# Locally Explainable Rules with Bigraphs

Dominik Grzelak
Chair of Software Technology
Technische Universität Dresden
Germany
Centre for Tactile Internet with Human-in-the-Loop (CeTI)
Technische Universität Dresden
Germany
dominik.grzelak@tu-dresden.de

Modeling large-scale reactive systems via graph transformation systems (GTS) [5] is often a complex task for beginners, particularly when model checking reveals correctness violations, where correctness properties are expressed via some formulae $S$ (i.e., a set of formal properties). Then, the model designer requires to fix the problem in the model $M$ until $M \models S$. In such cases, it is helpful to backtrack and provide counterexample traces for the model designer. However, before a user can benefit from generated counterexamples for debugging this first requires an understanding of the error cause and knowledge of the details of the graphs, rules and their disciplined interaction involved.

This can be difficult if there is a plethora of different rules, node and edge types. Moreover, visually interpreting graph-based state traces is difficult without appropriate tool support and may obscure in terms of pinpointing what went wrong. Additionally, reactive system models are heavily domain-specific since the static and dynamic semantics usually vary from application to application.

Therefore, we seek a universal method for reactive system models encoded as "(bi)graph programs" that generates human-friendly explanations for situations where rules cannot be applied. Specifically, we aim to understand why a rule fails at a given state and offer interpretable, natural language explications for the model designer. Additionally, even when graph programs are correct, providing live explications during runtime can elucidate why certain rules cannot be applied due to unmet preconditions. This could contribute to a better understanding of system behavior expressed by bigraph programs and may lower the barrier to entry for modeling reactive systems using bigraphical reactive systems (BRS) [8].

**Problem Definition** The problem of *locally explainable graph transformation rules* is defined here as follows.

- **Given:** A system modeled by GT that is not correct w. r. t. some correctness properties. For example, a rule R could not be applied at some state $x$.
- **Solution:** An answer for the specific failure. For example, an answer to "Why did rule R fail for state $x$?"; or, equivalently "Which properties were violated in state $x$ so that rule R could not be applied?"

A solution to this problem seeks to provide self-explanatory error causes for model designers in natural language. The approach shall work on any graph under the condition that only meaningful domain-specific labels are used for nodes and edges as well as for the rules.

## Explainable Rules

**Bigraph Programs** In the spirit of *graph programs* [9, 4], we define *bigraph programs* based on the theory of BRSs. This allows us to define a GTS comprising a set of reaction rules applied in a disciplined

Submitted to:
GCM 2024

manner to achieve high-level computation goals, by either following explicit control flows, or implicitly by the causal dependencies of the rules itself. Here, bigraph programs are executed by model checking.

**Graph Transformation Properties** In addition to the fact that graph transformation rules explicitly inform about their intent via their label and provide structural insights via the left-hand and right-hand side, we also leverage their intrinsic properties when describing the effect of a rule application (see [5, p. 311]) to *shape* their explanatory power:

- **Complete:** Any effect specified in the rule is performed in the concrete transformation;
- **Minimal** Nothing is done beyond what is specified in the rules;
- **Local:** The transformation only affects the fraction of the host graph that the match covers.

This has a direct impact on the "boundary" of what we can actually explain when a transformation is performed.

**Approach** To demonstrate the approach, we model a simple vending machine using BRSs by taking a process-based view. The system contains two interacting processes $\mathsf{S} = \mathsf{VM} \mid \mathsf{PHD}$, and its behavior is specified by the five reaction rules "Insert Coin" ($\mathsf{R}_1$), "Push Button 1" ($\mathsf{R}_2$), "Push Button 2" ($\mathsf{R}_3$), "Give Coffee" ($\mathsf{R}_4$), and "Give Tea" ($\mathsf{R}_5$). Consider now a state in this example, where the system cannot apply the "Insert Coin" rule. The procedure for producing local explanations for such cases is illustrated in Fig. 1. This approach relies on the compositional property of bigraphs when a reaction takes places. Given a bigraph $a$, and if it can be decomposed into $a \doteq C.R.d$, the next state can be obtained by rewriting it to $a' \doteq C.R'.d$. The first step *State Comparison* evaluates differences between each part of the decomposition. By that we find *conflicting differences* between the previous state $a$ and the current state $a'$ w. r. t. rule $\mathsf{R}$. A selection of these matches is taken into account in the next step to instantiate a prompt template for the LLM to complete. The response of this prompt contains generic information of the program error message ("Reaction rule [insertCoin] failed") and details of *conflicting differences*.[1] Then, we execute a second prompt asking the LLM to provide a description of the bigraph state $a'$ that refers to the "real-world". In both cases, graphs are always translated into a textual description first, which the LLM can process, and later interpreted with the help of the user-defined node and edge labels. Finally, combining both responses of the two prompts, the LLM is asked to derive what "this" suggests.

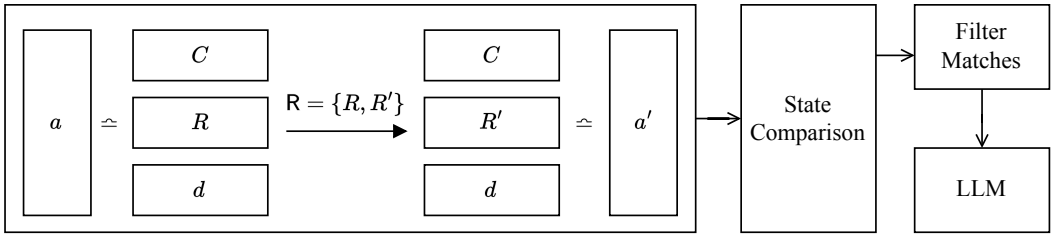


Figure 1: Schematic representation of the process for creating locally explainable rules with bigraphs.

[1] For example, a generated description for conflicting differences between the previous state $a$ and the current state $a'$ is "[Coin] is not contained in [Wallet]" and "[Coin] is contained in [VendingMachine]" for $a$ and $a'$, respectively.

**Response Categories** Following the approach in combination with Gpt 3.5, we achieved positive initial results. Without any fine-tuning, the LLM was more or less able to correctly interpret the system model. In this regard, we may distinguish different categories of explications w. r. t. our problem definition. For brevity, we present for each category an exemplary response without a formal definition:

- **Direct Explanation:** "This error message suggests that the program has attempted to apply a reaction rule that requires the presence of a coin, but none is found within the wallet. It is likely that the user has attempted to perform some action that requires coins to be present in their wallet, but none are found."
- **Thinking outside the box:** "This error message suggests that the user does not have the correct coins in their wallet to use in the vending machine. It is likely that the vending machine requires a certain type of coin, or a certain denomination of coin, that the user does not have."[2]
- **Solution-driven:** "This suggests that the user does not have enough coins stored in their wallet to perform the reaction rule 'insertCoin'. It is likely that the user needs to add more coins to their wallet in order to perform the desired action."

Especially, responses that fall into the last category allow to develop the notion of a "self-repairing GTS." An appropriate compensation rule could be triggered to repair the system (e.g., adding one coin to the wallet of the PhD student).

## Conclusion

Our preliminary result has shown that our naive approach can provide local explications for a simple vending machine system. To expand research, the following directions seem fruitful:

**Rule Types** Investigating the general interpretability of other rule types beyond the preliminary work, including stochastic/probabilistic [7, 2], conditional [1], prioritized, and nested rules.

**Connecting Local Explanations** Extending explications beyond local rule effects to explain more complex bigraph programs, such as the multi-agent path finding problem modeled in [6] using BRSs. Certainly, this involves integrating advanced notions of causality such as tracking rules, traceability links, and possibly quantum causal graph dynamics [3] for comprehensive reasoning. Furthermore, methods are required that connect *local explanations* into complete *cause-effect narratives* using backtracking and counter-examples. For example: "The PhD student inserted a coin and ordered coffee, but the second attempt failed since the machine had no drinks left, causing it to malfunction and retain the money."

**Acknowledgments** Funded by the German Research Foundation (DFG, Deutsche Forschungsgemeinschaft) as part of Germany's Excellence Strategy – EXC 2050/1 – Project ID 390696704 – Cluster of Excellence "Centre for Tactile Internet with Human-in-the-Loop" (CeTI) of Technische Universität Dresden.

[2] Our vending machine use case did not employ a coin currency but it might be worth to take that into consideration when designing a system.

# References


[1] Blair Archibald, Muffy Calder & Michele Sevegnani (2020): *Conditional Bigraphs*. In Fabio Gadducci & Timo Kehrer, editors: *Graph Transformation*, Springer International Publishing, pp. 3–19, doi:10.1007/978-3-030-51372-6_1.

[2] Blair Archibald, Muffy Calder, Michele Sevegnani & Mengwei Xu (2021): *Probabilistic BDI Agents: Actions, Plans, and Intentions*. In Radu Calinescu & Corina S. Păsăreanu, editors: *Software Engineering and Formal Methods*, Springer International Publishing, pp. 262–281, doi:10.1007/978-3-030-92124-8_15.

[3] Pablo Arrighi & Simon Martiel (2017): *Quantum Causal Graph Dynamics*. *Physical Review D* 96(2), p. 024026, doi:10.1103/PhysRevD.96.024026.

[4] Graham Campbell, Brian Courtehoute & Detlef Plump (2022): *Fast Rule-Based Graph Programs*. *Science of Computer Programming* 214, p. 102727, doi:10.1016/j.scico.2021.102727.

[5] Hartmut Ehrig, Karsten Ehrig, Ulrike Prange & Gabriele Taentzer (2006): *Fundamentals of Algebraic Graph Transformation*. Monographs in Theoretical Computer Science. An EATCS Series, Springer-Verlag, doi:10.1007/3-540-31188-2.

[6] Dominik Grzelak, Martin Lindner, Mikhail Belov, Oleksandr Husak, Uwe Aßmann & Hartmut Fricke (2024): *A Bigraphical Framework for Modeling and Simulation of UAV-based Inspection Scenarios*. Available at `https://nbn-resolving.org/urn:nbn:de:bsz:14-qucosa2-908655`. Revised Preprint.

[7] Reiko Heckel, Georgios Lajios & Sebastian Menge: *Stochastic Graph Transformation Systems*. In Hartmut Ehrig, Gregor Engels, Francesco Parisi-Presicce & Grzegorz Rozenberg, editors: *Graph Transformations*, Springer, pp. 210–225, doi:10.1007/978-3-540-30203-2_16.

[8] Robin Milner (2009): *The Space and Motion of Communicating Agents*, 1 edition. Cambridge University Press.

[9] Detlef Plump (2012): *The Design of GP 2*. *Electronic Proceedings in Theoretical Computer Science* 82, pp. 1–16, doi:10.4204/EPTCS.82.1.