

Big Red: A Development Environment for Bigraphs

Alexander Faithfull, Gian Perrone, and Thomas T. Hildebrandt

IT University of Copenhagen, Denmark
{alef,gdpe,hilde}@itu.dk

Abstract. We present Big Red, a visual editor for bigraphs and bigraphical reactive systems, based upon Eclipse. The editor integrates with several existing bigraph tools to permit simulation and model-checking of bigraphical models. We give a brief introduction to the bigraphs formalism, and show how these concepts manifest within the tool using a small motivating example bigraphical model developed in Big Red.

Keywords: bigraphs, editor, reactive systems

1 Introduction

Bigraphical reactive systems are a class of graph-rewriting systems designed to capture orthogonal notions of *connectivity* and *locality* through the use of two graph structures — a *place* graph, and a *link* graph. They were first proposed by Robin Milner [5] to address the challenges associated with modelling of ubiquitous computing applications. Bigraphs have been successful in capturing the syntax and semantics of a number of well-known formalisms and real-world applications¹.

The Big Red tool is a prototype editor to support the development of bigraphs and bigraphical reactive systems in a visual manner. It interfaces with existing bigraph tools such as the BigMC bigraphical model checker [6] to permit the execution of models. Big Red aims to make bigraphs more accessible to novice users, as well as providing development support to more experienced bigraph users. Bigraphs have a visual presentation that is formal and unambiguous, and one of the major benefits is the ability to present a relatively complex bigraphical model in a way that is comprehensible by non-experts. This is the motivation for the development of Big Red: making it easier to create and interact with bigraphs increases the applicability and utility of the formalism in more diverse application areas.

Below we first briefly in Section 2 describe the previous efforts to implement bigraphical reactive systems. We then proceed in Section 3 describing how bigraphs are expressed in Big Red, using a small example of a context-aware printing system, inspired by that given in [1]. Finally, Section 4 briefly describes the implementation of the tool, and suggests ways in which it may be extended using additional modules.

¹ Some examples are available from <http://bigraph.org/papers/gcm2012/>.

2 Related Work

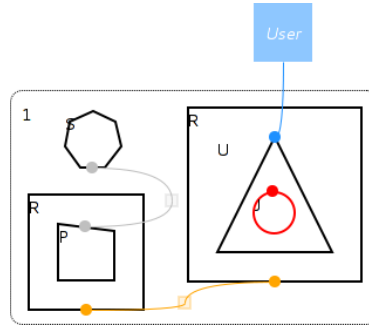
One of the first attempts at creating a graphical editor for bigraphs was within the Bigraphspace [3] project in 2009, during which a prototype bigraph editor based upon eclipse was developed; however, this work was never completed, and no usable editor currently exists. Bigraphspace used the correspondence between the structure of bigraphs and XML documents [4] to provide a tuplespace-like API with which to manipulate bigraphs. Big Red differs from these efforts in that it implements editing of Milner’s bigraphs in such a way as to enable external tools to perform further analysis of these models, rather than imposing a particular (extra-bigraphical) semantics upon the models.

To date, bigraphical reactive systems have been largely implemented using term representations of bigraphs, such as that used in the BPLtool [2], or in the BigMC model checker [6]. While this is appropriate for bigraph experts, it presents a significant learning curve for novice users, and ignores the benefits provided by the formal graphical syntax provided by bigraphs.

3 Bigraphs in Big Red by Example

A bigraph is a forest of node labelled trees called the *place graph*, the roots of which are indexed by integers, and referred to as *regions*. The place graph parent relationship is usually drawn by nesting in the graphical syntax. The label of a non-root place graph node is referred to as its *control*, and is drawn from the bigraph *signature*. We will use “X node” to mean a node that is labelled with the control X. The control of a node also defines the number of *ports* of the node, referred to as the *arity* of the control and provided by a function $ar : \Sigma \rightarrow \omega$ given as part of the signature. The link graph can be viewed as a directed hypergraph with edges mapping a subset of the complete set of all ports of nodes in the place graph and the *inner* names of the bigraph to either a (single) outer name or a single edge. By convention, outer names are drawn above the bigraph, while inner names are drawn below the bigraph.

Fig. 1 is an example bigraph constructed within Big Red for a context-aware printing system, inspired by that given in [1]. The example scenario involves a building in which users can submit print jobs to a print spool, and then move into a room with any printer connected to that print spool, at which point the printer will complete the job. The rooms of the building are represented by place graph nodes labelled with the control R. Similarly, the users, printers, central print spool and print jobs are represented by place graph nodes labelled with the controls U, P, S, and J respectively. All controls in this example have arity 1, meaning that every (non-root) node of the place graph has a single port. The port of a room (R) node is linked to ports of other R nodes that are connected to the room by a door. The port of a user (U) node is linked to an outer name representing the identity of the user. The port of a printer (P) node is linked to the port of the spool (S) node representing the printer spool the printer is connected to. Finally, the port of a job (J) node will be linked to the port of the user node to which the job belongs.



`Room[b].Printer[a] | Spool[a] | Room[b].User[User].Job[-];`

Fig. 1. A example bigraph expressed in the visual syntax of the Big Red tool

The bigraph in Fig. 1 thus represents a single print spool and two rooms connected by a door (represented by the yellow link between the ports of the room nodes). The left room contains a printer which is linked to the print spool, and the right room contains a user with the identity `User` given as an outer name. The user has a job which has not yet been linked to the user (as long as it has not left the user, its ownership is given implicitly by its location).

Big Red permits users to specify custom shapes for nodes associated with each type of control. In this example, rooms are represented as boxes, users as triangles, jobs as circles, etc. The shaded box `User` is an outer name that is linked to a port of the `U` node. We reiterate Milner’s bigraph definition [5] here:

$$(V, E, ctrl, prnt, link) : \langle n, X \rangle \rightarrow \langle m, Y \rangle$$

where V is the set of nodes, E is a set of edges, $ctrl : V \rightarrow \Sigma$ assigns controls to nodes drawn from a signature Σ , $prnt : m \uplus V \rightarrow V \uplus m$ is the *parent map* that gives the nested place graph structure, and $link : X \uplus P \rightarrow E \uplus Y$ is the link map, where $P = \{(v, i) : v \in V \wedge i \in 0 \dots ar(ctrl(v))\}$ is the set of all ports.

Dynamic behaviour is added to a bigraph model by adding a set of *reaction rules*. The *JobToSpool* rule shown in Fig. 2 allows a print job to be transferred from a user to a spool, adding an identifying link to connect users to their submitted print jobs. The *JobToPrinter* rule in Fig. 3 will transfer a job from the spool to a printer that is co-located with the user associated with that job. The full example contains also rules for removing completed jobs and allowing users to move between rooms and is available from <http://bigraph.org/papers/gcm2012>.

Note that the rules are *parametric* in the sense that both the user and the spool node in Fig. 2 contains a *hole*, which Big Red represents as shaded place graph leaf nodes, indexed by integers. This means that both the user and spool node may contain other nodes (e.g. other job nodes) which in this example are left un touched by the rule. Interested readers are referred to [5] for detailed description of bigraphs and bigraphical reactive systems.

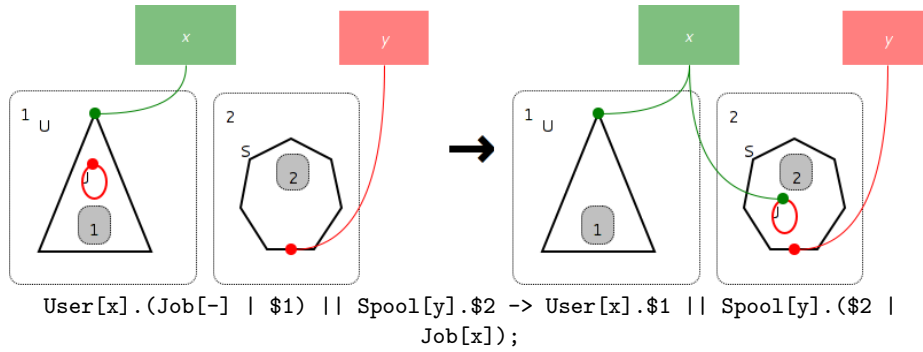


Fig. 2. The *JobToSpool* rule

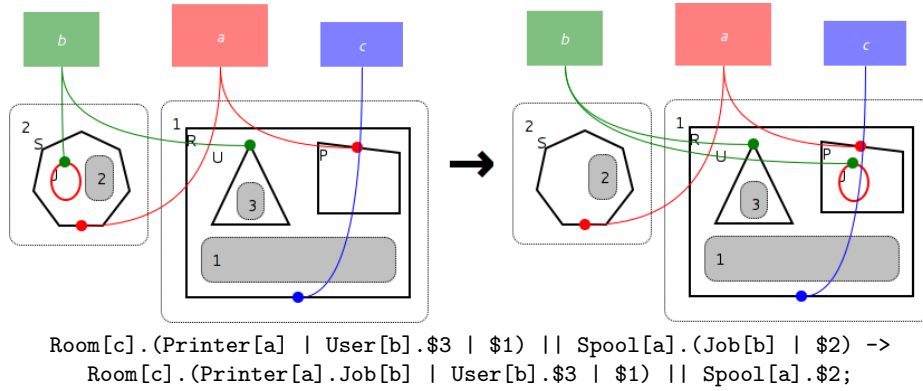


Fig. 3. The *JobToPrinter* rule

4 Implementation

Big Red is implemented as an Eclipse *plugin*, which extends the Eclipse platform with additional file formats representing the objects of a bigraphical reactive system, wizards to create model files, and editors to modify them. In turn, Big Red defines several Eclipse *extension points*: these allow other plugins to contribute extensions to Big Red, adding support for new external tools and export formats.

Big Red’s implementation of the bigraphical model strikes a balance between theoretical purity and practicality. No extensions to the bigraphical model are implemented, but the classes have been designed with extensibility in mind; as a result, adding new concepts to the model is easy. Indeed, Big Red uses these mechanisms to implement some of its own functionality — the information used to draw and position objects, for example, has no specific support in the underlying model.

The Eclipse platform includes comprehensive libraries to support the building of modelling tools. At the heart of these is the Graphical Editor Framework, a

toolkit for implementing model-view-controller-based editors. Big Red’s bigraph and reaction rule editors are both built on this powerful and flexible component.

Quite apart from its specific support for modelling tools, Eclipse is a very portable and widely-used platform with a vibrant community and many pre-built features, which makes it an ideal choice for the rapid development of an editor.

4.1 Interacting with External Tools

Big Red defines a special extension point for plugins that want to operate on a complete bigraphical reactive system. When an extension registered with this point is activated, Big Red’s user interface is suspended, and the extension takes over — essentially, this gives developers the ability to write subprograms that work with Big Red’s model objects without having to delve too deeply into the workings of Eclipse.

The integration between Big Red and BigMC [6] is implemented in this way — as an external plugin which converts a Big Red model into its BigMC term language representation, executes BigMC as a subprocess, and parses the results back into Big Red model objects so that they can be visualised.

5 Conclusion

We have presented a brief introduction to the Big Red tool, and described the motivation for developing such a tool. Big Red and accompanying user documentation are available under the Eclipse Public License from <http://bigraph.org>.

Big Red is still under active development. We intend to integrate support for other bigraph tools, and — together with the University of Udine — we are working on developing Big Red into a generally-useful platform for building and hosting new tools for bigraphs.

References

1. L. Birkedal, S. Debois, E. Elsborg, T.T. Hildebrandt, and H. Niss. Bigraphical models of context-aware systems. In *Foundations of Software Science and Computation Structures*, pages 187–201. Springer, 2006.
2. A.J. Glenstrup, T.C. Damgaard, L. Birkedal, and E. Højsgaard. An implementation of bigraph matching. 2007.
3. C. Greenhalgh. *bigraphspace*, 2009.
4. T.T. Hildebrandt, H. Niss, and M. Olsen. Formalising Business Process Execution with Bigraphs and Reactive XML. In *COORDINATION’06*, volume 4038 of *Lecture Notes in Computer Science*, pages 113–129. Springer-Verlag, January 2006.
5. R. Milner. *The Space and Motion of Communicating Agents*. Cambridge University Press, 2009.
6. G. Perrone, S. Debois, and T.T. Hildebrandt. A Model Checker for Bigraphs. In *ACM Symposium on Applied Computing 2012 – Software Verification and Testing Track*. ACM, 2012.