Operadic Modeling of Dynamical Systems: Mathematics and Computation

Sophie Libkind
Stanford University
Palo Alto, California, USA
slibkind@stanford.edu

Andrew Baas
Georgia Tech Research Institute
Atlanta, Georgia, USA
andrew.baas@gtri.gatech.edu

Evan Patterson
Topos Institute
Berkeley, California, USA
evan@topos.institute

James Fairbanks
University of Florida
Gainesville, Florida, USA
fairbanksj@ufl.edu

Dynamical systems are ubiquitous in science and engineering as models of phenomena that evolve over time. Although complex dynamical systems tend to have important modular structure, conventional modeling approaches suppress this structure. Building on recent work in applied category theory, we show how deterministic dynamical systems, discrete and continuous, can be composed in a hierarchical style. In mathematical terms, we reformulate some existing operads of wiring diagrams and introduce new ones, using the general formalism of C-sets (copresheaves). We then establish dynamical systems as algebras of these operads. In a computational vein, we show that Euler's method is functorial for undirected systems, extending a previous result for directed systems. All the ideas in this paper are implemented as practical software using Catlab and the AlgebraicJulia ecosystem, written in the Julia programming language for scientific computing.

1 Introduction

Category theory is about finding the right abstractions—identifying the salient, general features of the objects of study. In applied category theory the chosen objects of study lie outside of pure mathematics. One important thread of finding the right abstractions in the sciences has been understanding the composition of dynamical systems. Dynamical systems are a general and ubiquitous class of models which capture changing phenomenon. For example, automata model the changing of states in a computer, Petri nets model the changing concentrations of chemicals in a reaction network, and flows on a manifold model the evolution of physical systems. While scientists informally conceive of models compositionally, the modular structure is often lost in implementation. In this paper, we present a Julia package for dynamical systems that preserves the compositional structure as formalized by past research. If applied category theory is about *finding* the right abstractions for science, then the present work exemplifies *implementing* the right abstractions for science.

Existing work on composing dynamical systems varies along two axes. The first axis is semantic: *what* is a dynamical system? Dynamical systems are an extremely broad class of models, as reflected by previous work falling on many different points of the semantic axis. These points include circuit diagrams, Petri nets, Markov processes, finite state automata, ODEs, hybrid systems, and Lagrangian and Hamiltonian systems [5, 2, 1, 21, 11, 10, 3]. In this paper we focus on two kinds of dynamics: continuous flows and discrete transitions.

Submitted to:
Applied Category Theory 2021

The second axis is syntactic: *how* do dynamical systems compose? Two distinct styles of composition have emerged: directed and undirected, also called machine composition and resource sharing. In directed composition, information is transferred from designated senders to designated receivers. Systems are indexed by the behavior of other systems but otherwise have independent dynamics. In undirected composition, systems compose by sharing resources or observations. Composed systems affect each other only by acting on the shared medium. An important distinction is that undirected composition is not equivalent to symmetric directed composition. The directed and undirected perspectives are unified in [11].

1.1 Contributions

1. A practical implementation of operads and their algebras in the programming language Julia, which is widely used for scientific computing.
2. A reformulation of previously studied operads using C-sets, a diagrammatic approach to defining data structures. A new instance of this abstraction, the operad of circular port graphs, is also introduced.
3. Two new algebras for composing dynamical systems. The first algebra represents a directed composition of continuous and discrete systems that extends the syntax of algebras previously studied in [18, 15, 21] to include merging and creating wires. The second algebra represents the undirected composition of discrete dynamical systems.
4. A proof that Euler's method is functorial for undirected systems, plus an implementation in Julia of functorial Euler's method for both directed and undirected systems.

Acknowledgements The authors were supported by DARPA Awards W911NF2010292 and HR00112090067 along with AFOSR Award FA9550-17-1-e. The authors thank Micah Halter and Owen Lynch for support in developing Catlab and the AlgebraicDynamics software packages. They also thank David Spivak for his helpful insights into operads of wiring diagrams.

2 Preliminary Definitions

2.1 Operads and Operad Algebras

Operads and operad algebras formalize notions of syntax and semantics. In contrast, modeling tools generally obscure the distinction between syntax and semantics. These blurred lines make it challenging to interoperate between modeling frameworks and to independently adjust model syntax and model semantics. In this section we give the mathematical background for operad and operad algebras which will form the foundation of our software implementation.

Throughout we use *operads* to refer to symmetric colored operads or equivalently symmetric multicategories. We will also refer to the objects of an operad as its *types* and the morphisms of an operad as its *terms* in order to highlight the connection with syntax.

Definition 2.1. An *operad* $\mathcal{O}$ consists of a collection of types $\mathtt{ob}\,\mathcal{O}$ and for each $n \in \mathbb{N}^+$ and types $s_1, \ldots, s_n, t \in \mathtt{ob}\,\mathcal{O}$, a collection of terms $\mathcal{O}(s_1, \ldots, s_n; t)$, along with

- an identity term $1_t \in \mathcal{O}(t;t)$ for each type $t \in \mathtt{ob}\,\mathcal{O}$,
- substitution maps $\circ_i : \mathcal{O}(r_1, \ldots, r_m; s_i) \times \mathcal{O}(s_1, \ldots, s_n; t) \to \mathcal{O}(s_1, \ldots, s_{i-1}, r_1, \ldots, r_m, s_{i+1}, \ldots, s_n; t)$,

and permutation maps satisfying associativity, unitality, and symmetry laws.

An *operad functor* $F : \mathcal{O} \to \mathcal{O}'$ is a map on types and on terms that commutes with the identity, substitution, and permutation. Operads and their functors form a category Oprd.[1]

A rich source of operads and operad algebras are symmetric monoidal categories (SMCs) and lax monoidal functors. Specifically, a functor $\mathcal{O} : \mathsf{SMC} \to \mathsf{Oprd}$ sends each SMC $(\mathcal{C}, \otimes, 1)$ to its underlying operad $\mathcal{O}(\mathcal{C})$ with types $\mathsf{ob}\,\mathcal{C}$ and terms $\mathcal{O}(\mathcal{C})(s_1, \ldots, s_m; t) := \mathcal{C}(s_1 \otimes \cdots \otimes s_m, t)$. When clear from context, we denote $\mathcal{O}(\mathcal{C})$ simply by $\mathcal{C}$.

Definition 2.2. Given an operad $\mathcal{O}$, an *algebra of* $\mathcal{O}$ or simply an $\mathcal{O}$*-algebra* is an operad functor $F : \mathcal{O} \to \mathcal{O}(\mathbf{Set})$. We call a pair $(t \in \mathsf{ob}\,\mathcal{O}, m \in Ft) \in \int_{\mathsf{ob}\,\mathcal{O}} F$ an *element* of the algebra.

Symmetric monoidal categories ease the way for mathematical formalization and analysis. In this work, all of the operads and operad algebras for modeling dynamical systems are induced by symmetric monoidal categories and lax monoidal functors, respectively. However, the operadic perspective is better suited to computing because it directly supports n-ary operations rather than requiring that n-ary operations be decomposed into a tree of binary operations. The operadic viewpoint is highlighted in the Julia implementation (Section 4). Now we give two SMCs whose underlying operads define syntaxes for directed and undirected composition of dynamical systems, respectively.

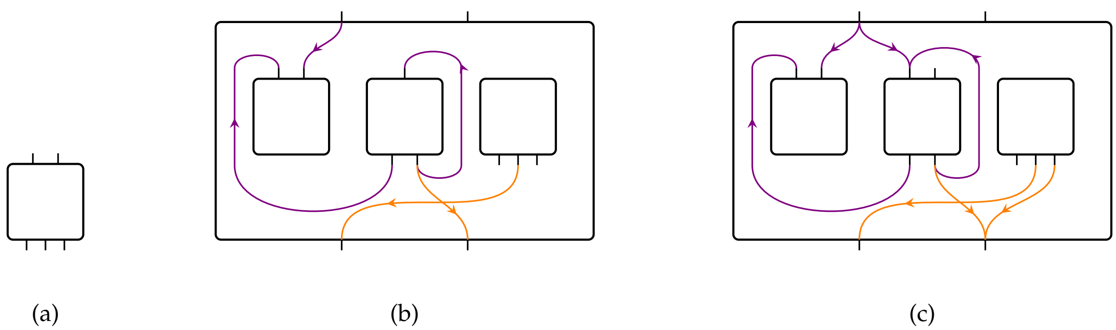


Figure 1: The graphical representation of DWD and its wide suboperad $\mathcal{O}(\mathsf{Lens}_{\mathbf{FinSet}^{\mathrm{op}}})$. (a) The type $\binom{2}{3}$. (b) A term $\binom{2}{0} + \binom{1}{2} + \binom{0}{3} \leftrightarrows \binom{2}{2}$ in $\mathcal{O}(\mathsf{Lens}_{\mathbf{FinSet}^{\mathrm{op}}})$. (c) A morphism $\binom{f^\#}{f} : \binom{2}{0} + \binom{2}{2} + \binom{0}{3} \leftrightarrows \binom{2}{2}$ in DWD. The orange and purple wires represent the apexes of f and $f^\#$ respectively. This syntactic diagram depicts the merging of wires (e.g., the first in-port of the second inner box) and the creation of wires (e.g., the second in-port of the second inner box).

Example 2.3 (Operad of directed wiring diagrams)**.** Given a category $\mathcal{C}$ with finite products, there is a lens category, $\mathsf{Lens}_{\mathcal{C}}$, whose objects are pairs $\binom{X_{\mathrm{in}}}{X_{\mathrm{out}}}$ where $X_{\mathrm{in}}, X_{\mathrm{out}} \in \mathsf{ob}\,\mathcal{C}$ and whose morphisms $\binom{f}{f^\#} \in \mathsf{Lens}_{\mathcal{C}}\left(\binom{X_{\mathrm{in}}}{X_{\mathrm{out}}}, \binom{Y_{\mathrm{in}}}{Y_{\mathrm{out}}}\right)$ are pair of morphisms $f : X_{\mathrm{out}} \to Y_{\mathrm{out}}$, $f^\# : Y_{\mathrm{in}} \times X_{\mathrm{out}} \to X_{\mathrm{in}}$ [19, Definition 2.2]. The cartesian monoidal structure on $\mathcal{C}$ induces a symmetric monoidal structure on $\mathsf{Lens}_{\mathcal{C}}$ [12]. Therefore $\mathsf{Lens}_{\mathcal{C}}$ has an underlying operad.

In [21, 15, 7] the directed syntax for composing dynamical systems is defined by the operad underlying $\left(\mathsf{Lens}_{\mathbf{FinSet}^{\mathrm{op}}}, +, \binom{0}{0}\right)$, often referred to as the operad of wiring diagrams. Following the Catlab implementation, we instead focus on operad underlying $\left(\mathsf{Lens}_{\mathsf{Cospan}(\mathbf{FinSet}^{\mathrm{op}})}, +, \binom{0}{0}\right)$. We define $\mathsf{DWD} := \mathcal{O}(\mathsf{Lens}_{\mathsf{Cospan}(\mathbf{FinSet}^{\mathrm{op}})})$ and call it the *operad of directed wiring diagrams*. In

[1]See [17] for a detailed exposition of operads and operad functors that aligns with their usage here. See also [9].

contrast to $\mathcal{O}(\mathsf{Lens}_{\mathbf{FinSet}^{op}})$, which allows only copying and deletion of wires, the syntax defined by DWD can also represent merging and creation of wires.

The graphical representation of DWD extends the standard graphical representation of $\mathcal{O}(\mathsf{Lens}_{\mathbf{FinSet}^{op}})$ (Figure 1). Types $\binom{X_{in}}{X_{out}}$ are represented as boxes with in-ports X_{in} and out-ports X_{out}. Let $\binom{f^\#}{f} : \binom{X_{in}}{X_{out}} \leftrightarrows \binom{Y_{in}}{Y_{out}}$ be a term in DWD. The morphism $f = X_{out} \leftarrow V \rightarrow Y_{out}$ represents a set V of wires with sources and targets given by the left and right legs of the span. Likewise, $f^\# : Y_{in} + X_{out} \leftarrow W \rightarrow X_{in}$ represents another set W of wires. The graphical representation emphasizes the operadic structure by having a separate box for each type in the term's domain.

Example 2.4 (Operad of undirected wiring diagrams)**.** We define UWD to be the operad underlying the symmetric monoidal category $(\mathsf{Cospan}(\mathbf{FinSet}), +, 0)$ and call it the *operad of undirected wiring diagrams*. Graphically, a type M is represented by a box with M exposed ports and a term $M \rightarrow J \leftarrow N$ is represented by J junction nodes with wires connecting ports M and N to junctions according to the legs of the cospan [17].

(a) (b)

Figure 2: The graphical representation UWD. (a) The type 6. (b) A term $2+3 \rightarrow 6 \leftarrow 5 \in \mathsf{UWD}(2,3;5)$.

2.2 $\mathcal{C}$-sets

$\mathcal{C}$-sets are a powerful abstraction for capturing data of a fixed shape [14, 16]. In this section, we define $\mathcal{C}$-sets and introduce specific $\mathcal{C}$-sets implementing operad terms.

Definition 2.5. Let $\mathcal{C}$ be a small category. An *$\mathcal{C}$-set* or *instance of $\mathcal{C}$* is a copresheaf over $\mathcal{C}$, equivalently a functor $X : \mathcal{C} \rightarrow \mathbf{Set}$. If X factors through $\mathbf{FinSet}$, then we say that X is *finite*.

Definition 2.6. Let $F : \mathcal{C} \rightarrow \mathcal{D}$ be a functor. Then there is a *pullback data migration functor* $\Delta_F : [\mathcal{D}, \mathbf{Set}] \rightarrow [\mathcal{C}, \mathbf{Set}]$ given by precomposition with F. The action of Δ_F on objects turns instances of $\mathcal{D}$ into instances of $\mathcal{C}$.

The category $\mathcal{C}$ is a schema that structures data, and an instance $X : \mathcal{C} \rightarrow \mathbf{Set}$ is an instance of the data structure. Next, we present schemata for undirected wiring diagrams, directed wiring diagrams, and circular port graphs. In our examples, the schemata are finitely presented categories and the finite instances of each schema comprise the terms of an operad. We can thus can take advantage of the rich mathematical structure of $\mathcal{C}$-sets, such as functorial data migration and the existence of finite limits and colimits, to build syntactic terms.

Example 2.7 (Theory of undirected wiring diagrams)**.** The schema for undirected wiring diagrams is $\mathsf{Th}(\mathsf{UWD})$, defined in Figure 3. An instance X of $\mathsf{Th}(\mathsf{UWD})$ consists of a set of boxes XB, ports XP, outer ports XQ, and junctions XJ. Each box $b \in XB$ has ports $X\mathsf{box}^{-1}(b) \subseteq XP$. Each

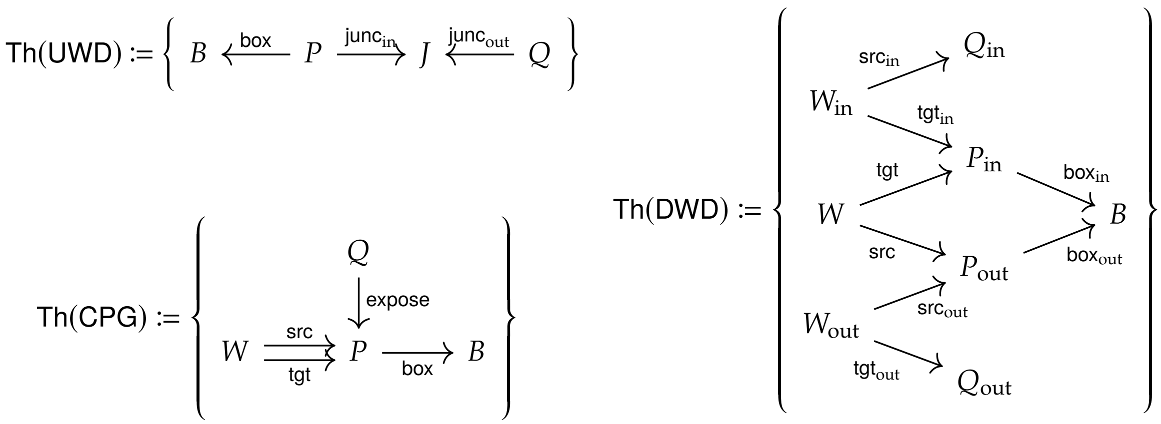


Figure 3: The schemata for the theories of undirected wiring diagram, directed wiring diagrams, and circular port graphs.

port $p \in XP$ connects to the junction $X\mathsf{junc}_{\mathrm{in}}(p)$, and likewise for outer ports. A finite instance X of $\mathsf{Th}(\mathsf{UWD})$ presents a term in $\mathcal{O}(\mathsf{Cospan}(\mathbf{FinSet}))$ that has domain types XB and defines the morphism $XP \xrightarrow{X\mathsf{junc}_{\mathrm{out}}} XJ \xleftarrow{X\mathsf{junc}_{\mathrm{out}}} XQ$ in $\mathsf{Cospan}(\mathbf{FinSet})$. Up to relabeling of the box elements and permutation of the domain types, finite elements of $\mathsf{Th}(\mathsf{UWD})$ correspond one-to-one with terms of UWD.

Example 2.8 (Theory of directed wiring diagrams)**.** The schema for directed wiring diagrams is $\mathsf{Th}(\mathsf{DWD})$, defined in Figure 3. An instance X of $\mathsf{Th}(\mathsf{DWD})$ consists of a set of boxes XB, sets of inner in-ports and out-ports XP_{in} and XP_{out}, sets of outer in-ports and out-ports XQ_{in} and XQ_{out}, and a set of wires $XW_{\mathrm{in}} + XW + XW_{\mathrm{out}}$. Each wire has source and target given by $X\mathsf{src}_{\mathrm{in}} + X\mathsf{src} + X\mathsf{src}_{\mathrm{out}}$ and $X\mathsf{tgt}_{\mathrm{in}} + X\mathsf{tgt} + X\mathsf{tgt}_{\mathrm{out}}$ respectively. A finite instance X of $\mathsf{Th}(\mathsf{DWD})$ presents a term in DWD that has domain types XB and defines the morphism

$$\left(\begin{array}{c} XQ_{\mathrm{in}} + XP_{\mathrm{out}} \xleftarrow{X\mathsf{src}_{\mathrm{in}} + X\mathsf{src}} XW_{\mathrm{in}} + XW \xrightarrow{(X\mathsf{tgt}_{\mathrm{in}}, X\mathsf{tgt})} XP_{\mathrm{in}} \\ XP_{\mathrm{out}} \xleftarrow{X\mathsf{src}_{\mathrm{out}}} XW_{\mathrm{out}} \xrightarrow{X\mathsf{tgt}_{\mathrm{out}}} XQ_{\mathrm{out}} \end{array}\right) : \begin{pmatrix} XP_{\mathrm{in}} \\ XP_{\mathrm{out}} \end{pmatrix} \leftrightarrows \begin{pmatrix} XQ_{\mathrm{in}} \\ XQ_{\mathrm{out}} \end{pmatrix}$$

in $\mathsf{Lens}_{\mathsf{Cospan}(\mathbf{FinSet}^{\mathrm{op}})}$. Up to relabeling of the box elements and permutation of the domain types, finite elements of $\mathsf{Th}(\mathsf{DWD})$ correspond one-to-one with terms of DWD.

Example 2.9 (Theory of circular port graphs)**.** The schema for circular port graphs is $\mathsf{Th}(\mathsf{CPG})$, defined in Figure 3. An instance X of $\mathsf{Th}(\mathsf{CPG})$ consists of a set of ports XP, outer ports XQ, and wires XW whose source and target are specified by $X\mathsf{src}$ and $X\mathsf{tgt}$ respectively.

Every circular port graph induces a directed wiring diagram by functorial data migration. Let the functor $F : \mathsf{Th}(\mathsf{CPG}) \to \mathsf{Th}(\mathsf{DWD})$ be defined on objects by $Q_{\mathrm{in}}, Q_{\mathrm{out}}, W_{\mathrm{in}}, W_{\mathrm{out}} \mapsto Q$, $P_{\mathrm{in}}, P_{\mathrm{out}} \mapsto P$, $W \mapsto W$, and $B \mapsto B$ and on morphisms by $\mathsf{src}_{\mathrm{in}}, \mathsf{tgt}_{\mathrm{out}} \mapsto \mathsf{id}_Q$, $\mathsf{tgt}_{\mathrm{in}}, \mathsf{src}_{\mathrm{out}} \mapsto$ $\mathsf{exposes}$, $\mathsf{src} \mapsto \mathsf{src}$, $\mathsf{tgt} \mapsto \mathsf{tgt}$, and $\mathsf{box}_{\mathrm{in}}, \mathsf{box}_{\mathrm{out}} \mapsto \mathsf{box}$. The pullback data migration functor $\Delta_F : [\mathsf{Th}(\mathsf{OpenCPG}), \mathbf{Set}] \to [\mathsf{Th}(\mathsf{DWD}), \mathbf{Set}]$ interprets circular port graphs as directed wiring diagrams by duplicating every port with one copy interpreted as an in-port and the other as an out-port. Example 2.8 gives an correspondence between finite instances of $\mathsf{Th}(\mathsf{DWD})$ and terms of DWD. Composing Δ_F with this correspondence defines a map from finite instances of $\mathsf{Th}(\mathsf{CPG})$ to terms of DWD.

Definition 2.10. The *operad of circular port graphs*, denoted CPG, is the suboperad of DWD whose types are pairs of the form $\binom{X_{\mathsf{port}}}{X_{\mathsf{port}}}$ and whose terms are generated by finite instances of $\mathsf{Th}(\mathsf{CPG})$.

Circular port graphs formalize the composition syntax used in stencil-based numerical algorithms. We prove that every term of CPG can be represented by a finite instance of $\mathsf{Th}(\mathsf{CPG})$ in Proposition A.3.

3 Algebras for Composing Dynamical Systems

Scientists often use diagrams informally to represent relationships between the components of a system. Over the last decade, applied category theorists have formalized these notions of compositional and hierarchical dynamical systems. The categorical frameworks offer a methodology for scientific modeling and their categorical structures can be implemented as modeling tools. Techiques for formalizing the composition of open dynamical systems often follow a general strategy. (1) An operad captures the syntax of interacting systems, and operadic substitution nests syntactic terms to give a more fine-grained description of the interactions. (2) An algebra over the operad assigns a concrete interpretation to the syntactic diagrams. To each type, the algebra gives a set of models of that type, and to each term, a function that defines how to compose the chosen models.

In this section, we define operads and operad algebras for the syntax and semantics of composing open dynamical systems. The algebras are denoted $\mathsf{Dynam}^{\mathrm{syn}}_{\mathrm{sem}}$ where sem $\in \{\mathsf{D}, \mathsf{C}\}$ indicates the model semantics (discrete or continuous) and syn $\in \{\rightarrow, \multimap\}$ indicates the composition syntax (directed or undirected). The algebras are all defined by lax monoidal functors, and we use the same notation for both the lax monoidal functor and the underlying algebra. For the proofs of propositions in this section, see Appendix A.2

3.1 Directed Composition

The framework for directed composition of dynamical systems relies heavily on the generalized lens construction defined in [19]. Although this theory is robustly developed in [7], here we present a variant that aligns with our Julia implementation. In particular, we restrict our attention to dynamical systems defined on Euclidean spaces.

In this section, we define algebras over DWD, which factor through the algebras DS and CS defined in [18]. We define the algebras in two steps. First, let Euc be the full subcategory of the category of smooth manifolds spanned by the Euclidean spaces. A strong monoidal functor $\mathbf{ev}_{\mathbb{R}} : \mathsf{Cospan}(\mathbf{FinSet}^{\mathrm{op}}) \to \mathsf{Euc}$ is defined on objects by $P \mapsto \mathbb{R}^P$ and on morphisms by $P \xleftarrow{f} W \xrightarrow{g} Q$ maps to $g_* \circ f^* : \mathbb{R}^P \to \mathbb{R}^Q$ (see Proposition A.5). By functoriality of the Lens construction, $\mathbf{ev}_{\mathbb{R}}$ induces a strong monoidal functor $\mathsf{Lens}_{\mathbf{ev}_{\mathbb{R}}} : \mathsf{Lens}_{\mathsf{Cospan}(\mathbf{FinSet}^{\mathrm{op}})} \to \mathsf{Lens}_{\mathsf{Euc}}$. Next, consider the lax monoidal functor $\int_{S:\mathbf{FinSet}} \mathsf{Lens}_{\mathsf{Euc}}\left(\binom{\mathbb{R}^S}{\mathbb{R}^S}, -\right) : \mathsf{Lens}_{\mathsf{Euc}} \to \mathbf{Set}$. Explicitly, this functor maps an object $\binom{\mathbb{R}^{X_{\mathrm{in}}}}{\mathbb{R}^{X_{\mathrm{out}}}}$ to the set of pairs $\left(S \in \mathbf{FinSet}, \binom{u}{r} : \binom{\mathbb{R}^S}{\mathbb{R}^S} \leftrightarrows \binom{\mathbb{R}^{X_{\mathrm{in}}}}{\mathbb{R}^{X_{\mathrm{out}}}}\right)$ and maps a morphism $\binom{f^\#}{f}$ to the set map sending $(S, \binom{u}{r})$ to $(S, \binom{f^\#}{f} \circ \binom{u}{r})$. Finally, consider the composite

$$\mathsf{Lens}_{\mathsf{Cospan}(\mathbf{FinSet}^{\mathrm{op}})} \xrightarrow{\mathsf{Lens}_{\mathbf{ev}_{\mathbb{R}}}} \mathsf{Lens}_{\mathsf{Euc}} \xrightarrow{\int_{S:\mathbf{FinSet}} \mathsf{Lens}_{\mathsf{Euc}}\left(\binom{\mathbb{R}^S}{\mathbb{R}^S}, -\right)} \mathbf{Set}.$$

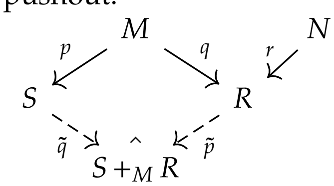
This DWD algebra maps an object $\binom{X_{\mathrm{in}}}{X_{\mathrm{out}}}$ to the set of pairs $\left(S \in \mathbf{FinSet}, \binom{u}{r} : \binom{\mathbb{R}^S}{\mathbb{R}^S} \leftrightarrows \binom{\mathbb{R}^{X_{\mathrm{in}}}}{\mathbb{R}^{X_{\mathrm{out}}}}\right)$. We interpret S as a set of state variables and $\mathbb{R}^S$ as the state space. Depending on whether we interpret $u : \mathbb{R}^{X_{\mathrm{in}}} \times \mathbb{R}^S \to \mathbb{R}^S$ as an indexed endomorphism of the state space or as an indexed vector field on the state space, the algebra represents either discrete dynamical systems or continuous dynamical systems. In other words, for an input $a \in \mathbb{R}^{X_{\mathrm{in}}}$ and state $x \in \mathbb{R}^S$, we can either think of $u(a,x) \in \mathbb{R}^S$ as the next state or as defining the vector $\dot{x} = u(a,x)$. We use the notation $\mathsf{Dynam}_{\mathsf{D}}^{\to}$ and $\mathsf{Dynam}_{\mathsf{C}}^{\to}$ respectively to highlight the distinct interpretations.[2]

3.2 Undirected Composition

Just as there is an algebra of continuous systems over the directed syntax DWD, there is an algebra over the undirected syntax UWD. We define the algebra $\mathsf{Dynam}_{\mathsf{C}}^{\multimap} : \mathsf{Cospan}(\mathbf{FinSet}) \to \mathbf{Set}$ which on objects takes M to the set of triples $(S \in \mathbf{FinSet}, v : \mathbb{R}^S \to \mathbb{R}^S, p : M \to S)$ and on morphisms maps the cospan $f = M \xrightarrow{q} R \xleftarrow{r} N$ to the set map $\mathsf{Dynam}_{\mathsf{C}}^{\multimap}(f) : \mathsf{Dynam}_{\mathsf{C}}^{\multimap}(M) \to \mathsf{Dynam}_{\mathsf{C}}^{\multimap}(N)$ given by

$$\mathsf{Dynam}_{\mathsf{C}}^{\multimap}(f)(S, v, p) = (S +_M R, \tilde{q}_* \circ v \circ \tilde{q}^*, \tilde{p} \circ r)$$

where $\tilde{q}$ and $\tilde{p}$ are defined by the pushout:



This composition of continuous systems is an operadic perspective of the hypergraph category Dynam presented in [2]. Next, we define undirected composition of discrete systems.

Proposition 3.1. There is an algebra $\mathsf{Dynam}_{\mathsf{D}}^{\multimap} : \mathsf{Cospan}(\mathbf{FinSet}) \to \mathbf{Set}$ which on objects maps M to the set of triples $(S \in \mathbf{FinSet}, u : \mathbb{R}^S \to \mathbb{R}^S, p : M \to S)$ and on morphisms maps the cospan $f = M \xrightarrow{q} R \xleftarrow{r} N$ to the set map $\mathsf{Dynam}_{\mathsf{D}}^{\multimap}(f) : \mathsf{Dynam}_{\mathsf{D}}^{\multimap}(M) \to \mathsf{Dynam}_{\mathsf{D}}^{\multimap}(N)$ defined by

$$\mathsf{Dynam}_{\mathsf{D}}^{\multimap}(f)(S, u, p) = (S +_M R, 1_{\mathbb{R}^{S+_M R}} + \tilde{q}_* \circ (u - 1_{\mathbb{R}^S}) \circ \tilde{q}^*, \tilde{p} \circ r).$$

3.3 Functorial Analysis

Compositional modeling paves the way for compositional analysis. Informally, an analysis of an operad algebra $F : \mathcal{O} \to \mathbf{Set}$ is an algebra $G : \mathcal{O} \to \mathbf{Set}$ and a natural transformation $\blacksquare : F \Rightarrow G$ which obscures the details of the system and highlights some feature of the behavior. Examples

[2]In the literature, the algebra of continuous dynamical systems explicitly represents a vector field as a section $u : \mathbb{R}^S \to T\mathbb{R}^S$ of the tangent bundle. However, we present $\mathsf{Dynam}_{\mathsf{C}}^{\to}$ and $\mathsf{Dynam}_{\mathsf{D}}^{\to}$ by the same algebra because they are implemented identically in AlgebraicDynamics.

Terminology	Mathematical abstraction	Julia implementation
diagram of systems	$\phi \in \mathcal{O}(s_1,\ldots,s_n;t)$ $\phi_{\text{inner}} \in \mathcal{O}(r_1,\ldots,r_m;s_i)$	`diagram::ACSet{TheoryO}` `inner_diagram::ACSet{TheoryO}`
elementary models	$(m_1,\ldots,m_n) \in Fs_1 \times \cdots \times Fs_n$	`models::Vector{T}`
composition of models	$F(\phi)(m_1,\ldots,m_n) \in Ft$	`oapply(diagram, models)`
hierarchical diagram	$\phi \circ_i \phi_{\text{inner}}$	`ocompose(diagram,i,inner_diagram)`

Table 1: Comparing mathematical abstractions with their Julia implementation. Note that `diagram` stores the number of domain types n and the types $s_1,\ldots,s_n,t$. Likewise `models` stores the types $s_1,\ldots,s_n$. The definitions of `ocompose` and `oapply` check that the arguments have appropriate types.

include identifying fixed points and orbits, solving trajectories, and computing approximations [2, 20, 4, 13]. The naturality of ■ implies that the behavior of the total system is defined by the behaviors of its components.

For both undirected and directed dynamical systems, there exists a natural transformation which performs Euler's method. For a map $v : \mathbb{R}^M \times \mathbb{R}^S \to \mathbb{R}^S$ and step size $h \in \mathbb{R}_+$, define the map $\mathsf{Euler}_h(v) : \mathbb{R}^M \times \mathbb{R}^S \to \mathbb{R}^S$ by $\mathsf{Euler}_h(v)(u_0, x_0) = x_0 + hv(u_0, x_0)$.

Proposition 3.2 (Euler's method for directed systems, [18])**.** For $h \in \mathbb{R}_+$, there exists a natural transformation $\mathsf{Euler}_h^{\to} : \mathsf{Dynam}_\mathsf{C}^{\to} \Rightarrow \mathsf{Dynam}_\mathsf{D}^{\to}$ with components $\mathsf{Euler}_h^{\to}\binom{X_{\text{in}}}{X_{\text{out}}} : \mathsf{Dynam}_\mathsf{C}^{\to}\binom{X_{\text{in}}}{X_{\text{out}}} \to \mathsf{Dynam}_\mathsf{D}^{\to}\binom{X_{\text{in}}}{X_{\text{out}}}$ defined by

$$\mathsf{Euler}_h^{\to}\begin{pmatrix} X_{\text{in}} \\ X_{\text{out}} \end{pmatrix}\left(S \in \mathbf{FinSet}^{\text{op}}, \begin{pmatrix} v \\ r \end{pmatrix} : \begin{pmatrix} \mathbb{R}^S \\ \mathbb{R}^S \end{pmatrix} \leftrightarrows \begin{pmatrix} \mathbb{R}^{X_{\text{in}}} \\ \mathbb{R}^{X_{\text{out}}} \end{pmatrix}\right) = \left(S, \begin{pmatrix} \mathsf{Euler}_h(v) \\ r \end{pmatrix}\right).$$

Proposition 3.3 (Euler's method for undirected systems)**.** For $h \in \mathbb{R}_+$, there exists a natural transformation $\mathsf{Euler}_h^{\multimap} : \mathsf{Dynam}_\mathsf{C}^{\multimap} \Rightarrow \mathsf{Dynam}_\mathsf{D}^{\multimap}$ with components $\mathsf{Euler}_h^{\multimap}(M) : \mathsf{Dynam}_\mathsf{C}^{\multimap}(M) \to \mathsf{Dynam}_\mathsf{D}^{\multimap}(M)$ defined by $\mathsf{Euler}_h^{\multimap}(M)(S, v, p) = (S, \mathsf{Euler}_h(v), p)$.

4 Julia Implementation

In Julia, the specification of an algebra $F : \mathcal{O} \to \mathbf{Set}$ consists of

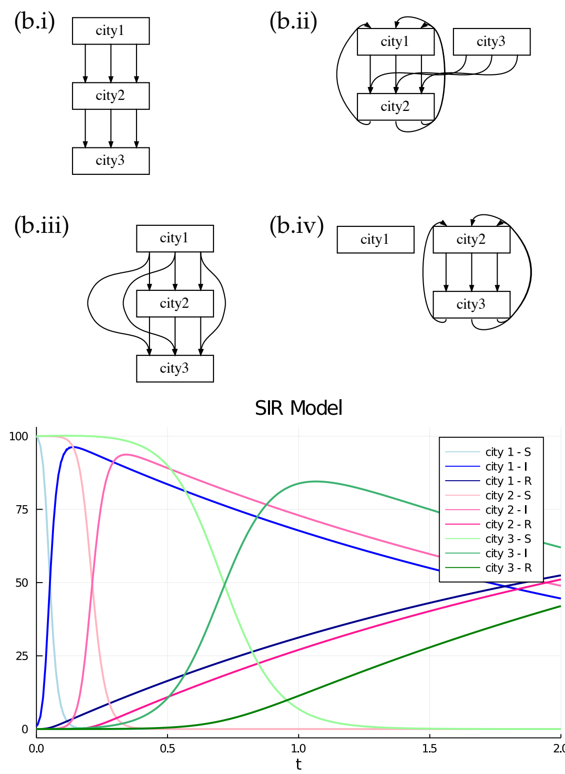
- a schema `TheoryO` such that finite instances of the schema represent terms of $\mathcal{O}$,
- a method `ocompose` implementing operadic substitution,
- a Julia type `T` such that values of type `T` implement algebra elements ($t \in \mathsf{ob}\,\mathcal{O}, m \in Ft$),
- and a method `oapply` implementing the action of F on terms.

We highlight the correspondence between the mathematical and modeling terminologies. Let $\phi : s_1,\ldots,s_n \to t$ be a term in $\mathcal{O}$. We say that ϕ represents a diagram or composite of subsystems. Let $m_i \in F(s_i)$ for $i = 1,\ldots,n$. The elements (s_i, m_i) of F are called models. We often speak of component or elementary models to emphasize their role in the expression $m = F(\phi)(m_1,\ldots,m_n) \in Ft$. Likewise, we say that m is the composite or total model. The correspondences between the modeling terminology, mathematical abstractions, and Julia code constructs are listed in Table 1.

The AlgebraicJulia ecosystem is a family of tools built on categorical techniques. Catlab.jl provides the schemas defined in Examples 2.7, 2.8, and 2.9, along with corresponding `ocompose` methods. AlgebraicDynamics.jl provides Julia types and `oapply` methods that implement the models and their composition for the Dynam algebras of Section 3. The following examples illustrate how applied category theory provides flexible, powerful abstractions for implementing scientific models.

```julia
ncities = 3
roads = [1 => 2, 2 => 3, 1 => 3]
nout_roads = map(i->count(r->r.first==i, roads), 1:ncities)

# Define the syntacitic diagram of systems
multicity_diagram = WiringDiagram([], [])
cities = map(1:ncities) do i
  add_box!(multicity_diagram,
          Box(Symbol(:city, i), [:S, :I, :R], [:S, :I, :R]))
end

wires = map(Base.Iterators.product(roads, 1:3)) do ((src, tgt), j)
  add_wire!(multicity_diagram, (cities[src], j) => (cities[tgt], j))
end

# Define the component models
city_models = map(1:ncities) do i
  ContinuousMachine{Float64}(3, 3, 3,
    (u,x,p,t) -> p.μ*(x - nout_roads[i]*u) + [
      -p[Symbol(:β,i)]*u[1]*u[2], # Ṡ
      p[Symbol(:β,i)]*u[1]*u[2] - p[Symbol(:γ,i)]*u[2], # İ
      p[Symbol(:γ,i)]*u[2] ],                          # Ṙ
    u -> u)
end

# Compose component models according to the diagram
sir_model = oapply(multicity_diagram, city_models)

# Solve the ODEs
params = LVector(μ = 0.01,
    β1 = 0.7, γ1 = 0.4,
    β2 = 0.4, γ2 = 0.4,
    β3 = 0.1, γ3 = 0.4
)

u0 = [100.0, 1.0, 0, 100.0, 0, 0, 100.0, 0, 0]
tspan = (0.0, 1.0)
prob = ODEProblem(sir_model, u0, tspan, params)
sol = solve(prob, Tsit5())
```

(a)



(c)

Figure 4: (a) Julia code for the multi-city SIR model using AlgebraicDynamics. (b) The graphical display of four possible terms for composition, of which (b.iii) is defined as `multicity_diagram` in the code in (a). (c) The solution to the multi-city SIR model produced by the code in (a).

Example 4.1. The SIR model is a classic model of the spread of an infectious disease [8]. A single-city SIR model has a susceptible population S, an infected population I, and a recovered population R, which evolve according to the continuous dynamics

$$\dot{S} = -\beta S I, \quad \dot{I} = \beta S I - \gamma I, \quad \dot{R} = \gamma I. \tag{1}$$

Such a model assumes a mixing of the population which fails to account for geographic or social distinctions between sub-populations. To model such distinctions, we compose multiple single-city SIR models using the operad algebra $\mathsf{Dynam}^{\rightarrow}_{\mathsf{C}} : \mathsf{DWD} \to \mathbf{Set}$ defined in Section 3.1.

Figure 4(a) shows the code for a multi-city SIR model using AlgebraicDynamics. The directed wiring diagram, `multicity_diagram`, is an instance of $\mathsf{Th}(\mathsf{DWD})$ and so corresponds to a term ϕ in DWD. Four different wiring diagrams for composing three single-city SIR models are shown in Figure 4(b), of which (b.iii) is used in the code. The Julia type `ContinuousMachine` implements elements of $\mathsf{Dynam}_{\mathsf{C}}^{\rightarrow}$, and the array `city_models` defines for each city a model with three states (corresponding to the local S, I, and R populations), the vector field from Equation 1 modulated by an inflow and outflow, and an identity readout. The call of the function `oapply` applies the set map $\mathsf{Dynam}_{\mathsf{C}}^{\rightarrow}(\phi)$ to the single-city elements defined by `city_models` and returns a multi-city model `sir_model`. Figure 4(c) shows a solution to the multi-city model.

Several aspects of the algebraic formalism translate to useful features of the software. First, the clear delineation between syntax and semantics enables the user to modify them independently. For example, the choice of the composition term `multicity_diagram` is independent of the choice of the single-city models `city_models`. Moreover, adding more components, such as additional cities, to an existing model is a straightforward application of compositionality. Without the algebraic abstraction, this procedure would involve many coordinated changes to the code. Finally, the syntactic diagram gives information about the behavior of the model. For example, the composition term in Figure 4(b.iv) implies that the behavior of city 1 is independent of cities 2 and 3 regardless of the choice of component models. Therefore, any analysis of these subsystems can be done in parallel.

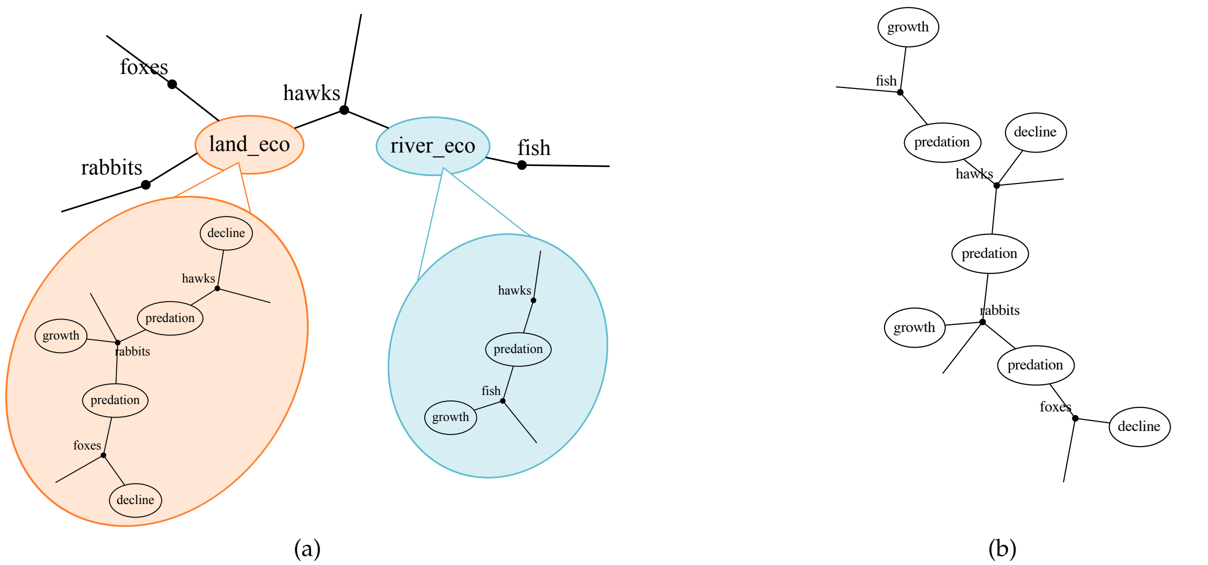


Figure 5: (a) The graphical depiction of three terms of UWD and their hierarchical relationships. Top: `total_diagram`. Bottom left: `land_diagram`. Bottom right: `river_diagram`. The graphics (in black) are produced by the `Catlab.Graphics` module. The coloring highlights the nested structure. (b) The result of substituting the terms for the land and river systems into the term for the total ecosystem is `ocompose(total_diagram, [land_diagram, river_diagram])`.

Example 4.2. A model of an ecosystem is composed of many primitive growth, decline, and predation models where a single species may be involved in multiple subsystems. The compositional approach allows us to divide the problem into interacting subsystems and conquer them independently. At the highest level there are two components to the total ecosystem: a land system and a river system. These subsystems compose by identifying species which act in both. In this case, the hawk populations belonging to the land and river systems are identified in the total system. Figure 5(a) depicts the corresponding term in the operad UWD.

This perspective is modular and hierarchical. It is modular because the models for the land and river ecosystems are defined independently. It is hierarchical because the models for the land and river ecosystems can themselves be composed of even more fine-grained interactions. Figure 5(a) gives terms for the fine-grained interactions within the land and river subsystems respectively and shows how they nest into the high-level diagram. Given this hierarchical structure, there are two different but equivalent strategies to construct the complete ecosystem model. The model elements for the growth, decline, and predation interaction types can be applied either at the level of sub-ecosystems:

```
land_sys = oapply(land_diagram, land_models)
river_sys = oapply(river_diagram, river_models)
total_sys = oapply(total_diagram, [land_sys, river_sys])
```

or at the level of the total ecosystem[3]:

```
eco_diagram = ocompose(total_diagram, [land_diagram, river_diagram])
total_sys = oapply(eco_diagram, vcat(land_models, river_models))
```

Functoriality of $\mathsf{Dynam}_{\mathsf{C}}^{\multimap}$ implies that these strategies produce models with identical denotational semantics, a solution for which is shown in Figure 6. Since syntatic terms can be built and interpreted hierarchically with equivalent results, hierarchical modeling is flexible, scalable, and parallelizable. For example, if we discover a fourth species involved in the land system, then we can adjust its internal syntax and semantics independent of the river system.

```
W = 10; H = 20
stencil = grid(W, H)
point_model =
  DiscreteMachine{Float64}(4, 1, 4,
    (u,x,p,t) -> [p.δ*(sum(x)-4*u[1])+u[1]],
    u -> repeat(u, 4)
  )
heat_model = oapply(stencil, point_model)
```

(a)

(b)

(c)

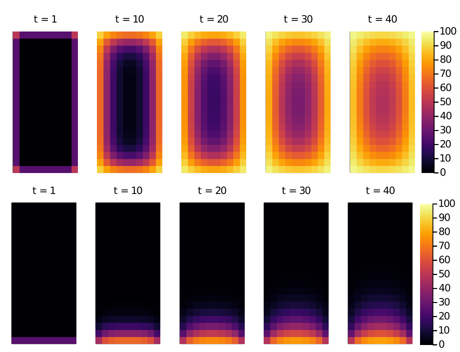


Figure 7: (a) Julia code for a discrete approximation of the heat equation. (b) Simulation of the model over time where the boundary conditions specify that heat enters the system from all sides. (c) Simulation of the model over time where the boundary conditions specify that heat enters the system from the bottom.

Example 4.3. In Definition 2.10, we introduced the operad of circular port graphs CPG and showed that it is a suboperad of DWD. The compositions

$$\mathsf{CPG} \hookrightarrow \mathsf{DWD} \underset{\mathsf{Dynam}_{\mathsf{D}}^{\rightarrow}}{\overset{\mathsf{Dynam}_{\mathsf{C}}^{\rightarrow}}{\rightrightarrows}} \mathcal{O}(\mathbf{Set})$$

define algebras of continuous and discrete systems over circular port graphs. In Algebraic-Dynamics, the `oapply` method for circular port graphs is implemented independently of the `oapply` method for directed wiring diagrams to improve performance.

[3]The term of UWD corresponding to `eco_diagram` is shown in Figure 5(d).

The operad and operad algebras for circular port graphs formalize standard numerical analysis techniques, namely the method of stencils and the finite difference method. Figure 7 gives a solution to the 2D heat equation using the finite difference method. The method `grid(W::Int, H::Int)` returns a circular port graph with $W \times H$ nodes arranged in a grid. Each node has four ports and is connected to its four adjacent neighbors via symmetric wires. The unattached ports of boundary nodes are exposed by open ports. This syntactic term recovers the 5-point stencil.

Example 4.4. So far we have presented three examples of *constructing* models compositionally. The next step is to *analyze* models compositionally, using the mathematical abstraction of natural transformations defined in Section 3.3. Let $\blacksquare : F \Rightarrow G$ be a natural transformation between operad algebras. If the Julia type `S` implements elements of F and the Julia type `T` implements elements of G, then $\blacksquare$ is implemented by a Julia method `blackbox(::S)::T`. AlgebraicDynamics.jl implements Euler's method for both undirected and directed systems.

5 Conclusion

Applied category theory offers rigorous denotational semantics for scientific modeling. The development of Catlab and the AlgebraicJulia ecosystem provide the computational framework for implementing these semantics and making them accessible and practical for scientists. The implementation in AlgebraicDynamics of operad algebras for composing open dynamical systems is a first example of this work and enables modelers to leverage the abstractions of applied category theory to solve real-world problems. In this paper, we have showcased the many advantages that these abstractions offer to modellers, such as hierarchical modeling, hierarchical model interpretation, and independence of model syntax and semantics.

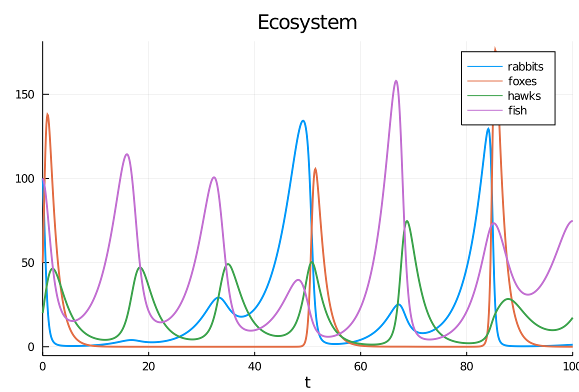


Figure 6: The solution to the complete ecosystem model defined hierarchically in Example 4.2.

In future work, we will lay a foundation for using higher category theory to study relationships between dynamical models. For example, we can use double categories of structured cospans with resource sharing semantics to study how localized changes to model structure affect system behavior. Furthermore, while a theme of this work is "implementations informed by abstractions," its converse "abstractions informed by implementations" is a rich source of mathematical ideas. For instance, we saw that Euler's method is a functorial process, yet many standard numerical methods are not functorial. We aim to develop abstractions characterizing this lossiness and use them to prove accuracy results. As another direction, the algebras presented in Section 3 formalize the concept of real-valued data flowing along wires. However, the Julia implementation suggests that these algebras generalize to types with Frobenius structure. Finally, implementing the terms of operads as C-sets is a useful device for treating syntactic terms as data structures. We conjecture that this is an artifact of a more general theory of operads defined by C-sets.

References

[1] John C. Baez, Brendan Fong & Blake S. Pollard (2016): *A Compositional Framework for Markov Processes*. *Journal of Mathematical Physics* 57(3), p. 033301, doi:10.1063/1.4941578.

[2] John C. Baez & Blake S. Pollard (2017): *A Compositional Framework for Reaction Networks*. *Reviews in Mathematical Physics* 29(09), p. 1750028, doi:10.1142/S0129055X17500283.

[3] John C. Baez, David Weisbart & Adam M. Yassine (2021): *Open Systems in Classical Mechanics*. *Journal of Mathematical Physics* 62(4), p. 042902, doi:10.1063/5.0029885.

[4] Lee DeVille, Eugene Lerman & James Schmidt (2019): *Runge-Kutta and Networks*. *arXiv*:1908.11453 [*cs, math*].

[5] Brendan Fong (2015): *Decorated Cospans*. *Theory and Applications of Categories* 30(33), pp. 1096–1120.

[6] Brendan Fong & David I. Spivak (2019): *Hypergraph Categories*. *arXiv*:1806.08304 [*cs, math*].

[7] David Jaz Myers (2021): *Double Categories of Open Dynamical Systems (Extended Abstract)*. *Electronic Proceedings in Theoretical Computer Science* 333, pp. 154–167, doi:10.4204/EPTCS.333.11.

[8] William Ogilvy Kermack, A. G. McKendrick & Gilbert Thomas Walker (1927): *A Contribution to the Mathematical Theory of Epidemics*. *Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character* 115(772), pp. 700–721, doi:10.1098/rspa.1927.0118.

[9] Tom Leinster (2004): *Higher Operads, Higher Categories*. *London Mathematical Society Lecture Note Series* 298, Cambridge University Press, Cambridge, UK ; New York.

[10] Eugene Lerman & James Schmidt (2020): *Networks of Hybrid Open Systems*. *Journal of Geometry and Physics* 149, p. 103582, doi:10.1016/j.geomphys.2019.103582.

[11] Sophie Libkind (2020): *An Algebra of Resource Sharing Machines*. *arXiv*:2007.14442 [*math*].

[12] Joe Moeller & Christina Vasilakopoulou (2020): *Monoidal Grothendieck Construction*. *Theory and Applications of Categories* 35(31), pp. 1159–1207.

[13] Timothy Ngotiaoco (2017): *Compositionality of the Runge-Kutta Method*. *arXiv*:1707.02804 [*math*].

[14] Marie La Palme Reyes, Gonzalo E Reyes & Houman Zolfaghari (2004): *Generic figures and their glueings: A constructive approach to functor categories*. Polimetrica.

[15] Patrick Schultz, David I. Spivak & Christina Vasilakopoulou (2020): *Dynamical Systems and Sheaves*. *Applied Categorical Structures* 28(1), pp. 1–57, doi:10.1007/s10485-019-09565-x.

[16] Patrick Schultz, David I. Spivak, Christina Vasilakopoulou & Ryan Wisnesky (2017): *Algebraic Databases*. *Theory and Applications of Categories* 32(16), pp. 547–619.

[17] David I. Spivak (2013): *The Operad of Wiring Diagrams: Formalizing a Graphical Language for Databases, Recursion, and Plug-and-Play Circuits*. *arXiv*:1305.0297 [*cs, math*].

[18] David I. Spivak (2016): *The Steady States of Coupled Dynamical Systems Compose According to Matrix Arithmetic*. *arXiv*:1512.00802 [*math*].

[19] David I. Spivak (2020): *Generalized Lens Categories via Functors* $C^{op} \to \mathsf{Cat}$. *arXiv*:1908.02202 [*cs, math*].

[20] David I. Spivak, Magdalen R. C. Dobson, Sapna Kumari & Lawrence Wu (2017): *Pixel Arrays: A Fast and Elementary Method for Solving Nonlinear Systems*. *arXiv*:1609.00061 [*math*].

[21] Dmitry Vagner, David I. Spivak & Eugene Lerman (2015): *Algebras of Open Dynamical Systems on the Operad of Wiring Diagrams*. *Theory and Applications of Categories* 30(51), pp. 1793–1822.

A Proofs

A.1 Circular port graphs

Definition A.1. The theory of circular port graphs with a single box is

$$\mathsf{Th}(\mathsf{CPG})_! := \left\{ \begin{array}{ccc} & & Q \\ & & \downarrow{\scriptstyle \mathsf{expose}} \\ W & \underset{\mathsf{tgt}}{\overset{\mathsf{src}}{\rightrightarrows}} & P \end{array} \right\}.$$

Let X be a finite instance of $\mathsf{Th}(\mathsf{CPG})_!$. Then X induces the morphism

$$\begin{pmatrix} f^{\#}(X) \\ f(X) \end{pmatrix} := \begin{pmatrix} XQ+XP \xleftarrow{X1_Q+X\mathsf{src}} XQ+XW \xrightarrow{(X\mathsf{expose},X\mathsf{tgt})} XP \\ XP \xleftarrow{X\mathsf{expose}} XQ \xrightarrow{X1_Q} XQ \end{pmatrix} : \begin{pmatrix} XP \\ XP \end{pmatrix} \leftrightarrows \begin{pmatrix} XQ \\ XQ \end{pmatrix}$$

in $\mathsf{Lens}_{\mathsf{Cospan}(\mathbf{FinSet}^{\mathrm{op}})}$.

Lemma A.2. The objects of form $\binom{X_{\mathsf{port}}}{X_{\mathsf{port}}}$ for $X_{\mathsf{port}} \in \mathbf{FinSet}^{\mathrm{op}}$ and the morphisms generated by finite instances of $\mathsf{Th}(\mathsf{CPG})_!$ form a symmetric monoidal subcategory of $\mathsf{Lens}_{\mathsf{Cospan}(\mathbf{FinSet}^{\mathrm{op}})}$.

Proof. It suffices to show that the morphisms of $\mathsf{Lens}_{\mathsf{Cospan}(\mathbf{FinSet}^{\mathrm{op}})}$ induced by finite instances of $\mathsf{Th}(\mathsf{CPG})_!$ contain the identities, braidings, associators, and unitors, and are closed under monoidal products and composition.

Let $\binom{X_{\mathsf{port}}}{X_{\mathsf{port}}}$ be an object of $\mathsf{Lens}_{\mathsf{Cospan}(\mathbf{FinSet}^{\mathrm{op}})}$. Then the identity $1_{\binom{X_{\mathsf{port}}}{X_{\mathsf{port}}}}$ is $\binom{f^{\#}(X)}{f(X)}$ where X is the finite instance of $\mathsf{Th}(\mathsf{CPG})_!$ given by the following diagram:

$$\begin{array}{ccc} & & X_{\mathsf{port}} \\ & & \downarrow{\scriptstyle 1_{X_{\mathsf{port}}}} \\ 0 & \underset{!}{\overset{!}{\rightrightarrows}} & X_{\mathsf{port}} \end{array}$$

Similarly, the braidings, associators, and unitors are generated by finite instances of $\mathsf{Th}(\mathsf{CPG})$ that are defined by the braidings, associators, and unitors in $(\mathbf{FinSet}, +, 0)$.

Given finite instances X and Y of $\mathsf{Th}(\mathsf{CPG})_!$, there is a finite instance $X+Y : \mathsf{Th}(\mathsf{CPG})_! \to \mathbf{Set}$ induced by disjoint union in $\mathbf{Set}$. Explicitly, $X+Y$ is defined by the following diagram:

$$\begin{array}{ccc} & & XQ+YQ \\ & & \downarrow{\scriptstyle X\mathsf{expose}+Y\mathsf{expose}} \\ XW+YW & \underset{X\mathsf{tgt}+Y\mathsf{tgt}}{\overset{X\mathsf{src}+Y\mathsf{src}}{\rightrightarrows}} & XP+YP \end{array}$$

Thus, $\binom{f^{\#}(X)}{f(X)} + \binom{f^{\#}(Y)}{f(Y)} = \binom{f^{\#}(X+Y)}{f(X+Y)}$.

Let X and Y be finite instances of $\mathsf{Th(CPG)}_!$ such that $XP = YQ$. Define $X \circ Y$ to be the finite instance of $\mathsf{Th(CPG)}_!$ given by the following diagram:

$$\begin{array}{ccc} & & XQ \\ & & \big\downarrow {\scriptstyle Y\mathsf{expose}\circ X\mathsf{expose}} \\ XW + YW & \underset{(Y\mathsf{expose}\circ X\mathsf{tgt})+Y\mathsf{tgt}}{\overset{(Y\mathsf{expose}\circ X\mathsf{src})+Y\mathsf{src}}{\rightrightarrows}} & YP \end{array}$$

Then we have $\binom{f^\#(X)}{f(X)} \circ \binom{f^\#(Y)}{f(Y)} = \binom{f^\#(X\circ Y)}{f(X\circ Y)}$. □

Recall that a term of the operad DWD defined in Example 2.3 consists of

- a natural number n
- domain types $\binom{X_{\text{in},i}}{X_{\text{out},i}} \in \operatorname{ob}\mathsf{Lens}_{\mathsf{Cospan}(\mathbf{FinSet}^{\text{op}})}$ for $i = 1, \ldots, n$
- a codomain type $\binom{Y_{\text{in}}}{Y_{\text{out}}} \in \operatorname{ob}\mathsf{Lens}_{\mathsf{Cospan}(\mathbf{FinSet}^{\text{op}})}$
- a morphism $\binom{f^\#}{f} : \binom{X_{\text{in},1}}{X_{\text{out},1}} + \cdots + \binom{X_{\text{in},n}}{X_{\text{out},n}} \leftrightarrows \binom{Y_{\text{in}}}{Y_{\text{out}}}$ in $\mathsf{Lens}_{\mathsf{Cospan}(\mathbf{FinSet}^{\text{op}})}$.

Let X be a finite instance of $\mathsf{Th(CPG)}$ with $\#(XB) = n$. Fix an ordering $(b_1, \ldots, b_n)$ of the elements $\{b_1, \ldots, b_n\} = XB$. Note that $(X\mathsf{box})^{-1}(b_1) + \cdots + (X\mathsf{box})^{-1}(b_n) = XP$. The term of DWD induced by X has

- $n = \#(XB)$
- domain types $\binom{(X\mathsf{box})^{-1}(b_i)}{(X\mathsf{box})^{-1}(b_i)}$ for $i = 1, \ldots, n$
- codomain type $\binom{XQ}{XQ}$
- morphism $\binom{f^\#(\Delta_J X)}{f(\Delta_J X)}$ where $J : \mathsf{Th(CPG)}_! \hookrightarrow \mathsf{Th(CPG)}$ is the obvious inclusion.

In Definition 2.10 we defined CPG to be the suboperad of DWD whose type are pairs $\binom{X_{\mathsf{port}}}{X_{\mathsf{port}}}$ and whose terms are generated by finite instances of $\mathsf{Th(CPG)}$.

Proposition A.3. Every term of CPG is induced by a finite instance of $\mathsf{Th(CPG)}$.

Proof. Consider the term ϕ of CPG consisting of

- a natural number n
- domain types $\binom{X_{\mathsf{port},i}}{X_{\mathsf{port},i}} \in \operatorname{ob}\mathsf{Lens}_{\mathsf{Cospan}(\mathbf{FinSet}^{\text{op}})}$ for $i = 1, \ldots, n$
- a codomain type $\binom{Y_{\mathsf{port}}}{Y_{\mathsf{port}}} \in \operatorname{ob}\mathsf{Lens}_{\mathsf{Cospan}(\mathbf{FinSet}^{\text{op}})}$
- a morphism $\binom{f^\#}{f} : \binom{X_{\mathsf{port},1}}{X_{\mathsf{port},1}} + \cdots + \binom{X_{\mathsf{port},n}}{X_{\mathsf{port},n}} \leftrightarrows \binom{Y_{\mathsf{port}}}{Y_{\mathsf{port}}}$ in $\mathsf{Lens}_{\mathsf{Cospan}(\mathbf{FinSet}^{\text{op}})}$.

Since CPG underlies the symmetric monoidal category defined in Lemma A.2, there exists a finite instance $\tilde{X}$ of $\mathsf{Th(CPG)}_!$ such that $\binom{f^\#}{f} = \binom{f^\#(\tilde{X})}{f(\tilde{X})}$. Note that $\tilde{X}$ must satisfy $\tilde{X}P = X_{\mathsf{port},1} + \cdots + X_{\mathsf{port},n}$. Let $\pi : \tilde{X}P \to \{1, \ldots, n\}$ be the partition defined by $\pi^{-1}(i) = X_{\mathsf{port},i}$ for $i = 1, \ldots, n$.

Then ϕ is induced by the instance of $\mathsf{Th(CPG)}$ given by the diagram

$$\begin{array}{cccc} & & \tilde{X}Q & \\ & & \big\downarrow {\scriptstyle \tilde{X}\mathsf{expose}} & \\ \tilde{X}W & \underset{\tilde{X}\mathsf{tgt}}{\overset{\tilde{X}\mathsf{src}}{\rightrightarrows}} & \tilde{X}P & \xrightarrow[\pi]{} \{1, \ldots, n\} \end{array}$$

□

A.2 Algebras for Composing Dynamical Systems

Definition A.4. Let $f : S \to S'$ in **FinSet**. Define the pullback $f^* : \mathbb{R}^{S'} \to \mathbb{R}^S$ by $f^*(x')(\sigma) = x'(f(\sigma))$ for $x' \in \mathbb{R}^{S'}$, $\sigma \in S$. The pullback induces a contravariant functor **FinSet** $\to$ Euc.

Define the pushforward $f_* : \mathbb{R}^S \to \mathbb{R}^{S'}$ by

$$f_*(x)(\sigma') = \sum_{\sigma \in f^{-1}(\sigma')} x(\sigma).$$

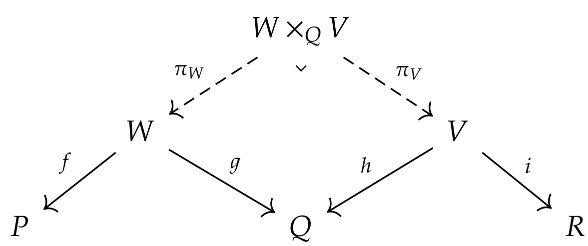
for $x \in \mathbb{R}^S$, $\sigma' \in S'$. The pushforward induces a covariant functor **FinSet** $\to$ Euc.

Both functors commute with scaling. So for $h \in \mathbb{R}$, $f_*(hx) = hf_*(x)$ and $f^*(hx) = hf^*(x)$.

A.2.1 Directed Systems

Proposition A.5. There exists a strong monoidal functor $\mathbf{ev}_{\mathbb{R}} : \mathsf{Cospan}(\mathbf{FinSet}^{\mathrm{op}}) \to \mathsf{Euc}$ defined on objects by $P \mapsto \mathbb{R}^P$ and on morphisms by $P \xleftarrow{f} W \xrightarrow{g} Q$ to $g_* \circ f^* : \mathbb{R}^P \to \mathbb{R}^Q$.

Proof. To show that $\mathbf{ev}_{\mathbb{R}}$ preserves composition, it suffices to show that for the diagram



we have $h^* \circ g_* = (\pi_V)_* \circ \pi_W^*$. We verify this explicitly. For $x \in \mathbb{R}^W$, $\sigma \in V$,

$$\begin{aligned}
(h^* \circ g_*)(x)(\sigma) &= g_* x(h(\sigma) \\
&= \sum_{\rho \in g^{-1}(h(\sigma))} x(\rho) \\
&= \sum_{(\rho,\sigma) \in W \times_Q V} x(\pi_W(\rho, \sigma)) \\
&= \sum_{(\rho,\sigma) \in W \times_Q V} \pi_W^* x(\rho, \sigma) \\
&= ((\pi_V)_* \circ \pi_W^*) x(\sigma).
\end{aligned}$$

Preservation of identity follows from the fact that the pullback and pushforward functors preserve identity. Strength follows from the natural isomorphism $\mathbb{R}^{P+P'} \simeq \mathbb{R}^P \times \mathbb{R}^{P'}$. □

A.2.2 Undirected Systems

Lemma A.6. There is a functor $D_{\mathsf{D}} : \mathbf{FinSet} \to \mathbf{Set}$ which on objects maps a finite set S to the set of set maps $\mathbb{R}^S \to \mathbb{R}^S$. For a morphism $f : S \to S'$ in **FinSet**, $D_{\mathsf{D}}(f)$ is defined by $D_{\mathsf{D}}(f)(u) = 1_{\mathbb{R}^{S'}} + f_* \circ (u - 1_{\mathbb{R}^S}) \circ f^*$.

Proof. D_{D} preserves identities. Let $1_S : S \to S$ and $u \in D_{\mathsf{D}}(S)$. Then

$$D_{\mathsf{D}}(1_S)(u) = 1_{\mathbb{R}^S} + 1_{\mathbb{R}^S} \circ (u - 1_{\mathbb{R}^S}) \circ 1_{\mathbb{R}^S} = u.$$

D_{D} preserves compositions. Let $f : S \to S'$, $g : S \to S''$, and $u \in D_{\mathsf{D}}(S)$. Then,

$$\begin{aligned} D_{\mathsf{D}}(g \circ f)(u) &= 1_{\mathbb{R}^{S''}} + (g \circ f)_*(u - 1_{\mathbb{R}^S})(g \circ f)^* \\ &= 1_{\mathbb{R}^{S''}} + g_* \circ \left(f_* \circ (u - 1_{\mathbb{R}^S}) \circ f^*\right) \circ g^* \\ &= 1_{\mathbb{R}^{S''}} + g_* \circ \left(D_{\mathsf{D}} f(u) - 1_{\mathbb{R}^{S'}}\right) \circ g^* \\ &= (D_{\mathsf{D}}(g) \circ D_{\mathsf{D}}(f))(u). \end{aligned}$$ □

Lemma A.7. $D_{\mathsf{D}} : (\mathbf{FinSet}, +, 0) \to (\mathbf{Set}, \times, 1)$ is a lax monoidal functor when equipped with the unique map $\eta : 1 \to D_{\mathsf{D}}(0)$ and the natural transformation

$$\delta_{S,S'} : D_{\mathsf{D}}(S) \times D_{\mathsf{D}}(S') \to D_{\mathsf{D}}(S + S')$$

defined by

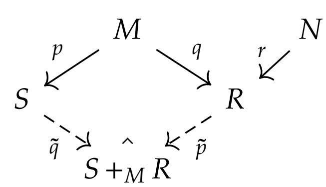
$$(u, u') \mapsto i_* \circ u \circ i^* + i'_* \circ u' \circ i'^*$$

where $i : S \to S + S', i' : S' \to S + S'$ are the natural inclusion maps.

Proof. Straightforward. □

Proposition 3.1. There is an algebra $\mathsf{Dynam}_{\mathsf{D}}^{-\circ} : \mathsf{Cospan}(\mathbf{FinSet}) \to \mathbf{Set}$ which on objects maps M to the set of triples $(S \in \mathbf{FinSet}, u : \mathbb{R}^S \to \mathbb{R}^S, p : M \to S)$ and on morphisms maps the cospan $f = M \xrightarrow{q} R \xleftarrow{r} N$ to the set map $\mathsf{Dynam}_{\mathsf{D}}^{-\circ}(f) : \mathsf{Dynam}_{\mathsf{D}}^{-\circ}(M) \to \mathsf{Dynam}_{\mathsf{D}}^{-\circ}(N)$ defined by

$$\mathsf{Dynam}_{\mathsf{D}}^{-\circ}(f)(S, u, p) = (S +_M R, 1_{\mathbb{R}^{S+_M R}} + \tilde{q}_* \circ (u - 1_{\mathbb{R}^S}) \circ \tilde{q}^*, \tilde{p} \circ r).$$



Proof. The decorating functor D_{D} defines a hypergraph category of decorated cospans [5]. The algebra $\mathsf{Dynam}_{\mathsf{D}}^{-\circ}$ is induced by the 1-equivalence between hypergraph categories and $\mathsf{Cospan}(\mathbf{FinSet})$ algebras defined in [6]. □

Proposition 3.3 (Euler's method for undirected systems)**.** For $h \in \mathbb{R}_+$, there exists a natural transformation $\mathsf{Euler}_h^{-\circ} : \mathsf{Dynam}_{\mathsf{C}}^{-\circ} \Rightarrow \mathsf{Dynam}_{\mathsf{D}}^{-\circ}$ with components $\mathsf{Euler}_h^{-\circ}(M) : \mathsf{Dynam}_{\mathsf{C}}^{-\circ}(M) \to \mathsf{Dynam}_{\mathsf{D}}^{-\circ}(M)$ defined by $\mathsf{Euler}_h^{-\circ}(M)(S, v, p) = (S, \mathsf{Euler}_h(v), p)$.

Proof. Theorem 17 of [2] defines a hypergraph category Dynam as a decorated cospan category with decorating functor $D : (\mathbf{FinSet}, +, 0) \to (\mathbf{Set}, \times, 1)$ defined

- on objects: S maps to the set of continuous maps $v : \mathbb{R}^S \to \mathbb{R}^S$[4]
- on morphisms: $f : S \to S'$ in **FinSet** maps to $Df : \mathbb{R}^S \to \mathbb{R}^{S'}$ defined by $Df(v) = f_* \circ v \circ f^*$.

The algebra $\mathsf{Dynam}_{\mathsf{C}}^{-\circ}$ is the $\mathsf{Cospan}(\mathbf{FinSet})$ algebra induced by this hypergraph category.

Let $h \in \mathbb{R}_+$. Recall that for a map $v : \mathbb{R}^S \to \mathbb{R}^S$, $\mathsf{Euler}_h(v) = 1_{\mathbb{R}^S} + hv$. Let $f : S \to S'$. Then,

$$\begin{array}{ccc} D(S) & \xrightarrow{\mathsf{Euler}_h} & D_{\mathsf{D}}(S) \\ {\scriptstyle D(f)}\big\downarrow & & \big\downarrow{\scriptstyle D_{\mathsf{D}}(f)} \\ D(S') & \xrightarrow[\mathsf{Euler}_h]{} & D_{\mathsf{D}}(S') \end{array}$$

commutes because for $v : \mathbb{R}^S \to \mathbb{R}^S$,

$$\begin{aligned} (D_{\mathsf{D}}(f) \circ \mathsf{Euler}_h)(v) &= 1_{\mathbb{R}^{S'}} + f_* \circ (\mathsf{Euler}_h v - 1_{\mathbb{R}^S}) \circ f^* \\ &= 1_{\mathbb{R}^{S'}} + f_* \circ (hv) \circ f^* \\ &= 1_{\mathbb{R}^{S'}} + h(f_* \circ v \circ f^*) \\ &= \mathsf{Euler}_h(f_* \circ v \circ f^*) \\ &= (\mathsf{Euler}_h \circ Df)(v). \end{aligned}$$

Therefore, $\mathsf{Euler}_h : D \Rightarrow D_{\mathsf{D}}$ is a natural transformation. This natural transformation defines a functor between the hypergraph categories defined by D and D_{D} (see [5]) which in turn induces the transformation $\mathsf{Euler}_h^{-\circ}$ (see [6]). □

[4]In [2] DS is defined to be the set of *algebraic* maps $v : \mathbb{R}^S \to \mathbb{R}^S$. The more general definition presented here does not affect the results.