Towards User-Friendly Bigraphs

Paulius Dilkas

3rd July 2018

Abstract

We adapt an OCaml Jupyter kernel to support BigraphER.

1 Introduction

1.1 Jupyter

Project Jupyter [2], notebook [1], kernel, OCaml kernel[1], cell, magic.

1.2 Bigraphs and BigraphER

Bigraphs, BigraphER[2] [3]

begin-end block, stochastic and non-stochastic reaction rules

Goals: convenience of use, supporting a similar workflow to how Jupyter works with Python, not surprising the user with unexpected behaviour.

2 Assumptions

- Keywords `big`, `react`, `begin` start the beginning of line.
- Single spaces or no spaces.

3 Features

- Variables defined in one cell persist to the next (unless the cell contains a begin-end block or fails to run). This is done in the same order as the cells are run, including running the same cell multiple times.
- The output of each cell corresponds to everything defined in that cell (bigraphs and reaction rules): name first, then diagrams.

[1] `https://github.com/akabe/ocaml-jupyter`
[2] `http://www.dcs.gla.ac.uk/~michele/bigrapher.html`

- Reaction rules are visualised in an HTML table, connecting the diagrams side-by-side.
- All images are saved in a directory `jupyter-images`, with a subdirectory for each cell. If a subdirectory doesn't exist, it is created. Otherwise, its contents are cleared before the new images are added. Stale directories are not deleted.
- Both stochastic and non-stochastic rules are supported, as long as they are not in the same cell.
- OCaml code can be run if the first line is `%ocaml`.
- BigraphER API can be called from an OCaml cell. That requires two lines in `.ocamlinit` to load stuff:

```
#use "topfind";;
#require "bigraph";;
```

- Auto-complete and integrated documentation for BigraphER OCaml API (and other OCaml code) using Merlin.
- `%clear` magic to flush the buffer before evaluating the current cell. The motivation for this magic comes from the fact that multiple control definitions (`ctrl`) cause errors rather than warnings.

4 Implementation

- Separate buffers for OCaml and BigraphER code.
- Tests with good code coverage.
- BigraphER's version along with OCaml version in the kernel's name.
- Dummies are implemented with random words that are not defined anywhere else.
- Code from buffer is written to a file, BigraphER is run to generate images, file is deleted. BigraphER's output is captured, but not used, since combining multiple cells results in warnings about multiple definitions.
- Directory permissions are 700.
- `react` keyword is ignored if it's not at the start of a line.

When evaluating a cell, we look for a begin-end block in a top-down fashion (but ignoring the buffer).

- In case the model turns out to be non-stochastic (starts with `begin brs`), no additional work needs to be done.

- If we find a stochastic (`sbrs`) model, we must remove all non-stochastic reaction rules from both the buffer and the cell.
- If such a block is not found, a dummy `brs` block is generated for the purpose of running BigraphER and generating the required images. The dummy `begin-end` block has the form:

```
ctrl C = 0;
big b = C.1;
react r = b --> b;
begin brs
  init b;
  rules = [{r}];
  preds = {b};
end
```

  `C`, `b`, and `r` are randomly generated words, distinct from each other and other variables defined in the cell or buffer. The words are generated by adding random letters until the constructed string becomes unique. For simplicity of implementation, `C` always starts with a capital letter C.

5 Conclusion

References

[1] Antonino Ingargiola and contributors. Jupyter/IPython Notebook quick start guide. `https://jupyter-notebook-beginner-guide.readthedocs.io/en/latest/index.html`, 2015.

[2] Project Jupyter. About Project Jupyter. `https://jupyter.org/about`, June 2018.

[3] Michele Sevegnani and Muffy Calder. *BigraphER: Rewriting and Analysis Engine for Bigraphs*, pages 494–501. Springer International Publishing, Cham, 2016.