AutoMoDe Editor: A visualization tool for AutoMoDe

Jonas Kuckling, Ken Hasselmann,
Vincent van Pelt, Cédric Kiere, and Mauro Birattari

IRIDIA, Université libre de Bruxelles, Belgium.

July 2021

1 Introduction

AutoMoDe is a family of automatic modular design methods [1]. To the methods of the AutoMoDe family belong AutoMoDe-`Vanilla` [3], AutoMoDe-`Chocolate` [2], AutoMoDe-`Gianduja` [5], AutoMoDe-`Waffle` [11], AutoMoDe-`Maple` [9], AutoMoDe-`TuttiFrutti` [4], AutoMoDe-`IcePop` [6], AutoMoDe-`Coconut` [12], AutoMoDe-`Arlequin` [8], AutoMoDe-`Cedrata` [7] and AutoMoDe-`Phormica` [10]. These design methods can design control software for different software architectures, such as finite-state machines or behavior trees. They use textual representations of the control software, which are not intuitively understandable. The AutoMoDe Editor aims to provide a tool that allows easy visualization and manipulation of finite-state machines and behavior trees for the design methods in the AutoMoDe family.

2 Installation

This project requires you to have a working installation of AutoMoDe[1] and the node package manager (npm)[2], in order to install its other dependencies. To install the AutoMoDe Editor, follow these steps: Download the project from GitHub[3]. Inside of the project repository, run the following command in the terminal:

```
$ npm install
```

This will install all other dependencies and requirements for the project. Inside of the project repository, also create a file `.env`. Inside of this file put the following content:

```
AUTOMODE_PATH=<path_to_automode_executable>
EXPERIMENT_PATH=<path_to_argos_experiment_file>
```

[1] https://github.com/demiurge-project/ARGoS3-AutoMoDe
[2] https://github.com/npm/cli
[3] https://github.com/demiurge-project/AutoMoDe-Editor

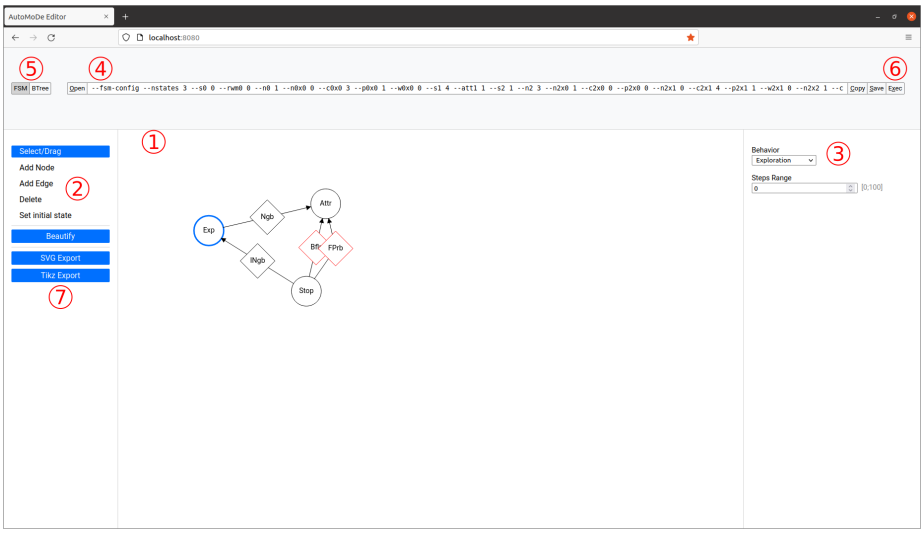

Figure 1: A screenshot of the AutoMoDe Editor, opened in Firefox. Points of interest are highlighted with red circled numbers.

3 Using the editor

In order to start the web editor, run the following command in a terminal inside of the project directory:

```
$ npm start
```

While keeping this terminal open, open `http://localhost:8080` in a browser. This should open a page like the one displayed in Figure 1. Note that Figure 1 shows an advanced state and the initial view will not contain a previous instance of control software.

3.1 Designing control software

The user can visually design control software in the visual editor (1). The editor allows to design both finite-state machines and behavior trees. The user can change the current architecture by selecting the corresponding button in the top left corner (5). By using the buttons on the left (2), the user can change the interaction mode within the editor, e.g., he can select "Select/Drag" to select and drag nodes in the editor. When an element is selected, detailed information for this element can be seen in the right panel (3). Here the user can change information, such as the behavior of a state or its parameters. The same instance of control software that is shown in the visual editor is also shown as a textual representation, that is understandable by the AutoMoDe executable (4). Changes to the visual representation will be immediately reflected in the textual representation. Changes in the textual representation will be reflected in the visual representation by pressing the Enter key. By pressing the button "Exec" (6), the user can execute the designed instance of control software, using the AutoMoDe executable and experiment file specified in the `.env` file. Additionally, the AutoMoDe Editor allows the export of the designed control

software, either as a SVG vector graphic file or as a Tikz description that can be embedded in LaTeX code (7).

3.2 Extending the editor

The AutoMoDe Editor currently includes the modules of AutoMoDe-`Chocolate` []. In order to include other modules (or change the currently available ones), the user can edit the files included in the folder `config/`. These files are JSON formatted and control the appearance and content of the right sidebar.

For finite-state machines, the `config/fsm/nodeCategories.json` contains the definition of the behavioral modules. Here the user can adjust the behaviors and their parameter spaces. The `config/fsm/edgeCategories.json` contains the definition of the condition modules. Here the user can adjust the conditions and their parameter spaces.

For behavior trees, the file `config/btree/nodeTypes.json` contains the definition of all node types in the behavior tree. Here the user can adjust the available control-flow nodes. The file `config/btree/nodeCategories.json` contains the definition of the modules. Here the user can adjust the available behaviors and conditions, as well as their parameter spaces.

4 The AutoMoDe Editor in the literature

The AutoMoDe Editor has been used as a tool in other studies. For example, Kuckling et al. [7] used the AutoMoDe Editor as a tool for human designers, to design behavior trees within the same restrictions as AutoMoDe-`Cedrata`.

Acknowledgments

The project presented in this report has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (DEMIURGE Project, grant agreement No 681872); from Belgium's Wallonia-Brussels Federation through the ARC Advanced Project GbO–Guaranteed by Optimization; and from the Belgian Fonds de la Recherche Scientifique–FNRS via the crédit d'équippement SwarmSim.

References

[1] Mauro Birattari, Antoine Ligot, and Gianpiero Francesca. AutoMoDe: a modular approach to the automatic off-line design and fine-tuning of control software for robot swarms. In Nelishia Pillay and Rong Qu, editors, *Automated Design of Machine Learning and Search Algorithms*. Springer, Cham, Switzerland, 2021. In press.

[2] Gianpiero Francesca, Manuele Brambilla, Arne Brutschy, Lorenzo Garattoni, Roman Miletitch, Gaëtan Podevijn, Andreagiovanni Reina, Touraj Soleymani, Mattia Salvaro, Carlo Pinciroli, Franco Mascia, Vito Trianni, and Mauro Birattari. AutoMoDe-Chocolate: automatic design of control software for robot swarms. *Swarm Intelligence*, 9(2–3):125–152, 2015.

[3] Gianpiero Francesca, Manuele Brambilla, Arne Brutschy, Vito Trianni, and Mauro Birattari. AutoMoDe: a novel approach to the automatic design of control software for robot swarms. *Swarm Intelligence*, 8(2):89–112, 2014.

[4] David Garzón Ramos and Mauro Birattari. Automatic design of collective behaviors for robots that can display and perceive colors. *Applied Sciences*, 10(13):4654, 2020.

[5] Ken Hasselmann and Mauro Birattari. Modular automatic design of collective behaviors for robots endowed with local communication capabilities. *PeerJ Computer Science*, 6:e291, 2020.

[6] Jonas Kuckling, Keneth Ubeda Arriaza, and Mauro Birattari. AutoMoDe-IcePop: automatic modular design of control software for robot swarms using simulated annealing. In Bart Bogaerts, Gianluca Bontempi, Pierre Geurts, Nick Harley, Bertrand Lebichot, Tom Lenaerts, and Gilles Louppe, editors, *Artificial Intelligence and Machine Learning: BNAIC 2019, BENELEARN 2019*, volume 1196 of *Communications in Computer and Information Science*, pages 3–17. Springer, Cham, Switzerland, 2020.

[7] Jonas Kuckling, Vincent van Pelt, and Mauro Birattari. Automatic modular design of behavior trees for robot swarms with communication capabilities. In Pedro A. Castillo and Juan Luis Jiménez Laredo, editors, *Applications of Evolutionary Computation: 24th International Conference, EvoApplications 2021*, volume 12694 of *Lecture Notes in Computer Science*, pages 130–145, Cham, Switzerland, 2021. Springer.

[8] Antoine Ligot, Ken Hasselmann, and Mauro Birattari. AutoMoDe-Arlequin: neural networks as behavioral modules for the automatic design of probabilistic finite state machines. In Marco Dorigo, Thomas Stützle, María J. Blesa, Christian Blum, , , and Volker Strobel, editors, *Swarm Intelligence: 12th International Conference, ANTS 2020*, volume 12421 of *Lecture Notes in Computer Science*, pages 109–122, Cham, Switzerland, 2020. Springer.

[9] Antoine Ligot, Jonas Kuckling, Darko Bozhinoski, and Mauro Birattari. Automatic modular design of robot swarms using behavior trees as a control architecture. *PeerJ Computer Science*, 6:e314, 2020.

[10] Muhammad Salman, David Garzón Ramos, Ken Hasselmann, and Mauro Birattari. Phormica: photochromic pheromone release and detection system for stigmergic coordination in robot swarms. *Frontiers in Robotics and AI*, 7:195, 2020.

[11] Muhammad Salman, Antoine Ligot, and Mauro Birattari. Concurrent design of control software and configuration of hardware for robot swarms under economic constraints. *PeerJ Computer Science*, 5:e221, 2019.

[12] Gaëtan Spaey, Miquel Kegeleirs, David Garzón Ramos, and Mauro Birattari. Evaluation of alternative exploration schemes in the automatic modular design of robot swarms. In Bart Bogaerts, Gianluca Bontempi, Pierre Geurts, Nick Harley, Bertrand Lebichot, Tom Lenaerts, and Gilles Louppe, editors, *Artificial Intelligence and Machine Learning: BNAIC*

2019, BENELEARN 2019, volume 1196 of *Communications in Computer and Information Science*, pages 18–33. Springer, Cham, Switzerland, 2020.