Experimenting with Reaction Systems using Graph Tansformation

Roberto Bruni[1] and Arend Rensink[2]

[1] University of Pisa, Italy, roberto.bruni@unipi.it
[2] University of Twente, Netherlands, arend.rensink@utwente.nl

We explore the capabilities of a state-of-the-art toolset based on graph transformation to perform reachability and causal analysis of Reaction Systems.

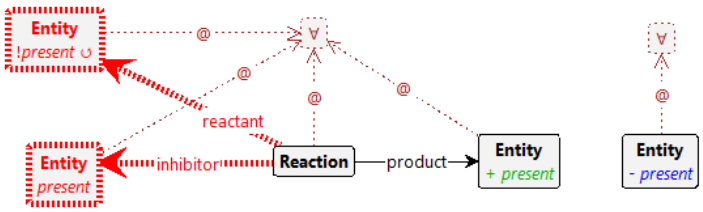
Reaction systems (RS) [2] are a computational model inspired by the functioning of biochemical reactions within living cells. RS focus on the interaction of entities through a set of reactions. Each reaction relies on some reactants, inhibitors, and products to mimic two fundamental mechanisms found in nature: facilitation and inhibition. At each time instant, the next state of the system is determined by the products of all enabled reactions plus some additional entities that are possibly provided by environment. Unlike traditional models of concurrency, like Petri nets, the theory of RS is based on three principles: *no permanency*, any entity vanishes unless it is sustained by a reaction; *no competition*, an entity is either available for all reactions, or it is not available at all; and *no counting*, the exact concentration level of available entities is ignored. Recent applications concerned, e.g., with the efficacy of medical treatments for comorbidities and with the selection of the best environment to achieve some desired phenomena, led to experiment with environments that exhibit nondeterministic and recursive behaviour. Then, performing reachability and causal analysis required the exploration of large state spaces for which the available prototype tool [1] struggled in terms of memory consumption and response time.

Graph Transformation (GT) [3,5] is a modelling technique that is widely applicable in problem domains where the objects of study have an inherent graphical structure, and the task at hand is to study their properties and evolution. Besides the graphs themselves, the core concept is that of a (transformation) *rule* capturing a particular change to such a graph. Rules can be used, for instance, to describe the change of a system over time, but can also be instrumental in composing and decomposing graphs and so exposing structural properties.

Importantly for the utility of the technique, there are a number of (academic) tools supporting its use. The research described here crucially relies on GROOVE [4], one of the most prominent tools in this area, which was designed precisely to enable GT-based system analysis of the kind described above. Two features of GROOVE that are essential for the purpose of this research are *(i)* nested (i.e., quantified) rules, which capture simultaneous changes in all neighbourhoods that satisfy certain application conditions, rather than only locally in one such neighbourhood at a time; and *(ii)* exploration of the set of reachable graphs (under the given rules) using various strategies.

We are in the process of investigating how GT can help in addressing the analysis questions in Reaction Systems. For this purpose, we encode a given

RS as a single graph, upon which a small number of (fixed) rules operate to simulate the correct semantics. The core rule describes the simultaneous firing of all **Reaction**s of which no inhibitors and all reactants are present; the firing results in the presence of all products. Simultaneously, all currently present **Entity**s are removed. In GROOVE syntax, this looks as follows:



To parse this, note that (in the GROOVE rule syntax) the red structure must be absent for the rule to apply; moreover, green labels are added and blue ones deleted upon rule application. The *present* flag signals whether an **Entity** is considered to be currently present; hence, creating or deleting that flag comes down to creating or deleting the **Entity**. The ∀-nodes impose the desired quantification, causing a single application of this rule to model the firing of all enabled **Reaction**s, even if there are thousands of them.

Our model also supports environments that inject **Entity**s in a controlled manner. This is achieved through a second rule, not shown here. A configuration is reachable if it can be constructed by the alternating application of both rules.

Our first results are encouraging: GROOVE seems capable to explore larger Reaction Systems by halving the analysis time of both reachability and causal analyses. More precisely, we are able to find a trace towards unwanted patterns (if they exist) among millions of reachable configurations using different heuristics; then, we can also prune the trace to extract a graphical representation of the causal history of that entities, at least as far as reactants are concerned.

References

1. Brodo, L., Bruni, R., Falaschi, M.: A logical and graphical framework for reaction systems. Theor. Comput. Sci. **875**, 1–27 (2021). https://doi.org/10.1016/J.TCS.2021.03.024
2. Ehrenfeucht, A., Rozenberg, G.: Reaction systems. Fundam. Informaticae **75**(1-4), 263–280 (2007), http://content.iospress.com/articles/fundamenta-informaticae/fi75-1-4-15
3. Ehrig, H., Ehrig, K., Prange, U., Taentzer, G.: Fundamentals of Algebraic Graph Transformation. Monographs in Theoretical Computer Science. An EATCS Series, Springer (2006). https://doi.org/10.1007/3-540-31188-2
4. Ghamarian, A.H., de Mol, M., Rensink, A., Zambon, E., Zimakova, M.: Modelling and analysis using GROOVE. Int. J. Softw. Tools Technol. Transf. **14**(1), 15–40 (2012). https://doi.org/10.1007/S10009-011-0186-X
5. Heckel, R., Taentzer, G.: Graph Transformation for Software Engineers - With Applications to Model-Based Development and Domain-Specific Language Engineering. Springer (2020). https://doi.org/10.1007/978-3-030-43916-3