A graphical language for rewriting-based programs and agent-based models

Kristopher Brown David I. Spivak
Topos Institute
Berkeley, USA
kris@topos.institute david@topos.institute

AlgebraicRewriting.jl is a computational category theory which focuses on implementing ideas in the graph rewriting literature [1]. Rewrite rules offer a combinatorial syntax to declaratively encode processes of data deletion, addition, merging, and copying. This provides an improvement over general purpose code for visualization and static analysis; however, some additional syntax is needed for applications that require application of rewrite rules in a controlled sequence. We have recently extended AlgebraicRewriting.jl with such a syntax, built on top of directed wiring diagrams with traces. The syntax is justified by a proposed trace structure on a dynamic Kleisli category, $\mathbf{DK}_t$, which is introduced in [2]. Let $\mathfrak{c}\langle - \rangle$ be the cofree comonoid construction on **Poly** [4], and let $\mathbf{Cat}^\sharp$ be the symmetric monoidal category of categories and cofunctors. Given a polynomial monad t, $\mathbf{DK}_t$ is defined as the following category enriched in $\mathbf{Cat}^\sharp$:

$$Ob(\mathbf{DK}_t) := \mathrm{Ob}(\mathbf{Set}) \qquad \mathrm{Hom}_{\mathbf{DK}_t}(A,B) := \mathfrak{c}\langle [Ay, t \triangleleft By] \rangle$$

We implement a DSL for graph transformation programs by taking certain morphisms in $\mathbf{DK}_t$ as primitive generators: construction of a composite program is performed by composing these primitive morphisms with the syntax of traced directed wiring diagram. The domain and codomain of our morphisms of interest consist of sets of diagrams of the form in Figure 1, and coproducts thereof. These are diagrams in a category of ACSets, i.e. attributed $\mathscr{C}$-Sets for some finitely-presented category $\mathscr{C}$, which are a category-theoretic model of databases which extends $\mathscr{C}$-Sets (i.e. copresheaves) to include noncombinatorial data [3]. Each such diagram, which we will call a *trajectory*, is a sequence of 'world states' X_i with distinguished focuses $A_i \to X_i$. A given 'focus' A can be thought of as the *shape* of a particular agent in the state of the world X, where the agent is picked out by the chosen morphism. This expressivity allows us to perform agent-based modeling (e.g. rewrite rules that express "add a loop to *this* vertex" rather than "add a loop *some* vertex"). Furthermore, the parameterization by a polynomial monad allows our implementation to naturally generalize to non-deterministic and probabilistic simulations.

We will introduce the generating morphisms of our DSL (for rewriting, pure control flow, and for changing agent focus) and then demonstrate how prototypical examples of agent-based modeling (e.g. NetLogo's wolf-sheep predation model, Kappa's molecular systems biological models) can be combinatorially expressed in this formalism. Furthermore advantages of this representation over code-based representations of agent-based models, such as functorial data migration and language-agnostic serialization, will be demonstrated. A toy example is shown in Figure 2.

Submitted to:
ACT 2023

$$\begin{tikzcd}
A_1 \arrow[d] & A_2 \arrow[d] & \cdots & A_n \arrow[d] \\
X_1 \arrow[r, "\shortmid" marking] & X_2 \arrow[r, "\shortmid" marking] & \cdots \arrow[r, "\shortmid" marking] & X_n
\end{tikzcd}$$

Figure 1: Reproduced from [2]: A trajectory in the space of ACSets. X_1 and X_n respectively represent the initial (resp. current) state of the world during the simulation, and each successive world state is related to the previous via a partial map (indicated by a ticked arrow). Each world state X_i also has a distinguished focus $A_i \to X_i$.

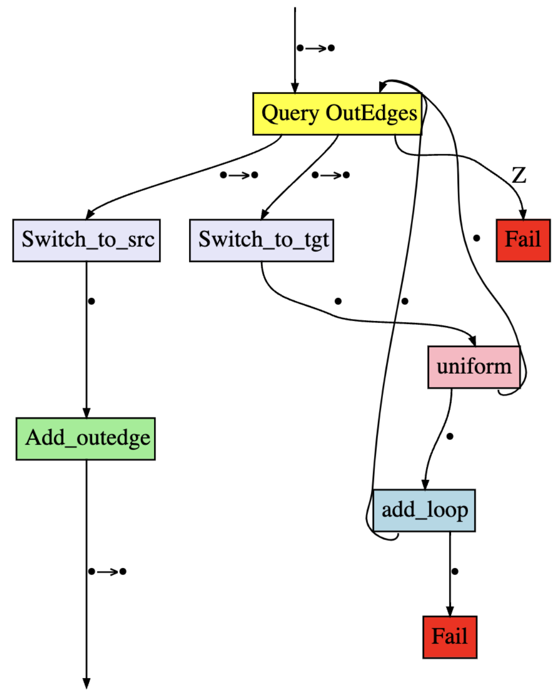

Figure 2: This toy simulation in the category of directed graphs showcases most of the primitive generators: Query (yellow), Rewrite (blue), Control Flow (light red), Agent Weakening (lavender), Agent Strengthening (green), and Fail (red). The required data and semantics for each box is described in Figures 8 and 9 of [2]. The current agent shape at each point in the control flow is known statically, visualized with a miniature graph. The overall simulation starts with a designated edge. It looks for all edges out of the target of the edge. For each of these, we focus on its target. We flip a coin and possibly add a loop to that vertex. After this is done, we focus on the source of our original edge. We simultaneously add a new vertex (and edge to that vertex) while making this new edge our focus as we exit the simulation.

References

[1] Kristopher Brown, Evan Patterson, Tyler Hanks & James Fairbanks (2022): *Computational category-theoretic rewriting*. In: *Graph Transformation: 15th International Conference, ICGT 2022, Held as Part of STAF 2022, Nantes, France, July 7–8, 2022, Proceedings*, Springer, pp. 155–172.

[2] Kristopher Brown & David I. Spivak (2023): *Dynamic Tracing: a graphical language for rewriting protocols*. arXiv:2304.14950.

[3] Evan Patterson, Owen Lynch & James Fairbanks (2022): *Categorical Data Structures for Technical Computing*. *Compositionality* 4, doi:10.32408/compositionality-4-5. Available at `https://doi.org/10.32408/compositionality-4-5`.

[4] David I. Spivak (2022): *A reference for categorical structures on* **Poly**, doi:10.48550/ARXIV.2202.00534. Available at `https://arxiv.org/abs/2202.00534`.