspection of the model, partially in the form of a reaction list, is insufficient for understanding the model. Only a line by line examination of the Matlab code will reveal modifications to the ODEs generated from the reaction data thus discovering the dependency between the data and the code. In contrast, the equivalent Bio-PEPA model requires every species to clearly identify its role in all the reactions it participates in. In this example, results can only be replicated, once it is clear that the EGF species acts as a catalyst in reaction $v1$. Without this insight the dynamics of the system cannot be properly understood.

**Static analysis:** Automated static analyses of formal model descriptions can be powerful tools for detecting modelling errors. Errors are inevitable and can be difficult to find, especially if they do not lead to results that are obviously wrong. Bio-PEPA can assist modellers by facilitating the implementation of automated static analyses for the detection of inconsistencies. Corresponding errors are not automatically detected if simulators or ODE systems are written by hand. For example, we know from biochemistry that a reaction with Michaelis-Menten kinetics requires the presence of three species. Thus, by defining the kinetic law for Michaelis-Menten as $\mathrm{fMM}(V_m, K_m)$ we can verify that three species do participate in this reaction, each performing one of the three required roles for Michaelis-Menten to be applicable (enzyme, substrate and product). When support for Bio-PEPA with locations (Ciocchetta and Guerriero 2009) is completed, static analysis will be augmented to ensure reactions only involve species in adjacent locations.

**Multiple analysis vectors:** A significant advantage offered by some high-level languages is the ability to map to a wide range of solvers. The definition of PEPA (Hillston 1996) detailed a mapping to Continuous Time Markov Chains (CTMCs), which was later expanded to include both Stochastic Simulation Algorithms (SSAs) and ODEs. Gillespie's SSA (Gillespie 1977) is built from first principles and shown to be exact in relation to the Chemical Master Equation which is a CTMC itself. Through the thermodynamic limit it is known that SSAs converge with ODEs (Gillespie 2008). Using these proofs, and being aware of any conditions as to when they apply, we can start from a high-level language and produce mappings to each family of solvers. This can even be taken one step further, by mapping to multiple implementations. The extent that the Bio-PEPA tools can do this is shown in Figure 1.

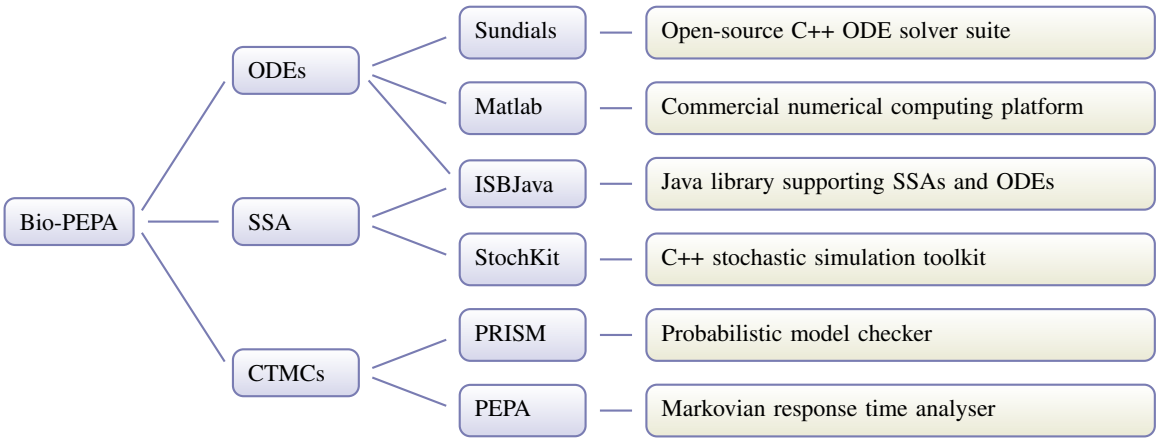


Figure 1: Bio-PEPA supports different types of analysis which are implemented by a range of software tools and libraries.

### 2.2 Mathematical and Computational Analyses

The world of Bio-PEPA is difficult to understand without a basic grasp of the importance of the various qualitative and quantitative analysis techniques that Bio-PEPA is designed to provide access to. For practical computational purposes it makes a big difference whether analysis techniques are "state-space based" or not. The latter can routinely handle models with large numbers of reactions and components. State-space based methods, instead, require the explicit enumeration of every single state that the whole model could be in. This leads to a combinatorial explosion, which severely limits the size of models that can by analysed by such approaches, despite various techniques that have been developed to reduce this problem (see, for instance, (Heath et al. 2008, Ciocchetta et al. 2009, Guerriero 2009)). The following analysis techniques are currently supported by Bio-PEPA.

**CTMCs: Continuous Time Markov Chains.** A CTMC is most often described by a set of states $S$ and the transition matrix $Q$: $S \times S$. Each state represents a *global* state of the system and not the local state of a species (in the case of Bio-PEPA). For example, let a system consist of two species $A$ and $B$, where each can have a molecular count between 0 and 10. If a state represents a particular molecular count, then each species has 10 local states, while the number of global states depends on the system. If $A$ and $B$ are independent, the system will have 100 states. Adding 10 molecules of a third species increases this to 1000 states. Thus it is easy to see how the size of a CTMC rapidly increases with model size. If the CTMC can be solved numerically, it can be used to compute the transient and steady-state distributions of species, as well as the expected number of occurrences for transitions and their durations. CTMC-based analysis can be performed using a probabilistic model-checker such as PRISM (Kwiatkowska, Norman, and Parker 2002) which allows the formal verification of model properties by performing an exhaustive exploration of the state-space. The formal verification of properties can also be used to detect modelling errors by checking if the observed behaviour of the system conforms to its desired behaviour. Detailed examples of system properties which can be formally verified through model-checking are described elsewhere (Guerriero 2009). In short:

1. absence of *deadlocks/livelocks*, which are states or groups of states that are never left once they are reached;
2. *reaction liveness*, indicating whether a reaction can occur in given states;
3. presence of *species invariants*, which are species or sets of species with constant amounts;
4. *state reachability*, indicating whether a state can be reached in principle;
5. *reversibility*, indicating that a given state can be reached repeatedly;
6. *reaction ordering/causality*, which identifies events that must occur before a given event ("sequences") or that will necessarily follow it ("consequences").

The strength of CTMC analysis lies in the wealth of information that can be calculated and the fact that the analysis can return the correct probability for very rare events too. The weakness of these CTMC analyses is the problem

of state-space explosion. Even small numbers of each species result in intractable CTMCs. To reduce this problem, Bio-PEPA supports the derivation of a *CTMC with levels* (Calder, Gilmore, and Hillston 2004), which discretises the molecular counts into levels. Each level represents a range of values rather than a single copy of a species. The resulting CTMCs are reduced in size and in doing so can become tractable with current CTMC solvers. The resolution of this approach can be controlled by adjusting the size of the level until the desired trade-off is reached between efficiency and correctness.

**SSA: Stochastic Simulation Algorithm.** An alternative to constructing the entire state-space is to simulate individual traces of stochastic walks through the state-space. This is possible by using Gillespie's Stochastic Simulation Algorithm (SSA, see (Gillespie 1977)) or one of its more efficient newer equivalents (Gibson and Bruck 2000). This procedure is exact for numerically simulating the time evolution of a well-stirred chemically reacting system by taking proper account of the randomness inherent in such a system. The Bio-PEPA Workbench uses the StochKit (Li et al. 2008) implementation of Gillespie's Direct Method to perform stochastic simulations of systems while the Bio-PEPA Eclipse Plug-in uses ISBJava, the back-end library for Dizzy (Ramsey, Orrell, and Bolouri 2005).
The cost of an exact simulation can be expensive; the algorithm needs to guarantee that only one reaction can take place in a given time interval and therefore the time step for the next reaction can be very small. When compared to ODEs the analysis of chemical reactions using SSAs is computationally less efficient, as ensembles of independent runs are needed to compute average trajectories and variance. Also, if the observation of rare events is important, the increased number of independent runs could become too computationally expensive. Despite this increased analysis cost the SSA is often preferred over ODEs because it is an exact analysis, not an approximate one. In addition SSAs, like CTMCs, can analyse systems with non-differentiable functions.

**ODE: Ordinary Differential Equations.** In some systems only the mean behaviour is of interest, especially if very large quantities of molecules react in the same way. If information on variability is unnecessary, then ODEs can approximate such systems much faster than simulations. The precision of ODEs breaks down as soon as molecule numbers approach zero (ODE solvers would continue with fractions of molecules, even if the real system will have either 1 or 0 molecules, as reflected by the SSA). To support ODEs, Bio-PEPA tools need to generate the reaction rate equations in the form of a system of coupled ordinary differential equations where the system variables of the differential equations allow us to determine the quantity of each chemical species in the reaction at any point up to a finite time horizon.

As can be seen from the list above, each type of analyses has advantages and limitations. Thus modellers might employ more than one type of analysis to increase their understanding of the system.

### 2.3 Comparing Bio-PEPA to Alternative Biochemical Languages

The desired properties of Bio-PEPA determine important features that distinguish Bio-PEPA from the many other languages that are also designed for modelling biochemical species. The stochastic simulator Dizzy (Ramsey, Orrell, and Bolouri 2005) implements the reaction-style Chemical Model Definition Language (CMDL) and supports mapping to SSAs and ODEs, but not to state-space based CTMCs. Rule-based systems like the $\kappa$-calculus (Danos et al. 2007) or BioNetGen (Faeder, Blinov, and Hlavacek 2009) are powerful tools for simulating stochastic systems where the number of *potential* types of complexes with different internal states might exceed the number of *actual* molecules in the system. However, they also do not support direct state-space based CTMC analyses and for ODE analyses they have to reduce rule-based models to the level of Bio-PEPA models, where all different types of molecules are explicitly known at the beginning of the simulation. The Stochastic Pi Machine (Phillips 2009), BIOCHAM (Fages and Soliman 2008) and the Petri-net based tool Snoopy (Heiner et al. 2008) are other related simulation environments for biochemical systems that differ in their syntax, semantics and specific model analysis capabilities. SPiM is a process algebra that supports the dynamic opening of new "communication channels" between molecules (the basic property that makes rule-based modelling possible and direct ODE analysis impossible), BIOCHAM is another rule-based reaction-style modelling language and Snoopy relies on the graphical representation of Petri-nets to describe models (as opposed to the textual representations of process algebras).

A hallmark of Bio-PEPA code is its concise clutter-free style that provides a very quick overview of all reactions that a particular molecular species is involved in (as opposed to reaction-style representations that provide a quick overview of all molecular species that are involved in a particular reaction; both views are useful). Bio-PEPA was recently compared with other process algebras (Calder and Hillston 2009). Work on Bio-PEPA is motivated by the belief that high-level languages such as process algebras, Petri nets, or notations like SBML have a useful role to play in systems biology research. An advantage of a domain-specific high-level language is that models can be expressed much more concisely than in a general

purpose programming language such as C++ or Java. It has the added benefit that the model can be compiled to a number of other formats for different types of analysis without requiring further error-prone work from the modeller or the need to learn how to use new tools.

## 3 DESIGN GOALS AND ORGANISATION OF DEVELOPMENT

### 3.1 Requirements of Modelling Tools for Bio-PEPA

Bio-PEPA software tools should meet the following requirements to maximise their usefulness:

1. Facilitate running several different types of quantitative analysis on a single Bio-PEPA model.
2. Facilitate combining the results of several runs of one particular type of analysis of a Bio-PEPA model.
3. Require as little explicit programming as possible from users.
4. Allow users to choose the parameters of interest for closer investigation.
5. Build on other mature simulators and numerical libraries where possible to avoid re-implementing existing functionality.
6. Users should be involved in deciding which features are important through suggesting enhancements to current versions.

### 3.2 How Development of Bio-PEPA Tools is Organised

A by-product of developing mappings to a variety of solvers is the ability to make use of pre-existing tools and libraries. As can be seen from Figure 1, Bio-PEPA links to a number of powerful and well-tested tools. In reusing these tools, we avoided addressing difficult numerical problems such as the control of errors which must be managed in the implementation of a numerical integrator for ODEs (Shampine and Reichelt 1997).

There are many other advantages for tool re-use. They facilitate the rapid development of new modelling environments. The incorporation of new tools allows comparing results from different implementations to increase the confidence that they work correctly. Such an "end-to-end" testing of model analysis code at the highest possible level is a powerful means for detecting errors in the mapping or in the solver. Such errors become more apparent when the results of many solvers are compared. More tools will increase the probability that modellers can use their favoured tools or make them confident about using a free tool that has been shown by Bio-PEPA developers to work as a replacement for a commercial product such as Matlab. Thus the availability of multiple solvers increases the overall reliability of simulation results.

For organising our development work we use the concept of loosely coupled "tool chains" that is somewhat related to classical pipelining. Here each element accepts models in one particular form as input and then produces its output, which is either the same model in another representation or numerical results that are generated from that model. Figure 2 provides an overview of the various options for selecting tool chains from the set of Bio-PEPA tools. We describe the tool chains as loosely coupled, as their elements are mostly independent applications that accept input in the form of ASCII files and produce output in the form of ASCII files; at the same time the order of input and output formats enforces a coupling that only allows to go from applications on the left to those on the right in Figure 2. Thus, our system supports more than one entry point for users that want to build Bio-PEPA models (e.g. by using a graphical editor). It also supports arbitrary numbers of solvers. This is a key feature of how our system is organised and facilitates the use of a wide range of independent third party tools, each supporting a specific mode of analysis. Thus we can tap into the development efforts of a world-wide community of tool developers by simply constructing a translator that links a particular tool to Bio-PEPA. Translators can take two forms:

- *Low-level internal translators.* Here the Bio-PEPA model has already been parsed and now exists as a data structure or object in memory. To make use of a particular linked library that implements a particular analysis method the details of the parsed model are translated into the type required by this library. This is done by the code that glues together the library and the internal representation of the parsed Bio-PEPA model. Everything happens within one executable and is hidden from the user. Example: use of the ISBJava library within the Bio-PEPA Eclipse Plug-in.
- *External translators.* Here the Bio-PEPA file is translated into another format that can be processed by a specific tool. The syntax is determined by the downstream tool and can build on a wide variety of formats, including the production of C++ code that is ready for compiling against a particular library like StochKit. External translators allow users to easily get access to the produced code even if they are not meant to have such access. The file formats of intermediate states can range from code to mature external standards like SBML. Development of the

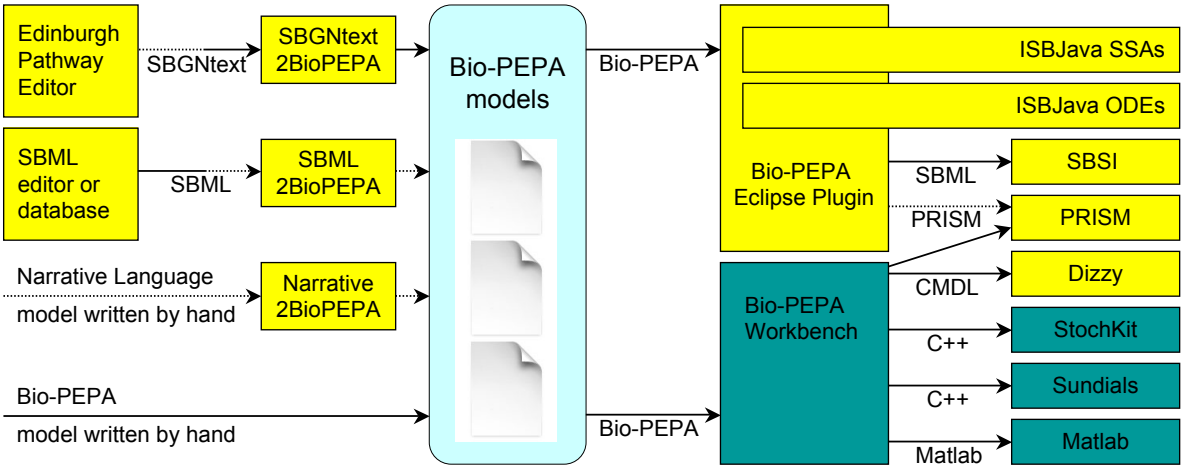


Figure 2: **Overview of the Bio-PEPA tools.** Rectangles indicate independent executables written in Java (yellow or light grey) or other languages (blue or dark grey) and arrows symbolise files that encode the model of interest in the corresponding format. The Bio-PEPA Workbench mostly serves as a prototype that facilitates exploration of new analysis tools while the Bio-PEPA Eclipse Plug-in is being developed to become the primary, user-friendly tool. The standard language of choice for new tools is Java in order to facilitate their eventual integration into Eclipse plug-ins. Dotted lines indicate that the corresponding functionality is not yet implemented.

"Bio-PEPA Workbench" made extensive use of this mechanism to quickly implement access to a wide range of third party tools for Bio-PEPA.

The organisation of Bio-PEPA tools into tool chains that translate from one model representation into another has important advantages for software development and design:

- *Minimal dependencies.* Tool chains are highly modular and result in a powerful separation of concerns that substantially reduces the possibility to introduce bugs. Each tool is well shielded from any side-effects of changes in other tools as long as the syntax and semantics of the file formats is not affected. Thus software developers only need to ensure that both tools understand the same file format.
- *Maximal independence of software developers.* Software development within a university context brings its own set of challenges, which are substantially different from those faced by industry. A key difference is that within universities software developers need to be much more independent, as they are rewarded for innovative contributions that are easily attributable to them. This can make it difficult to implement complex designs that require the coordinated effort of many developers. Allowing independent researchers to contribute a recognisable tool in a reasonable amount of time is highly advantageous in a university context. Our loosely coupled tool chains approach facilitates this, as different researchers can work on different modules without communicating, as long as they agree on the syntax and semantics of the related file formats. This approach also helps researchers working on the theory of new analysis methods to quickly link their work into existing tool chains by simply adding a translation mechanism.
- *Maximal code reuse.* The independence of the various modules facilitates maximal code reuse, since a well tested tool can be used by others as a black box without the need for further testing if correct input can be provided. This substantially reduces the complexity of the implementation of an environment for the wide range of analyses supported by Bio-PEPA.

All these advantages come at the price of introducing the threat of divergence and incompatibility between the reading module and the writing module of each file format. If developers are too independent and do not plan ahead careful enough, then two interfaces that should implement one format will end up speaking different dialects that can make interoperability difficult. One can of course introduce a further translation to resolve this, but the resulting increase of overall system complexity is undesirable. Thus all developers need to take some care in order to minimise the complexity of the various translations that glue the system together.

In developing these tools for the systems biology community we took the position that we were not domain experts in this area and so we wished to ensure that we maintained an active dialogue with modellers who wanted to use the tools, and that we responded quickly to their requests for feature enhancements. Several of these were for the addition of features which we would not have developed had requests not come from the users of our tools.

## 4 THE BIO-PEPA TOOLS

As seen in Figure 2, there exist two main tools for working with Bio-PEPA models, the Bio-PEPA Workbench and the Bio-PEPA Eclipse Plug-in. While both are under active development, they each serve a particular role. The Bio-PEPA Workbench is a prototype tool for introducing new language features and types of analysis. The Bio-PEPA Eclipse Plug-in is an environment which targets end-users wishing to model in Bio-PEPA. In addition, various tools are under development that enable the production of Bio-PEPA models by implementing mappings from other biologist orientated formalisms. For example, a textual representation of SBGN process diagrams has been developed and can be compiled to Bio-PEPA by the SBGNtext2BioPEPA tool(SBGNtext2BioPEPA ) in order to build a bridge from visual SBGN editors like the Edinburgh Pathway Editor to Bio-PEPA. Efforts are also under way to generate Bio-PEPA models from the "Narrative Language", which is a textual narrative-style language explicitly designed for making biochemical modelling easy for biologists (Guerriero, Heath, and Priami 2007). Lastly, a prototype has been independently developed to translate SBML models to Bio-PEPA. Since all these modules are developed in Java, they can easily be included in the Eclipse Plug-in for Bio-PEPA or be deployed as stand-alone applications as indicated in Figure 2.

### 4.1 The Bio-PEPA Workbench

The Bio-PEPA Workbench (named for historical reasons) is a lightweight modelling tool chain whose primary function is to compile Bio-PEPA models into a variety of formats for analysis. The Workbench can export to well-known tools such as Matlab and PRISM but also allows users to work with the StochKit (Li et al. 2008) and the Sundials ODE solver suite (Hindmarsh et al. 2005). StochKit is a C++ library for stochastic simulation and Sundials is a suite of numerical software which includes numerical integrators for systems of ordinary differential equations.

Written in the functional programming language ML, the Workbench is ideally suited for prototyping and acts as a testbed for new ideas or alternative solvers. The functions that constitute the Workbench can be separated into two distinct groups. The first group transforms the Bio-PEPA file into a data structure that represents the model in ML. This structure can then be used as the input to any function in the second group, each written to output a different format as indicated in Figure 2. Exporting to a new tool can be as simple as writing a new function to output a string in the correct format, a task that ML is ideally suited to. Where possible and required, the Workbench also generates scripts to automate the compilation and execution of the relevant tool before passing on the results to gnuplot. The use of the Workbench for prototyping new analysis methods implies that it expects modellers to be already experienced with Bio-PEPA, as it does not help in the generation of Bio-PEPA models. As the Workbench is primarily a tool for prototyping, design aspects such as code reuse and intuitive user interfaces were secondary priorities.

### 4.2 The Bio-PEPA Eclipse Plug-in

In contrast to the Workbench, the purpose of the Bio-PEPA Eclipse Plug-in (from now on plug-in) is to offer a complete environment for working with Bio-PEPA models. The plug-in is an extension to the Eclipse Platform, an Integrated Developing Environment (IDE) for popular programming languages such as Java and C++ that is itself built in Java to allow deployment to multiple operating systems. Many of the design decisions regarding the development of the plug-in were influenced by the design of the Eclipse platform as well as prior PEPA software projects. Copying decisions made for the PEPA Eclipse Plug-in (Tribastone 2006), the plug-in consists of an Eclipse front-end and a separate back-end library for all of the functionality required for working with Bio-PEPA models (Figure 3).

The back-end library has been designed to be independent not only from the front-end but also from file system calls to avoid the issue of relying on platform dependent file objects. This has been an issue when dealing with environments such as Netbeans (in the case of a previous tool called Choreographer) and is equally pertinent in Eclipse. Instead, the tool exposes a small set of high-level calls designed to either allow parsing and compilation of the Bio-PEPA model from a supplied String object or to bypass the parsing by programmatically constructing the Abstract Syntax Tree directly. This allows developers wishing to access the features of the plug-in the possibility of bypassing the generation of yet another textual representation. Once parsed the model can either be exported to an alternative formats like SBML or be analysed by solvers to produce

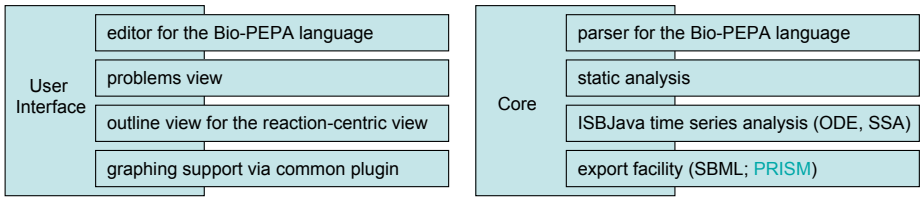


Figure 3: Overview of the components contained in the Bio-PEPA Eclipse Plug-in.

time-series. While the Workbench relies on external tools, the plug-in has 3rd party stochastic simulators and numerical integrators built-in, making it much easier for non-expert users to install and use. The solvers are contained within the ISBJava library, the back-end to Dizzy. Through ISBJava, the plug-in currently supports two numerical integrators for ODEs, a Runge-Kutta implicit-explicit implementation and the Dormand-Prince adaptive step-size solver, both through the odeToJava library. Stochastic simulations are enabled through Gillespie's SSA, Gibson-Bruck and the Tau-Leap algorithm. Comparing simulation results and ODE solutions has allowed us to discover errors in published modelling studies in computational biology (Calder et al. 2006). This illustrates the strength of a high-level modelling language such as a process calculus.

The front-end is specific to Eclipse and adopts the approach of plug-ins and the registering of services. The plug-in registers the file extension `.biopepa` before registering views and editors to the file type. This allows Eclipse to load plug-ins as required for handling any given file, as well as defining the services required for the file type in question. For Bio-PEPA models the front-end defines a built-in editor, enhanced by syntax highlighting of keywords and operators. It makes use of existing Eclipse views, such as a unified location for reporting errors caught during the parsing of the model and during static analysis. The plug-in differentiates between those that stop analysis of the model (errors) and those that may suggest a mistake when creating the model (warnings). An example of a warning would be the declaration of an unused variable or species. While the species may have been forgotten when constructing the model, it may also have been specifically excluded by the modeller in a particular simulation. Debugging models is also facilitated by the Outline view of the plug-in that lists all reactions with the species that are affected by them (the reaction-style representation that is frequently used by other tools). Access to both views helps modellers to quickly assess how their models are interpreted by the plug-in and hence to find corresponding errors.

Once the model is deemed correct it can be analysed using the solvers available to the plug-in. At this point the front-end displays the results to allow inspection by the modeller. Any obvious errors in the results can be spotted quickly in this way before the results are exported for graphing in the modeller's preferred tool. Examples of all these UI components can be seen in Figure 4.

The benefits of adopting the Eclipse approach are best demonstrated by highlighting the integration of the plug-in with "SBSI Visual", part of the Systems Biology Software Infrastructure currently under development at the Centre for Systems Biology at Edinburgh. SBSI is an expandable initiative designed to facilitate many aspects of computational systems biology, including parallelised parameter optimisation algorithms, access to shared data and model repositories, as well as access to supercomputers. SBSI Visual is a desktop client application that provides access to background computing services. The easy addition of support for new tools and algorithms in SBSI Visual is facilitated by the plug-in mechanisms of the Eclipse Rich Client Platform, which is a custom-built instance of Eclipse stripped of all but the plug-ins required to support the tool at hand. This substantially reduces size, as superfluous plug-ins are removed to create a smaller, more focussed application. Integrating SBSI Visual and the Bio-PEPA Eclipse Plug-in will allow Bio-PEPA modellers to analyse systems much faster than by using locally run solvers and also provide access to parameter optimisation facilities. In return, current users of SBSI Visual are exposed to the advantages of process algebras.

The integration work started with supporting the export to SBML within the core Bio-PEPA library. The design of Eclipse makes incorporating the Bio-PEPA Plug-in into SBSI Visual as simple as downloading the plug-in and its dependencies. This of course does not integrate the two environments, which at this point are simply co-existing within the instance of Eclipse. Several options are available that vary how tightly the tools are integrated. The SBML can be passed on internally, either by building an awareness of the respective plug-in into both SBSI Visual and Bio-PEPA or through developing a third plug-in to bridge the existing two. The SBML can also be passed on externally through the Eclipse file system by offering the ability to export the SBML in the Bio-PEPA Eclipse Plug-in. This is a much looser coupling that appears to result in little interaction. However, such a solution removes the need for more complex plug-in interactions and in fact fits in with how the user is expected to interact with SBSI Visual.

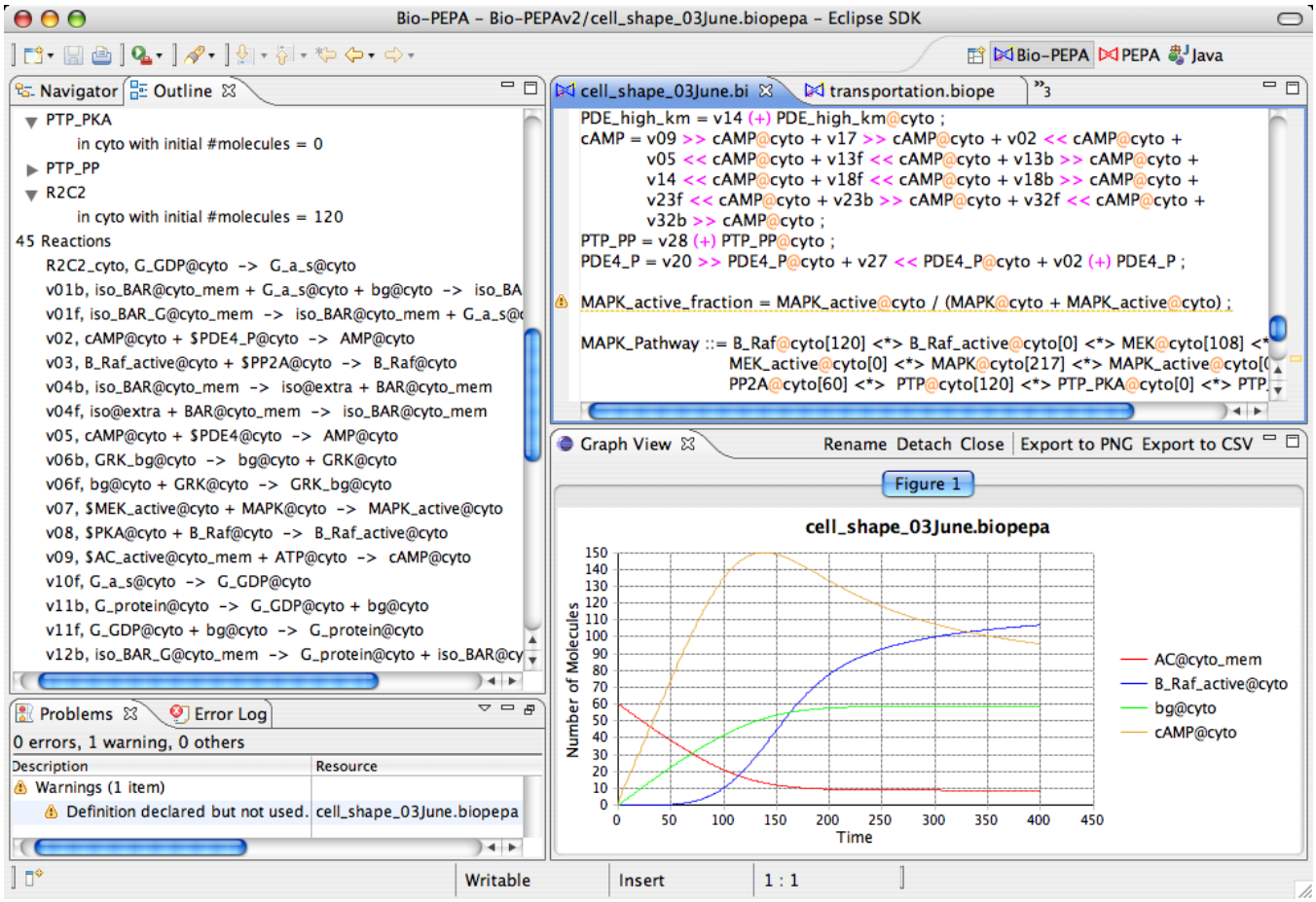


Figure 4: A screenshot of the Bio-PEPA Eclipse Plug-in. Starting in the top-right corner of the window and moving anti-clockwise we have the Bio-PEPA Editor pane, the Outline view of the model, the Problems view and a Graph view displaying results in the final quadrant.

## 5 CASE STUDIES

Bio-PEPA has been applied to model a number of biologically relevant systems, some of which are described in the following.

***Neurospora crassa* circadian clock.** A large Bio-PEPA case study is presented in (Akman et al. 2009), where a Bio-PEPA model of the circadian clock of the fungus *Neurospora crassa* is developed, based on an existing ODE model. Circadian clocks are oscillatory gene networks, present in nearly all living organisms, whose function is to regulate the expression of specific mRNAs and proteins to adapt to the 24 hour day/night cycle. Using the Bio-PEPA Workbench, both discrete-stochastic and continuous-deterministic results are obtained using Dizzy, the front-end to ISBJava, in order to investigate the effect of stochasticity on the robustness of the clock's behaviour. Studying the variations in the phase and amplitude of oscillations of protein amounts showed that oscillations are more robust in the stochastic than in the deterministic scenarios. The time-dependent sensitivity of the model is investigated with respect to some key kinetic parameters in order to determine the time within the circadian cycle at which the system is most responsive to parameter variations. This work used a novel sensitivity analysis method implemented in an extended version of Dizzy (Degasperi and Gilmore 2008). The light on/off mechanism responsible for entrainment to the day/night cycle is expressed using discrete time-dependent events in Bio-PEPA (Ciocchetta 2009). Such events are translated into time-dependent reaction rates in the Dizzy model (defined in terms of a step function).

**Gp130/JAK/STAT signalling pathway.** Bio-PEPA is used to model the Gp130/JAK/STAT signalling pathway (Guerriero 2009). This is a well-studied biological system of great clinical interest because of the major role it plays in several biological processes both in human and other organisms. Stochastic simulation and model-checking are used in

order to study various qualitative and quantitative aspects of the system, which are then compared with results from existing models. PRISM is used to verify several desired qualitative and quantitative properties on the PRISM model generated by the Bio-PEPA Workbench. Qualitative properties are defined in order to check for the presence of possible human errors that are easy to introduce but hard to find in large models (see *consistency checks* in Section 2.2). Quantitative analysis is performed by generating time-series data using the Dizzy stochastic simulator and by verifying reward-based properties in PRISM. PRISM was also used to compute a few (semi-)quantitative measures such as the number of occurrences of a given reaction and the relative probabilities of consumption of different reactants.

**Genetic network in *E. Coli*.** An abstract representation of a genetic network with negative feedback is used to explore the control circuit for the $\lambda$ repressor protein CI of the $\lambda$-phage in *E. Coli* by integrating stochastic simulation and model-checking (Ciocchetta et al. 2009). This work focused on reducing the state space of PRISM models by parameterising them with the help of stochastic simulations. This is done by deriving upper and lower bounds for molecular amounts from averaging over the results of many stochastic simulation runs and by determining interesting time intervals on which model-checking should focus. The problem of identifying bounds is crucial for structurally unbounded systems where molecular amounts could grow indefinitely in theory, even if they are generally limited by degrading reactions in practice. This work on the genetic network is exemplary for similar systems and shows how the correct quantitative behaviour can be observed through model-checking when *a priori* bounds are imposed on molecular amounts.

**Nicotin Acetylcholine Receptors.** The functional properties of nicotin acetylcholine receptors in Edelstein's model (Edelstein et al. 1996) are studied with the help of time-dependent events (Ciocchetta 2009). These receptors are transmembrane channel proteins that switch between an open and a closed state under the control of neurotransmitters. The removal of free agonists at a given time is represented by a Bio-PEPA event that describes the triggering condition and the resulting change in the system. This event blocks ligand binding reactions, which is equivalent to completely removing ligands. The behaviour of the original model (Edelstein et al. 1996) could be replicated by stochastic simulations that are explicitly capable of handling time-dependent events precisely.

**Intracellular calcium oscillations.** Intracellular $Ca^{2+}$ oscillations are modelled to demonstrate Bio-PEPA's support for multiple compartments (Ciocchetta and Guerriero 2009). Intracellular $Ca^{2+}$ oscillations are observed in a large variety of cell types and play an important role in the control of many cellular processes. Existing models demonstrate simple periodic oscillations for $Ca^{2+}$ (Borghans, Dupont, and Goldbeter 1997). The Bio-PEPA model is based on a simple model referred to as *CICR* ($Ca^{2+}$ -induced $Ca^{2+}$ release) (Borghans, Dupont, and Goldbeter 1997). In CICR, the transport of molecules among compartments plays a major role ($Ca^{2+}$ shuttles between the extracellular environment, the cytosol, and the sarcoplasmic reticulum). Bio-PEPA is able to handle the explicit locations and the reactions involving multiple locations and results show a good agreement with the original model (Ciocchetta and Guerriero 2009) .

## 6 CONCLUSIONS

Here we have shown the many benefits that a well designed high-level language with active tool development can bring to modelling. It allows the modeller to access powerful analytical techniques from a variety of tools while simultaneously removing the requirement to become an expert in several specialist modelling languages. The automated generation of the model through well-defined mappings limits the possibility of mistakes when dealing with complex or dense interactions. The results from different techniques can also be compared and the parsing and static analysis of the model can help to discover a variety of common errors when creating a model.

Future work with the Bio-PEPA tools involves developing additional mappings from formalisms such as the user-centred Narrative Language and the Systems Biology Graphical Notation (SBGN) as supported by graphical editors. We hope that these developments will encourage users in the Life Sciences to adopt the Bio-PEPA tools, as they will be able to obtain the benefits of using a high-level modelling language without having to work directly at the process calculus level.

**Availability:** The Bio-PEPA tools are freely available for download from `http://www.biopepa.org`.

## ACKNOWLEDGEMENTS

Adam Duguid is supported by the EPSRC Doctoral Training Grant EP/P501407/1. Stephen Gilmore and Maria Luisa Guerriero are supported by the EPSRC grant EP/E031439/1 "Stochastic Process Algebra for Biochemical Signalling Pathway Analysis". Jane Hillston is supported by the Engineering and Physical Sciences Research Council (EPSRC) Advanced Research Fellowship and research grant EP/C543696/1 "Process Algebra Approaches to Collective Dynamics". Laurence Loewe is supported by the Centre for Systems Biology at Edinburgh. The Centre for Systems Biology at Edinburgh is a Centre for Integrative Systems Biology (CISB) funded by BBSRC and EPSRC, reference BB/D019621/1.

## REFERENCES


Akman, O., F. Ciocchetta, A. Degasperi, and M. L. Guerriero. 2009. Modelling Biological Clocks with Bio-PEPA: Stochasticity and Robustness for the *Neurospora Crassa* Circadian Network. In *CMSB 2009*, LNCS.

Borghans, J., G. Dupont, and A. Goldbeter. 1997. Complex intracellular calcium oscillations: A theoretical exploration of possible mechanisms. *Biophysical Chemistry* 66 (1): 25–41.

Calder, M., A. Duguid, S. Gilmore, and J. Hillston. 2006. Stronger computational modelling of signalling pathways using both continuous and discrete-state methods. In *Proceedings of the Fourth International Conference on Computational Methods in Systems Biology (CMSB 2006)*, ed. C. Priami, Volume 4210 of *Lecture Notes in Computer Science*, 63–77. Trento, Italy: Springer.

Calder, M., S. Gilmore, and J. Hillston. 2004, August. Modelling the influence of RKIP on the ERK signalling pathway using the stochastic process algebra PEPA. In *Proceedings of the BioConcur Workshop on Concurrent Models in Molecular Biology*, ed. A. Ingolfsdottir and H. R. Nielson. London, England.

Calder, M., and J. Hillston. 2009. Process algebra modelling styles for biomolecular processes. *Transactions on Computational Systems Biology*. To appear.

Ciocchetta, F. 2009. Bio-PEPA with SBML-like events. *TCSB special issue on Computational models for cell processes*.

Ciocchetta, F., S. Gilmore, M. L. Guerriero, and J. Hillston. 2009. Integrated simulation and model-checking for the analysis of biochemical systems. *Electr. Notes Theor. Comput. Sci.* 232:17–38.

Ciocchetta, F., and M. L. Guerriero. 2009. Modelling Biological Compartments in Bio-PEPA. In *Proceedings of the 2nd International Meeting on Membrane Computing and Biologically Inspired Process Calculi (MeCBIC 2008)*, Volume 227 of *ENTCS*, 77–95.

Ciocchetta, F., and J. Hillston. 2008. Bio-PEPA: An extension of the process algebra PEPA for biochemical networks. *Electr. Notes Theor. Comput. Sci.* 194 (3): 103–117.

Ciocchetta, F., and J. Hillston. 2009. Bio-PEPA: A framework for the modelling and analysis of biological systems. *Theoretical Computer Science* In Press, Corrected Proof. To appear.

Danos, V., J. Feret, W. Fontana, R. Harmer, and J. Krivine. 2007, 3–8 September. Rule-based modelling of cellular signalling. In *Proceedings of the Eighteenth International Conference on Concurrency Theory, CONCUR'2007, Lisbon, Portugal*, ed. L. Caires and V. Vasconcelos, Volume 4703 of *Lecture Notes in Computer Science*, 17–41. Lisbon, Portugal: Springer, Berlin, Germany.

Degasperi, A., and S. Gilmore. 2008, June. Sensitivity analysis of stochastic models of bistable biochemical reactions. In *Formal Methods for Computational Systems Biology, 8th International School on Formal Methods for the Design of Computer, Communication, and Software Systems*, ed. M. Bernardo, P. Degano, and G. Zavattaro, Volume 5016 of *Lecture Notes in Computer Science*, 1–20: Springer.

Edelstein, S., O. Schaad, E. Henry, B. D., and J. Changgeux. 1996, November. A kinetic mechanism for nicotinic acetylcholine receptors based on multiple allosteric transitions. *Biological cybernetics* 75 (5): 361–379.

Faeder, J. R., M. L. Blinov, and W. S. Hlavacek. 2009. Rule-based modeling of biochemical systems with bionetgen. *Methods in molecular biology (Clifton, N.J.)* 500:113–167.

Fages, F., and S. Soliman. 2008, June. Formal cell biology in biocham. In *Formal Methods for Computational Systems Biology, 8th International School on Formal Methods for the Design of Computer, Communication, and Software Systems*, ed. M. Bernardo, P. Degano, and G. Zavattaro, Volume 5016 of *Lecture Notes in Computer Science*, 54–80: Springer.

Galpin, V., and J. Hillston. 2009. Equivalence and discretisation in Bio-PEPA. In *CMSB*, LNCS.

Gibson, M., and J. Bruck. 2000. Efficient exact stochastic simulation of chemical systems with many species and many channels. *J. Comp. Phys.* 104:1876–1889.

Gillespie, D. 1977. Exact stochastic simulation of coupled chemical reactions. *Journal of Physical Chemistry* 81 (25): 2340–2361.

Gillespie, D. T. 2008, June. Simulation methods in systems biology. In *Formal Methods for Computational Systems Biology, 8th International School on Formal Methods for the Design of Computer, Communication, and Software Systems*, ed. M. Bernardo, P. Degano, and G. Zavattaro, Volume 5016 of *Lecture Notes in Computer Science*, 125–167: Springer.

Guerriero, M. L. 2009. Qualitative and Quantitative Analysis of a Bio-PEPA Model of the Gp130/JAK/STAT Signalling Pathway. *TCSB special issue on Computational models for cell processes*.

Guerriero, M. L., J. K. Heath, and C. Priami. 2007. An Automated Translation from a Narrative Language for Biological Modelling into Process Algebra. In *Proceedings of Computational Methods in Systems Biology (CMSB'07)*, Volume 4695 of *LNCS*, 136–151.

Heath, J., M. Kwiatkowska, G. Norman, D. Parker, and O. Tymchyshyn. 2008. Probabilistic Model Checking of Complex Biological Pathways. *Theoretical Computer Science* 319 (3): 239–257.

Heiner, M., R. Richter, M. Schwarick, and C. Rohr. 2008, April. Snoopy - a tool to design and execute graph-based formalisms. *Petri Net Newsletter* 74:8–22.

Hillston, J. 1996. *A compositional approach to performance modelling*. Cambridge University Press.

Hindmarsh, A. C., P. N. Brown, K. E. Grant, S. L. Lee, R. Serban, D. E. Shumaker, and C. S. Woodward. 2005. Sundials: Suite of nonlinear and differential/algebraic equation solvers. *ACM Transactions on Mathematical Software* 31 (3): 363–396.

Kitano, H. 2002, November. Computational systems biology. *Nature* 420:206–210.

Kwiatkowska, M., G. Norman, and D. Parker. 2002. PRISM: Probabilistic symbolic model checker. In *TOOLS'02*, LNCS 2324, 200–204: Springer-Verlag.

Li, H., Y. Cao, L. R. Petzold, and D. T. Gillespie. 2008. Algorithms and software for stochastic simulation of biochemical reacting systems. *Biotechnology Progress* 24 (1): 56–61.

Loewe, L. 2009. A framework for evolutionary systems biology. *BMC Systems Biology* 3:27.

Phillips, A. 2009. *Symbolic systems biology: Theory and methods*, Chapter A Visual Process Calculus for Biology. Jones and Bartlett. In Press.

Ramsey, S., D. Orrell, and H. Bolouri. 2005. Dizzy: stochastic simulation of large-scale genetic regulatory networks. *J. Bioinf. Comp. Biol.* 3 (2): 415–436.

Regev, A., and E. Shapiro. 2002, September. Cells as computation. *Nature* 419 (6905): 343.

SBGNtext2BioPEPA. http://csbe.bio.ed.ac.uk/SBGNtext2BioPEPA/index.php.

Schoeberl, B., C. Eichler-Jonsson, E. D. Gilles, and G. Müller. 2002, April. Computational modeling of the dynamics of the MAP kinase cascade activated by surface and internalized EGF receptors. *Nature Biotechnology* 20:370–375.

Shampine, L. F., and M. W. Reichelt. 1997. The Matlab ODE suite. *SIAM J. Sci. Comput.* 18 (1): 1–22.

Tribastone, M. 2006, September. The PEPA plug-in project. In *Third International Conference on the Quantitative Evaluation of Systems*, 53–54: IEEE Computer Society Press.

## AUTHOR BIOGRAPHIES

**ADAM DUGUID** is a PhD student at Edinburgh University under the supervision of Stephen Gilmore. He is the developer of The Bio-PEPA Eclipse Plug-in and one of the main developers on the PEPA Eclipse Plug-in. His email address for these proceedings is `a.j.duguid@sms.ed.ac.uk`.

**STEPHEN GILMORE** is a Reader in the School of Informatics at the University of Edinburgh. He received his PhD from the Queen's University of Belfast in Northern Ireland. His research interests include quantitative modelling of systems using process algebras and the design and development of software tools to support the modelling process. His email address for these proceedings is `Stephen.Gilmore@ed.ac.uk`.

**MARIA LUISA GUERRIERO** is a Research Associate in the School of Informatics at the University of Edinburgh, UK. She received her Ph.D. in Bioinformatics from the University of Trento, Italy. Her research work is in the area of systems biology, and her research interests include computational modelling of biochemical systems, and the application of process algebra and model-checking techniques for simulation and formal verification. Her email address is `mguerrie@inf.ed.ac.uk`.

**JANE HILLSTON** holds a personal chair in quantitative modelling at The University of Edinburgh. Her PhD thesis was awarded the British Computer Society/Conference of Professors and Heads of Computing Distinguished Dissertation award in 1995. In 2004 she won the British Computer Society/Microsoft Roger Needham award. In 2005 she was awarded a five-year Advanced Research Fellowship from the UK Engineering and Physical Sciences Research Council. She was made a Fellow of the Royal Society of Edinburgh in 2007. Her email address for these proceedings is `Jane.Hillston@ed.ac.uk`.

**LAURENCE LOEWE** initiated and runs evolution@home, the first global computing simulation system for evolutionary biology. After a short period lecturing in evolutionary genetics he joined CSBE as a postdoc modelling molecular systems using Bio-PEPA. He initiated the first ESEB symposium on evolutionary systems biology and can be reached under `Laurence.Loewe@ed.ac.uk`.