eBCSgen 2.0: Modelling and Analysis of Regulated Rule-based Systems*

Matej Troják ✉, David Šafránek, Branislav Brozmann, and Luboš Brim

Systems Biology Laboratory, Masaryk University, Brno, Czech Republic
trojak@mail.muni.cz

Abstract. eBCSgen is a software tool for developing and analysing models written in Biochemical Space Language (BCSL). BCSL is a rule-based language designed for the description of biological systems with rewriting rules in the form of behavioural patterns. This tool paper describes a new version of the tool, implementing the support for regulations, a mechanism suitable for reducing the branching behaviour of concurrent systems. Additionally, the presented version provides export to SBML, and support for CTL model checking. The paper artefact is available via https://doi.org/10.5281/zenodo.6644973.

1 Introduction

Rule-based modelling is a well-established approach in system biology. It makes a natural extension to the mechanistic reaction-based approach used in chemistry by introducing abstraction of detailed properties of molecules, forming the behavioural *patterns*. These are then used in rewriting *rules* that allow avoiding the combinatorial explosion that occurs when underlying molecules are specified explicitly in reactions, thus allowing to express complex models compactly. There are multiple existing rule-based modelling languages [4,7,9,12,13,15,16,22,24]. To improve the usage and the interoperability among individual languages, the rule-based approach is supported by SBML [11] in terms of the package multi [23].

While the mechanistic approach is beneficial for understanding the modelled system, some processes or properties are not easy to describe in such detail. They can often be expressed quantitatively, but such information is not always easy or even possible to determine. Still, they are crucial to capturing the behaviour of biological phenomena. In [20], we introduced *regulation* mechanisms in the context of multiset rewriting systems, and in [21], we formalised them as an extension of a rule-based languages representative Biochemical Space Language (BCSL) [22]. In general, regulations influence conditions when a rewriting rule can be applied. The regulations allow compactly modelling the additional knowledge about the system, otherwise too laborious or even impossible to express. We support five types of regulations – *regular* given by regular language over rules, *ordered* given by strict partial order on rules, *programmed* given by

* This work has been supported by the Czech Science Foundation grant 22-10845S.

successors for each rule, *conditional* given by prohibited context for each rule, and *concurrent-free* given by resolving concurrent behaviour of multiple rules.

In [18], we introduced eBCSgen, a software tool to provide a development environment and analyse models written in BCSL. The tool is integrated into Galaxy [2], a web-based platform for data-intensive biomedical research. It provides a convenient way to use eBCSgen due to the extensive popularity of Galaxy in the biology-oriented community. Among other features, the tool provides an interactive model editor for the development of models, simulations, and analysis of the model with respect to PCTL [10] properties.

In this paper, we present the second version of eBCSgen. Compared to the original version described in [18], the new version supports models with regulations, a non-probabilistic variant of CTL model checking, and model export to SBML [11] standard. The regulations are directly integrated into the core of the tool, which required redesigning the architecture. Consequently, it enabled to perform simulations and analyses even on models with a defined regulation. We employ an explicit CTL model checking [5,6] to support analysis of the branching behaviour on a qualitative level (in addition to quantitative PCTL analysis [19]), which can be directly influenced by the effects of regulations. To join the community effort on interoperability among other rule-based languages, we implemented support for model export to SBML [11] standard using the package multi [23].

eBCSgen is available as a standalone Python package [17] distributed using conda package manager [1] in the bioconda channel [8]. Moreover, it can be accessed online within our Galaxy instance[1]. It is accompanied with a tutorial[2] on how to use the tool in the Galaxy environment and a detailed explanation of regulations on a running example.

2 Regulated Biochemical Space Language

In this chapter, we briefly summarise the main features of BCSL using an illustrative yet simplistic example. Then we intuitively explain the effects of regulations in general and describe individual classes of regulation approaches introduced in [20] on multiset rewriting systems (MRS). The regulations are formally established within BCSL in a technical report in [21], where MRS are used as a low-level formal basis for BCSL. To grasp regulations in BCSL in more detail, we recommend studying the language itself [22] and then going through chapter four of tutorial[2], providing a comprehensive description of individual regulations on illustrative examples.

In Figure 1, there is an example of BCSL model. It consists of a definition of three rewriting *rules*, each tagged by a unique label (`r1_S`, `r1_T`, and `r2`). The rules describe possible modifications of a molecule `P`. This molecule has two domains, `S` and `T`, with an internal state (`i` for inactive and `a` for active). Additionally, the molecule has assigned a physical compartment, denoted by its

[1] `https://biodivine-vm.fi.muni.cz/galaxy/`

[2] `https://biodivine.fi.muni.cz/galaxy/eBCSgen/tutorial`

name after a double colon. The model is initialised, containing a single molecule of an inactive form of `P`.

```
#! rules
r1_S ~ P(S{i})::cell ⇒ P(S{a})::cell
r1_T ~ P(T{i})::cell ⇒ P(T{a})::cell
r2   ~ P()::cell     ⇒ P()::out

#! inits
1 P(S{i},T{i})::cell
```

Fig. 1. Example of BCSL model. It consists of a set of rules and an initial state, containing a single molecule of inactive form of `P`.

The rule `r1_S` can change the internal state of domain `S` from inactive to active. Similarly, the rule `r1_T` can do the same with the domain `T`. Please note such a modification can be done regardless of the internal state of the other domain. This is the main feature of the rule-based approach, which allows us to omit context which does not play any role in the modification. We often say that such a rule describes a *pattern* instead of a particular biochemical interaction. Finally, the rule `r2` can export the molecule `P` outside the `cell` by changing its compartment to value `out`.

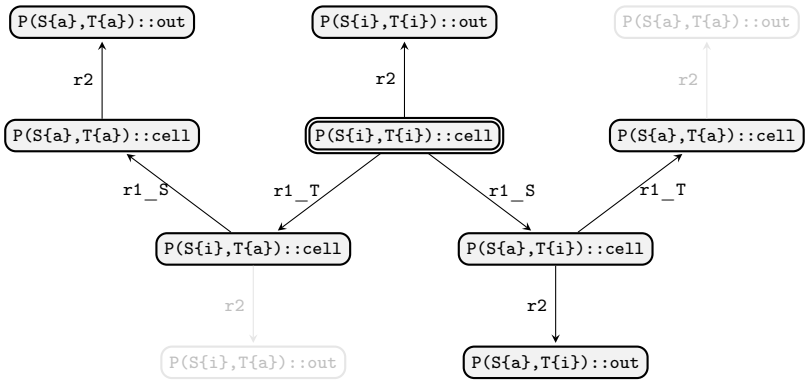

Fig. 2. Set of runs compactly represented by a transition system, starting in the double circled initial. In this case, the states always contain a single molecule `P` in a particular form. The label on an edge represents the rule responsible for a particular transition. The grey states and transitions are absent due to the effects of the regulation.

The semantics of BCSL model is based on *rewriting* process. A state, a (multi)set of molecules, can be rewritten by a rule to another state. There must be enough molecules present in the state that fit the pattern described by the rule. By chaining such rewriting, a *run* is formed. Transitive rewriting of the initial state by the rules forms a set of all possible runs, usually represented compactly by a transition system. The transition system of the model from Figure 1 is available in Figure 2 (including greyed-out states and transitions).

Regulated rewriting poses additional conditions on the rewriting process. A particular mechanism depends on an individual regulation approach, but it usually depends on the history of rewriting in the current (the system is not memoryless) or concurrent (the parallelism of biological systems) run. In general, the regulation restricts the set of runs, rendering some of the runs prohibited in the regulated model. In the following, we briefly present established regulation approaches.

Regular regulation is defined as a regular language over the rules. For practical reasons, it can be specified using a regular expression. Having a transition system, each run can be labelled by a sequence of used rules in the respective order. For the transition system of the regulated model then holds that every run must have its label from the defined regular language. This regulation is specifically useful in cases when we want to explicitly target particular runs that correspond to the desired behaviour of the model.

Ordered regulation is given by a partial order over the rules. It can be specified as a set of pairs. Focusing on each subsequent pair of rules in a run, it is required that the latter rule is greater than the former one. Every run in the transition system of the regulated model has to respect this order, otherwise it is eliminated. This regulation can be applied in cases when we want to ensure that some rules are not used immediately after each other in a sequence.

Programmed regulation specifies a set of successor rules for every rule. It is given by a function assigning an allowed set of successors to every rule. Each rule sequence can only be extended by the successor of the last rule. In terms of the regulated transition system, this eliminates runs that do not respect the successor function.

We provide an example to support this regulation. Let us assume the successor function for model from Figure 1 given by $\texttt{r1\_S} \to \{\texttt{r1\_T}, \texttt{r2}\}$ and $\texttt{r1\_T} \to \{\texttt{r1\_S}\}$. It allows specified successors for rules `r1_S` and `r1_T`, while the successors for rule `r2` are not given, which is considered as an empty set by default. The effect of the regulation is illustrated in Figure 2 by greyed-out states and transitions. In particular, the rule `r2` is not allowed to be used after the rule `r1_T`, resulting in the elimination of two states.

Conditional regulation is a natural extension to the rewriting mechanism where instead of requiring the presence of some molecules in the state, it requires their absence. The regulation is specified by a function assigning prohibited context to every rule. Every transition of every run in the transition system of the regulated model is enabled only if the prohibited context is not present in the corresponding state. Conditional regulation is beneficial when we need to

eliminate transitions under precise conditions while allowing the respective rule to be applied in all the other cases.

Concurrent-free regulation is targeting a natural phenomenon of concurrency in biological processes. It allows controlling concurrent processes by choosing the prioritised ones. The regulation is defined by an enumeration of pairs of rules, where the first member of the pair has priority over the second one. In the regulated transition system, branches of runs with a non-prioritised rule while a prioritised one is also enabled are disposed of.

3 Implementation

eBCSgen is implemented as a command-line Python package distributed using conda package manager [1] in the bioconda channel [8]. To improve its usability outside of computer science, it is wrapped into a series of tools for Galaxy [2], a web-based scientific analysis platform with three primary features – accessibility, reproducibility, and communication. It allows connecting individual tools into comprehensive pipelines and distribution of results among other scientists by sharing computation histories and datasets.

The previous version of the tool already supports a variety of analysis techniques and features. The Galaxy interface of eBCSgen offers interactive *model editor*, which can be used to create and edit BCSL models with automatic syntax highlighting and real-time code validation. Models with quantitative attributes (rule rates) exhibiting probabilistic behaviour can be analysed with respect to PCTL [10] properties. Such models can also be simulated, either using a *stochastic* (modified version of the standard Gillespie algorithm) or deterministic (inferring ODEs from generated reaction network) approach. Finally, there are several *static analysis* techniques to improve the scalability issues of exhaustive computational methods.

The major contribution to the new version of eBCSgen is the support of regulated models. This novelty influenced the tool on several levels. We extended the syntax of the BCSL and introduced these changes to the interactive editor. The presence of regulation in a model directly impacts the rewriting process of individual rules. This required changing the core of eBCSgen to support checking these extra conditions. Since the regulations typically require some sense of the history of the run, this also had to be included in the rewriting mechanism. To be more precise, individual regulations differ in their requirements on the depth of the history. Conditional and concurrent-free regulations do not need history. They can be validated directly based on local information (by inspecting the contents of the current state and checking other enabled rules, respectively). Ordered and programmed regulations require one-step history – to determine whether a rule can be used, information about the previously used rule is needed. Finally, regular regulation needs full history of used rules to determine whether the run belongs to the specified regular language.

Besides regulations, we also implemented support for CTL model checking. The main motivation behind this step was to support CTL analysis of qualitative

models and to be able to analyse the behaviour of such models with regulations. Indeed, the inherent feature of regulations is to reduce the branching of the transition system, making CTL model checking an ideal tool for validation of its effects. We implemented this feature using a python package `pyModelChecking` [5] and created a Galaxy wrapper.

Finally, to join the community effort on interoperability among other rule-based languages, we implemented support for model export to SBML standard using the package `multi`. The package extends the SBML Level 3 core with the *type* concept, and therefore reaction rules may contain species that can be patterns and be in multiple locations in reaction rules. It allows the SBML standard for encoding rule-based models using their native concepts for describing reactions instead of having to apply the rules and unfold the networks prior to encoding in an SBML format. This allows us to save BCSL models in a standard format, which enables their analysis beyond the scope of eBCSgen. We implemented the export using a package `libSBML` [3].

4 Evaluation

We demonstrated regulations on several models from biological domain[3]. In this section, we show usage of regulations on a well-studied fragment of `MAPK/ERK` signalling pathway [14], responsible for signal transduction from a receptor on the surface of the cell to the DNA in the nucleus of the cell. `MAPK` cascade contains three interconnected cycles of `MAPK`, `MAPK` kinase (`MAPKK`), and `MAPKK` kinase (`MAPKKK`). To capture the inhibitory effect of active `MAPKKK` on the activity of `MAPK`, we have applied conditional regulation to the model. It ensures that rule `r3k_fw` (responsible for the activation of `MAPKKK`) is not enabled when the active form of `MAPK` is present.

This regulation, in consequence, causes that when the deactivated form of `MAPKKK` and the activated form of `MAPK` coexist, then the activation of `MAPKKK` is not possible. To enable it again, `MAPK` first needs to be deactivated. In terms of biological effects, the active form of `MAPK` inhibits the activation of `MAPKKK`.

$$\texttt{EF}\ [\ \texttt{MAPK(R1\{a\},R2\{a\})::cyt} > 0\ \wedge\ \texttt{MAPKKK(R\{i\})::cyt} > 0\ \wedge\ \texttt{EX}\ [\ \texttt{P(S\{a\},T\{a\})::out} > 0\ ]\ ] \quad (1)$$

To support this claim about the behaviour of the regulated model, we perform CTL model checking. The property formulated above can be expressed as a CTL formula in Equation 1. The analysis concludes that while this formula is indeed `true` in the unregulated system, its truth value changes to `false` in the regulated system, confirming the desired behaviour effects were achieved. CTL analysis of both unregulated and regulated models is available as a Galaxy history[4].

[3] https://biodivine.fi.muni.cz/galaxy/eBCSgen/case-studies/cmsb-2022-regulations

[4] https://biodivine.fi.muni.cz/galaxy/eBCSgen/case-studies/cmsb-2022

5 Conclusion

We presented the tool eBCSgen with a focus on new features introduced in version `2.0`. The new version supports the development and analysis of models written in BCSL with regulations. These mechanisms can be used to ensure the correct sequence of execution of the individual processes and their mutual effects. We showed how the regulations could be used in a short case study and demonstrated their effects using CTL model checking, another newly introduced feature to eBCSgen. Finally, the tool supports SBML export as a key step toward standardisation of the rule-based representation.

References

1. Anaconda software distribution (2020), `https://docs.anaconda.com/`
2. Afgan, E., Baker, D., Batut, B., van den Beek, M., Bouvier, D., Čech, M., Chilton, J., Clements, D., Coraor, N., Grüning, B.A., Guerler, A., Hillman-Jackson, J., Hiltemann, S., Jalili, V., Rasche, H., Soranzo, N., Goecks, J., Taylor, J., Nekrutenko, A., Blankenberg, D.: The Galaxy Platform for Accessible, Reproducible and Collaborative Biomedical Analyses: 2018 Update. Nucleic Acids Res. **46**(W1), 537–544 (2018)
3. Bornstein, B.J., Keating, S.M., Jouraku, A., Hucka, M.: libSBML: An API Library for SBML. Bioinformatics **24**(6), 880–881 (2008)
4. Calzone, L., Fages, F., Soliman, S.: BIOCHAM: an Environment for Modeling Biological Systems and Formalizing Experimental Knowledge. Bioinformatics **22**(14), 1805–1807 (2006)
5. Casagrande, A.: pyModelChecking (2022), `https://pypi.org/project/pyModelChecking`
6. Clarke, E.M.: Model Checking. In: International Conference on Foundations of Software Technology and Theoretical Computer Science. pp. 54–56. Springer (1997)
7. Danos, V., Laneve, C.: Formal Molecular Biology. Theoretical Computer Science **325**, 69–110 (2004)
8. Grüning, B., Dale, R., Sjödin, A., Chapman, B.A., Rowe, J., Tomkins-Tinch, C.H., Valieris, R., Köster, J.: Bioconda: Sustainable and Comprehensive Software Distribution for the Life Sciences. Nature Methods **15**(7), 475–476 (2018)
9. Harris, L.A., Hogg, J.S., Tapia, J.J., Sekar, J.A., Gupta, S., Korsunsky, I., Arora, A., Barua, D., Sheehan, R.P., Faeder, J.R.: BioNetGen 2.2: Advances in Rule-based Modeling. Bioinformatics **32**, 3366 – 3368 (2016)
10. Hasson, H., Jonsson, B.: A Logic for Reasoning about Time and Probability. FAOC **6**, 512–535 (1994)
11. Hucka, M., Finney, A., Sauro, H.M., Bolouri, H., Doyle, J.C., Kitano, H., Arkin, A.P., Bornstein, B.J., Bray, D., Cornish-Bowden, A., et al.: The Systems Biology Markup Language (SBML): a Medium for Representation and Exchange of Biochemical Network Models. Bioinformatics **19**, 524–531 (2003)
12. Lopez, C.F., Muhlich, J.L., Bachman, J.A., Sorger, P.K.: Programming Biological Models in Python using PySB. Molecular Systems Biology **9** (2013)
13. Maus, C., Rybacki, S., Uhrmacher, A.M.: Rule-based Multi-level Modeling of Cell Biological Systems. BMC systems biology **5**(1), 1–20 (2011)
14. Pearson, G., Robinson, F., Beers Gibson, T., Xu, B.e., Karandikar, M., Berman, K., Cobb, M.H.: Mitogen-activated Protein (MAP) Kinase Pathways: Regulation and Physiological Functions. Endocrine reviews **22**(2), 153–183 (2001)

15. Pedersen, M., Phillips, A., Plotkin, G.D.: A High-Level Language for Rule-Based Modelling. Plos One **10**(6), 1–26 (06 2015)
16. Romers, J.C., Krantz, M.: rxncon 2.0: a Language for Executable Molecular Systems Biology. bioRxiv (2017)
17. Troják, M.: eBCSgen: A Bioconda package (2022), https://anaconda.org/bioconda/ebcsgen
18. Troják, M., Šafránek, D., Mertová, L., Brim, L.: eBCSgen: A Software Tool for Biochemical Space Language. In: Computational Methods in Systems Biology. pp. 356–361. Springer (2020)
19. Troják, M., Šafránek, D., Mertová, L., Brim, L.: Parameter Synthesis and Robustness Analysis of Rule-based Models. In: NASA Formal Methods Symposium. pp. 41–59. Springer (2020)
20. Troják, M., Pastva, S., Šafránek, D., Brim, L.: Regulated Multiset Rewriting Systems (2021), https://arxiv.org/abs/2111.13036
21. Troják, M., Šafránek, D., Brim, L.: Biochemical Space Language in Relation to Multiset Rewriting Systems (2022), https://arxiv.org/abs/2201.08817
22. Troják, M., Šafránek, D., Mertová, L., Brim, L.: Executable Biochemical Space for Specification and Analysis of Biochemical Systems. PLOS One **15**(9), 1–24 (2020)
23. Zhang, F., Meier-Schellersheim, M.: Multistate, Multicomponent and Multicompartment Species Package for SBML Level 3. COMBINE specifications (2017)
24. Zimmer, R.H., Millar, A.J., Plotkin, G.D., Zardilis, A.: Chromar, a Language of Parametrised Objects. Theoretical Computer Science (2017)