BIOINFORMATICS APPLICATIONS NOTE Vol. 30 no. 10 2014, pages 1491–1492
doi:10.1093/bioinformatics/btu050

Systems biology

Advance Access publication January 27, 2014

Bio Simulators: a web UI for biological simulation

Michael Pedersen[1,2,*], Nicolas Oury[3], Colin Gravill[2] and Andrew Phillips[2,*]

[1]Department of Plant Sciences, Cambridge University, Cambridge CB2 3EA, UK, [2]School of Informatics, Edinburgh University, Edinburgh EH8 9AB, Scotland and [3]Microsoft Research, Cambridge CB1 2FB, UK

Associate Editor: Alfonso Valencia

ABSTRACT

Summary: A host of formal, textual languages for modeling cellular processes have recently emerged, but their simulation tools often require an installation process which can pose a barrier for use. Bio Simulators is a framework for easy online deployment of simulators, providing a uniform web-based user interface to a diverse pool of tools. The framework is demonstrated through two plugins based on the KaSim Kappa simulator, one running directly in the browser and another running in the cloud.

Availability: Web tool: bsims.azurewebsites.net. KaSim client side simulator: github.com/NicolasOury/KaSimJS. KaSim cloud simulator: github.com/mdpedersen/KaSimCloud.

Contact: michael.d.pedersen@gmail.com or Andrew.Phillips@microsoft.com

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

Received on September 5, 2013; revised on December 28, 2013; accepted on January 22, 2014

1 INTRODUCTION

The last decade has seen the emergence of a host of formal languages for modeling cellular processes in systems biology. Examples include process calculi such as the Pi calculus, and rule-based languages (Chylek *et al.*, 2013) such as Kappa (Danos *et al.*, 2007). However, simulation tools often require installation on a local machine, a process which at best poses a barrier for use by the casual observer and at worst is riddled with non-trivial dependency problems.

Advances in web technology enable simulation tools to be made available online, directly through a browser. Bio Simulators is a web framework providing basic user interface components which are common to many simulators: an editor with syntax highlighting for writing textual models, and a plot for showing the results of time course simulation.

A screenshot is shown in Figure 1. At the top is a listing of available simulators. A new simulator can be added simply by pressing the '+' button and entering a URL to a simulator definition. The simulator definition, in turn, is a file in the JSON format which specifies a URL to a JavaScript simulator file together with meta-information for, e.g. syntax highlighting in the given language.

Bio Simulators thus facilitates easy prototyping and deployment of novel simulators, made available directly through a web browser with no need for user interface development. Bio Simulators also provides a foundation for online model repositories in which simulation of a model can be started simply by pressing a link which points to Bio Simulators; the link encodes an URL to the relevant simulator and to the model. Finally, assuming a critical mass of simulator plugins, Bio Simulators can serve as a uniform interface for learning and experimenting with the plethora of languages and simulators available.

We demonstrate the concepts with two Bio Simulators plugins, both of which provide simulators for models in the Kappa language and are based on the open source KaSim tool available at www.kappalanguage.org. In the first case, KaSim is translated to JavaScript and the resulting simulator plugin runs entirely within the browser. In the second case, KaSim is deployed to an Azure Cloud service. We end by outlining the simulator definition interface file format.

2 METHODS AND IMPLEMENTATION

2.1 A Kappa client side simulator

The KaSim tool is written in the OCaml language. We first compiled the KaSim source code to an intermediate OCaml bytecode representation. We then compiled this intermediate representation to JavaScript using the js_of_ocaml tool from Ocsigen. To do so, a number of extensions were developed to handle, e.g. file I/O which is needed by KaSim but is not directly supported by JavaScript.

Simulations using the resulting JavaScript code run ~25 times slower than simulations using the native KaSim executables. Better performance may be achievable using other means of compiling OCaml to JavaScript, for example by going through the C language as an intermediate step. However, JavaScript engines have already seen dramatic performance improvements over recent years, and this trend is likely to continue.

2.2 A Kappa cloud simulator

When performance is significant, the native KaSim simulator can be executed on a cloud server. The Bio Simulators plugin is still given as a JavaScript file, but the JavaScript file is simply a thin layer which communicates between Bio Simulators and the cloud server.

We have developed an Azure Cloud application for this purpose, written in C# and using the SignalR framework for pushing simulation results to the JavaScript plugin. The application comes packaged as two files which can be deployed to the Azure Cloud following the steps outlined below; full deployment details are available in the online Supplementary Material:

(1) Navigate to the Azure management portal web site.

(2) Start a cloud service instance by specifying an Azure URL prefix and uploading the two Bio Simulators plugin files.

(3) Click on the link provided on the emerging web page to automatically install the cloud plugin in Bio Simulators.

*To whom correspondence should be addressed.

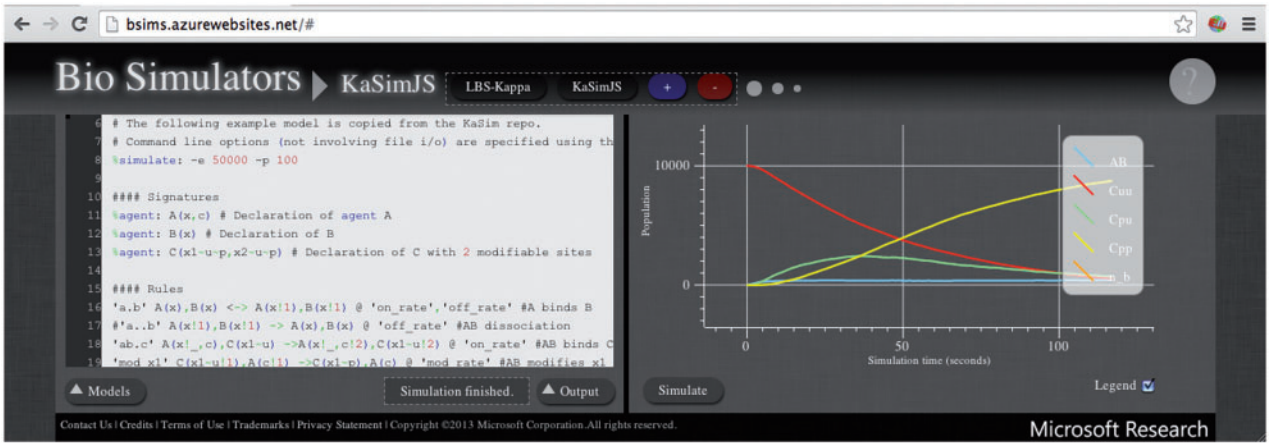

Fig. 1. A screenshot of the Bio Simulators tool with the KaSim simulator selected

Azure Cloud services are billed by the hour with costs depending on server specifications. The flexibility in performance and pricing is a key advantage of cloud services: a very powerful instance can be purchased for exactly the required duration, e.g. for just an hour or a day. Another advantage is the simplicity of deployment; there is no need for server administration or maintenance. Note that although we have used Azure, a similar approach can be followed for other cloud platforms such as Amazon's EC2.

Listing 1. Specification file for the KaSim cloud simulator.

```
{"name": "KaSim Cloud",
 "simulatorURL":  "http://.../simulator.js",
 "defaultModelURL": "http://.../ex.js",
 "exampleModels": {"Model1": "http://.../ex.js"},
 "keywords":["init", "obs", "agent", "simulate"],
 "symbols": ["%", "~", "!", ",", "->", "\\?"],
 "useJSONP": true,
 "runInWorker": false }
```

2.3 The simulator-specification file

An outline of the specification file for the KaSim cloud simulator is shown in Listing 1. The first line specifies the simulator name. An optional logo URL can also be given but is not shown here.

The second line specifies an URL to the JavaScript simulation file. This file must define two functions, one called 'simulate' which takes a model to be simulated and a callback function for updating the plot, and another function called 'stop' for stopping the simulation. Note that simulation parameters are provided implicitly through the model text, for example using a '%simulate' directive as shown in the first line of the model in Figure 1. Here, following the KaSim command line options, the -e flag specifies the number of simulation events and the -p flag specifies the number of data points to plot.

Lines 3 and 4 in the specification file provide Kappa model URLs. Lines 5 and 6 specify syntax highlighting. The last two lines provide details on how the JavaScript file is loaded and executed.

3 CONCLUSION

We have introduced a web UI framework for biological simulators and demonstrated its flexibility by providing a Kappa simulator plugin which runs directly in the client and another which runs in the cloud. These are to our knowledge the only Kappa simulators currently available directly through a web browser, although online simulators do exist for the related BNG language (vcell.org/bionetgen).

Funding: An Engineering and Physical Sciences Research Council Postdoctoral Fellowship [EP/H027955/1] (in part).

Conflict of Interest: none declared.

REFERENCES

Chylek,L.A. *et al.* (2013) Innovations of the rule-based modeling approach. In: *Systems Biology: integrative biology and simulation tools*. Springer, Dordrecht.

Danos,V. *et al.* (2007) Rule-based modelling of cellular signalling. In: *CONCUR*. Vol. 4703 of *Lecture Notes in Computer Sciences*, pp. 17–41. Springer, Berlin.