Operadic diagnosis in hierarchical systems

Spencer Breiner[1], Olivier Marie-Rose[2], Blake S. Pollard[1], and Eswaran Subrahmanian[1,3]

[1]US National Institute of Standards and Technology, Gaithersburg, MD

[2]Prometheus Computing, Gaithersburg, MD

[3]Carnegie Mellon University, Pittsburgh, PA

This paper applies operads and functorial semantics to address the problem of failure diagnosis in complex systems. We start with a concrete example, developing a hierarchical interaction model for the Length Scale Interferometer, a high-precision measurement system operated by the US National Institute of Standards and Technology. The model is expressed in terms of combinatorial/diagrammatic structures called port-graphs, and we explain how to extract an operad LSI from a collection of these diagrams. Next we show how functors to the operad of probabilities organize and constrain the relative probabilities of component failure in the system. Finally, we show how to extend the analysis from general component failure to specific failure modes.

1 Introduction

Hierarchical systems are ubiquitous in nature, in engineering and in commerce. We draw trees to represent anatomic features, component breakdowns and command structures. Hierarchies use abstraction barriers and separation of concerns to analyze and simplify complex problems into manageable parts.

However, to represent a system as a tree involves abstracting away the interactions between its branches. An alternative viewpoint, exemplified by models called port-graphs, emphasizes the leaves of the tree (components) and the way that they are wired together to form a larger system. This additional information is critical for system analysis, but such diagrams quickly

become unwieldy as the number of components grows.

Operads provide a nice mathematical formalism for interpolating between these two viewpoints. Our goal in this paper is to demonstrate, in concrete terms, the use of operads to organize both qualitative and quantitative descriptions of hierarchical systems. To that end, we begin by modeling a specific real-world system, the Length Scale Interferometer (LSI), a device for precise length calibration operated by the US National Institute for Standards and Technology.

We begin with a brief sketch of the LSI, its purpose and its design, followed by an informal review of operads. Next, we construct a specific operad LSI based on the architecture of the LSI system and expressed in terms of combinatorial/diagrammatic structures called port-graphs. This forms the basis for a functorial analysis of failure diagnosis. First, we can consider the relative probabilities of component and sub-component failure as a functor $\mathsf{LSI} \to \mathsf{Prob}$. Finally, we show how to integrate component level probabilities with more specific information about specific failure modes.

2 The Length Scale Interferometer

The Length Scale Interferometer (LSI) [1] is a precision length-measurement system operated by the US National Institute for Standards and Technology (NIST). Customers from around the world send length scales, marked plates or rods ranging in size from a millimeter to a meter, to be calibrated at NIST's Gaithersburg, MD campus. Using laser interferometry, the device accurately measures lengths to better than one part in one million (0.7 μm error across a one meter length), and can resolve markings on a scale down to 0.1 μm.

More formally, a *nominal length scale* is defined by two distances: the total span D and the spacing d, where $N = \frac{D}{d} \in \mathbb{N}$; for example, a typical meter stick with millimeter hatch marks would have $D = 1\ \mathrm{m}$ and $d = 1\ \mathrm{mm}$. The scale has $N + 1$ hatch marks located a $0, d, 2d, \ldots Nd = D$. For a real scale we can set our frame of reference Y by the first mark on the scale, but each of the others will exhibit an error ε_i, and the purpose of the LSI system is to identify these errors.

The basic idea is simple enough. The scale is installed on a movable chassis, which also contains one mirror of an interferometer. The length of the laser's path changes as the scale and chassis move, allowing us to infer the scale's position from the fringe count of the interferometer.

A calibration starts from the initial mark at $y_0 = 0$ and records the current fringe count f_0 on the interferometer's readout. The machine scans to the next hatch mark y_1, identified by an optical system, and locks the position of the chassis. The new fringe reading f_1, along with the laser's wavelength λ_0 and the index of refraction n, determines the associated error $y_1 = d + \varepsilon_1 = \lambda_0 \cdot n \cdot (f_1 - f_0)$. We do the same for each of the rest of the marks, noting that all distances and errors are calculated relative to y_0 (rather than y_{i-1}).

In practice, there are a number of complications, of which the most significant concern unavoidable fluctuations in the environment. Here we are concerned with two main effects. First, the index of refraction, which we use to calculate lengths and errors, depends on temperature.[1] Second, thermal expansion means that the relative positioning of system elements changes over time.

For example, the relative separation between (the first hatch mark of) the scale and the chassis' mirror will vary due to expansion of the metal that connects them. Similarly, the scale itself expands, so that the positions of the hatch marks and their errors will vary. Consequently, we should reformulate the goal of the LSI as measuring length scales *at a standard temperature of* $20\ ^\circ\mathrm{C}$.

This variation has two practical implications for the LSI system. First, we must apply temperature-dependent corrections to the gross measurements of the system (fringe counts). This is necessary both to calculate the true lengths that were measured and to convert from these to the desired (temperature-corrected) values. Second, in order to minimize correction error, the LSI must maintain environmental values as close to the target as possible; to obtain the $0.7\ \mu$m accuracy mentioned above, the system must operate within $.02\ ^\circ\mathrm{C}$ of the target value.

3 Operads

A (colored, symmetric) operad is a mathematical structure for representing hierarchical (tree-structured) composition and decomposition. The fundamental element of an operad is an *operation* α, which we usually write as $\alpha : Q_1, \ldots, Q_m \to P$ (or more compactly, $\langle Q_i \rangle \to P$). Every operation is pa-

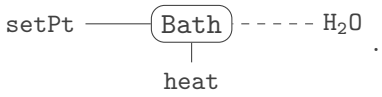
[1] The index of refraction also depends on pressure and humidity, but we ignore these in the interest of simplicity.

rameterized by a list of input objects $Q_1, \ldots, Q_m$ and a single output object P. When the input elements are not relevant, we may simply write $\alpha : P$

If we have additional operations $\beta_i : Q_i$, operadic composition substitute these for the "dummy variables" Q_i to obtain a new operation $\alpha(\beta_1, \ldots, \beta_n)$ (or $\alpha\langle\beta_i\rangle$). If β_i has n_i inputs, then the new composite has $n_1 + \ldots + n_m$. Operadic associativity guarantees a well-defined composite for more deeply nested terms.

We will work with a typed variant of Spivak's operad of wiring diagrams [2]. We begin with a set of *interfaces* IType, which represent all the channels of interaction that occur within our system. These include both physical interactions (`heat` flow)) and digital signals (`temp` measurements). Formally, we specify IType at the outset, although in practice it can be inferred from usage in system diagrams.

Given IType, a *boundary* is a set of *ports* P together with a map (often left implicit) $P \to$ IType. We draw an interface as a box with $|P|$-many terminals, each labeled by a type (distinguished, if necessary, by subscripts). For example, the bath used in the LSI's temperature regulation system has a boundary



This indicates that the bath has three main interactions: heat flow to the length measurement enclosure, chilled water provided from an outside system and a data stream that modifies the set point of an internal heating controller. For now our labels are just place-holders, but they will be refined as we elaborate the model.

An operation $\alpha : \langle Q_i \rangle \to P$ is a system architecture, modeled as a port-graph, in which Q_i are sub-component boundaries and P is an external system boundary. The ports in P and Q_i can be connected via wires, which are also typed, and more properly described as hyper-wires given they can connect any number of ports.

For example the LSI's temperature control system, shown in Figure 1, has three subsystems: one controls the ambient lab temperature (± 0.2 °C), another measures the internal temperature of the measurement enclosure ($\pm .005$ °C) and a third manages the chilled water bath used to control the system's temperature.

Physically, the bath and the lab environment both impact the enclosure through heat flow and this, in turn, affects the length measurement subsystem

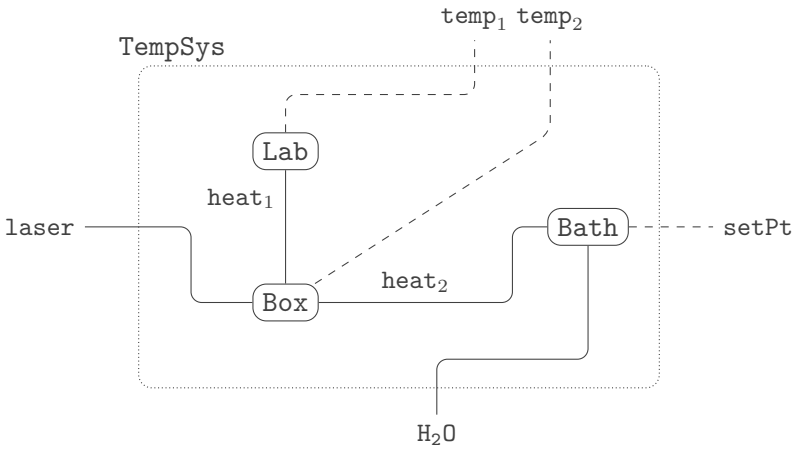


Figure 1: A schematic decomposition of the LSI's temperature regulation system.

via refraction of the laser. Another physical input to the system, a chilled water source, is a shared resource controlled outside of the lab.

There are also digital flows involved in the system. Two streams of temperature data are produced by the lab and the enclosure. There is also a control interaction, with a variable set point for a heating element in the bath. Diagrammatically, it can be useful to distinguish between physical and informational flows; for example, the latter have only indirect effects on the temperature of the system, in contrast to the physical inputs. Formally, the use of different notations for different interaction types can be justified by "typing the types" via a function $\mathsf{IType} \to \{\mathrm{physical}, \mathrm{digital}\}$.

We can succinctly represent a port-graph architecture α as a zig-zag of functions, two of which commute over types:

$$\begin{array}{ccccccc} C & \longleftarrow & Q & \longrightarrow & W & \longleftarrow & P \\ & & & \searrow & \downarrow & \swarrow & \\ & & & & \mathsf{IType} & & \end{array} \tag{1}$$

Here $C = \mathsf{comp}(\alpha) \cong \{1, \ldots, n\}$ is the set of (black-box) components in the architecture, $Q = \coprod_i Q_i$ is the set of internal ports, W is the set of wires and P is the set of external ports. Both ports and wires are typed, and the resulting triangles should commute.

Operadic composition acts by substituting one architecture into another. For example, Figure 2 shows the result of substituting a low-level architecture $\texttt{Mixer}, \texttt{Resevoir}, \texttt{Heater} \to \texttt{Bath}$ for the `Bath` component of in Figure 1.

In the general case we may substitute for all Q_i simultaneously, so suppose

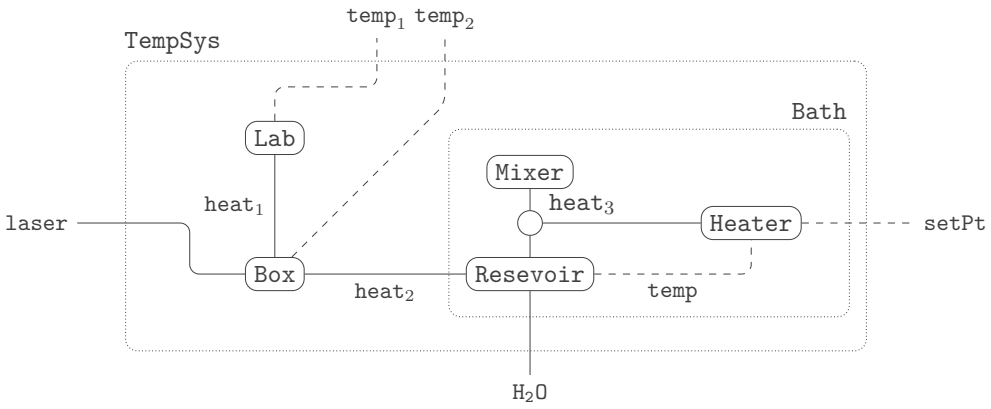


Figure 2: The operadic composition of two architectures $\texttt{Bath}, \texttt{Lab}, \texttt{Box} \to \texttt{TempSys}$ and $\texttt{Mixer}, \texttt{Resevoir}, \texttt{Heater} \to \texttt{Bath}$.

that we have a decomposition (suppressing types) $D_i \leftarrow R_i \to V_i \leftarrow Q_i$ for each of the components. Clearly the external ports P are unchanged by substitution inside Q_i. The set of internal boundaries is given by $D = \coprod_i D_i$; similarly the internal ports of the architecture will be $R = \coprod_i R_i$.

Finally, we can compute the wire set for the aggregate by identifying an outer wire w with an inner wire v_i whenever they share an intermediate port Q_i. Formally, this corresponds to a pushout, and typing on the new wires exists by virtue of the associated mapping property:

$$\begin{array}{ccccc} & & V \underset{Q}{+} W & & \\ & \nearrow & & \nwarrow & \\ \coprod_i D_i \leftarrow \coprod_i R_i \longrightarrow \coprod_i V_i & & & & W \leftarrow P \\ & \nwarrow & & \nearrow & \\ & & \coprod_i Q_i & & \end{array} \tag{2}$$

Equations 1 and 2 (along with the set IType) define an operad PortGraph.

One nice feature of operads is that, although they allow us to represent hierarchical information, they do not constrain us to work within a single hierarchy. In particular, different perspectives often suggest different decompositions for a system, and we can make sense of these distinctions using equations in the operad.

An example is shown in Figure 3. On one hand we can decompose the LSI system functionally (blue boundaries), separating the elements involved in length measurement from those used for temperature control. The second shows a more generic, control-theoretic perspective (red boundaries), separating out components that produce observations (sensors) from those which

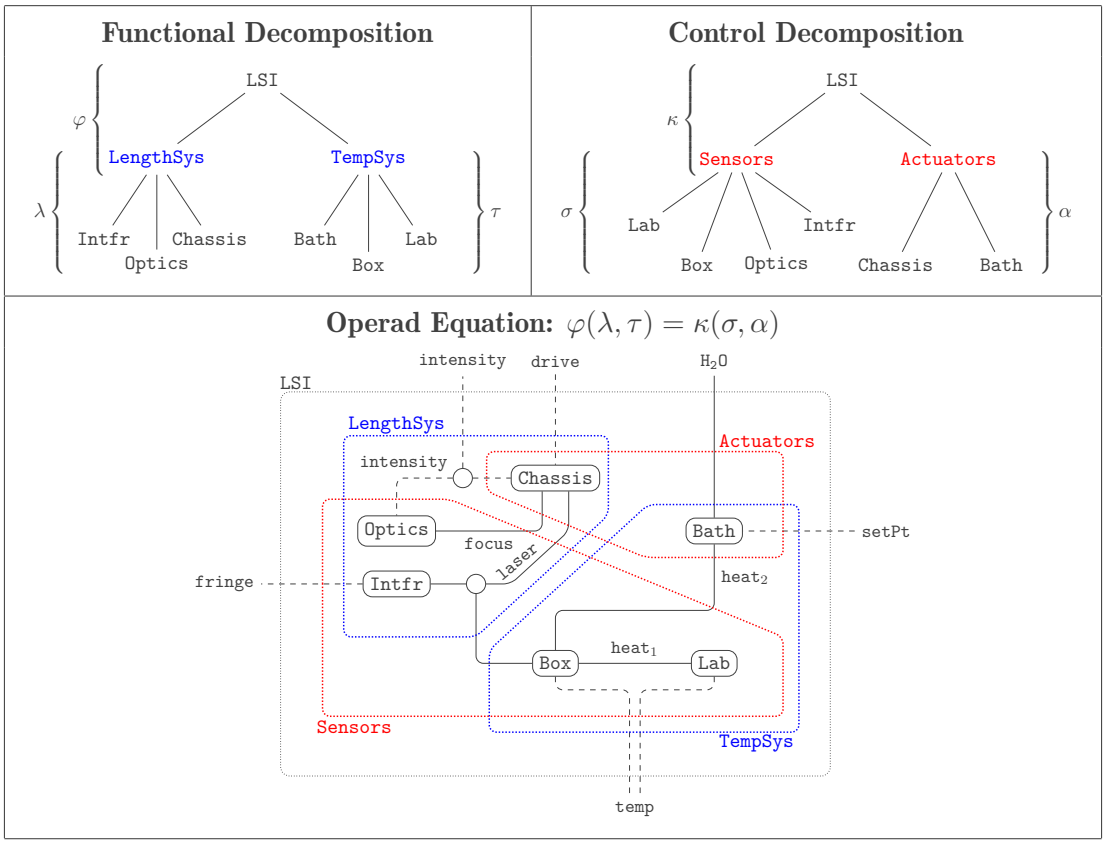


Figure 3: Operad equations represent common refinements between distinct hierarchies.

modify the state of the system (actuators). We could also consider a "geographic" decomposition based on the physical access to the components (e.g., inside/outside the system enclosure). An equation in the PortGraph operad indicates that two incompatible hierarchies (e.g., function vs. control) have a common refinement at a lower level of abstraction.

On first encounter, it can be easy to mix up port graph diagrams, which represent specific architectures, with the PortGraph operad, which represents *all possible* architectures. A specific system, as represented by a collection of diagrams, can be compiled into a "sub-operad" (faithful functor) LSI $\subseteq$ PortGraph involving only the boundaries and architectures that appear in our diagrams.[2] A complete description of the LSI operad, compiled from Figures 1, 2 and 3, is given in the Appendix, Tables 3 & 4.

[2]However, note that diagrams with nested interfaces encode multiple architectures; six operations (φ, λ, τ, κ, σ and α) and an equation can be extracted from the single diagram in Figure 3.

4 Component Failure

In the last section we constructed an operad LSI whose operations are hierarchically organized architectures of the Length Scale Interferometer. Once we have laid out the architecture of the system, we can use it to structure system analyses.

The general principle of functorial semantics is that a model can be regarded as a mapping (called a functor) from architectural (syntactic) elements to computational (semantic) ones, but that this mapping must be assigned in a way that respects the compositional structure of the syntax. Probability provides a good example.

A very simple model of component failure might conclude based on historical data that a failure in the LSI will be located in the temperature regulation subsystem 60% of the time, and in the length measurement subsystem the remaining 40% of the time. The probability distribution $p = (ls \mapsto 40\%, ts \mapsto 60\%)$ is itself an operation in an operad, and we can think of the failure model as a mapping from the functional architecture φ to the distribution p.

Suppose, moreover, that we also know the probabilities of failures within the temperature regulation system, which can involve the bath ($ba \mapsto 80\%$), the lab temperature system ($lb \mapsto 10\%$) or enclosure (box) temperature measurement system ($bx \mapsto 10\%$). Relative probabilities compose by multiplication (conditioning), so that given a fault somewhere in the LSI, there is a $80\% \times 60\% = 48\%$ chance that it involves the bath. This is an instance of functoriality: composite architectures map to composite distributions.

More precisely, there is an operad of probabilities called Prob. Prob is a "plain" operad, so that each operation has a fixed number of inputs (arity) m (in contrast to PortGraph, where each operation π has a "shape" $\langle Q_i \rangle \to P$). An operation Prob is just a probability distribution $p = \langle i \mapsto p_i \rangle_{i<m}$, and its arity is the size of the index set m. The operadic (tree-structured) composition of p with m-many additional distributions $q_i = \langle j \mapsto q_{ij} \rangle_{j<m_i}$ is defined by

$$p(q_1, \ldots, q_n) = \Big\langle (i,j) \mapsto p_i \cdot q_{ij} \Big\rangle, \tag{3}$$

where the index set is the collection of pairs $(i < m, j < m_i)$.

Now we can think of the simple failure model described above as a functor fail : LSI $\to$ Prob. The data associated with a specific instance is shown in Table 1. It sends each architecture $Q_1, \ldots, Q_n \to P$ to a probability distribution on n

LSI Component Failure Model

$\varphi(ls, ts)$	ls	$\mapsto$	40%	$\kappa(sn, ac)$	sn	$\mapsto$	28%
	ts	$\mapsto$	60%		ac	$\mapsto$	72%
$\lambda(la, op, ch)$	la	$\mapsto$	10%	$\sigma(lb, bt, op, la)$	lb	$\mapsto$	21.4%
	op	$\mapsto$	30%		bt	$\mapsto$	21.4%
	ch	$\mapsto$	60%		op	$\mapsto$	42.9%
$\tau(ba, bx, lb)$	ba	$\mapsto$	80%		la	$\mapsto$	14.3%
	bx	$\mapsto$	10%	$\alpha(ch, ba)$	ch	$\mapsto$	33.3%
	lb	$\mapsto$	10%		ba	$\mapsto$	66.7%
$\beta(ht, mx, rs)$	ht	$\mapsto$	50%				
	mx	$\mapsto$	30%				
	rs	$\mapsto$	20%				

Table 1: A probabilistic model of component failure presented as an operad functor LSI $\to$ Prob.

elements, which should be interpreted as the relative probability of a failure in Q_i, *given a failure in* P.

In order to define a functor, these probability assignments should satisfy the LSI's composition equation (Figure 3). This defines some global constraints on how probability can be apportioned between different systems. For example, if we know that problems with actuators are responsible for 72% of faults in the LSI ($\kappa : ac \mapsto 72\%$), then we can infer that the bath is twice as likely to malfunction as the chassis. After all, the bath is equally likely to fail whether we think of it as an actuator or a temperature system component. All in all, the LSI coherence equation $\varphi(\lambda, \tau) = \kappa(\sigma, \alpha)$ generates six probability equations, corresponding to the six components involved in the composed architecture.

$$\begin{array}{rcccccc}
in : \texttt{Intfr} & \mapsto & \overbrace{40\%}^{\varphi} \times \overbrace{10\%}^{\lambda} & = & 4\% & = & \overbrace{28\%}^{\kappa} \times \overbrace{14.3\%}^{\sigma} \\
op : \texttt{Optics} & \mapsto & 40\% \times 30\% & = & 12\% & = & 28\% \times 42.9\% \\
ch : \texttt{Chassis} & \mapsto & 40\% \times 60\% & = & 24\% & = & 72\% \times \overbrace{33.3\%}^{\alpha} \\
ba : \texttt{Bath} & \mapsto & 60\% \times \overbrace{80\%}^{\tau} & = & 48\% & = & 72\% \times 66.7\% \\
bt : \texttt{Box} & \mapsto & 60\% \times 10\% & = & 6\% & = & 28\% \times \overbrace{21.4\%}^{\sigma} \\
rt : \texttt{Lab} & \mapsto & 60\% \times 10\% & = & 6\% & = & 28\% \times 21.4\%
\end{array} \tag{4}$$

Even in the absence of data we can represent these equations symbolically–the top line corresponds to $\varphi(ls) \cdot \lambda(in) = \kappa(sn) \cdot \sigma(in)$–and in this form we can view

operadic coherence equations data integrity constraint or rules of inference.

5 Failure Modes

Though the operadic structure of the LSI model is useful for organizing failure probabilities, our analysis so far is deficient in that it considers only the components that fail, and not what goes wrong with them. In this section we describe a larger operad $\mathsf{LSI} \subseteq \mathsf{LSIFail}$ constructed from the failure modes of the LSI, and show how an extension of our original probability functor assigns error-specific probabilities consistent with the original model.

We start by associating each boundary P with a set of failure modes $\mathsf{Err}(P)$. These should be referenced to explicit features of the component boundary. For example, $\mathsf{Err}(\texttt{TempSys})$ will include at least four errors corresponding to temperature deviations in the lab and in the measurement enclosure: $\texttt{temp}_1 > 20.5\ ^\circ\mathrm{C}$, $\texttt{temp}_1 < 19.5\ ^\circ\mathrm{C}$, $\texttt{temp}_2 > 20.02\ ^\circ\mathrm{C}$ and $\texttt{temp}_2 < 19.98\ ^\circ\mathrm{C}$. Additionally, we will assume that each set $\mathsf{Err}(P)$ contains a distinguished element $* = *_P$ representing an unspecified failure (like those discussed in the previous section).

Furthermore, given an architecture $\alpha : Q_i \to P$ we can identify which errors in Q_i might lead to a given error in P. This defines a causality relation

$$(e, f) \in \mathsf{Err}(\alpha) \subseteq \mathsf{Err}(P) \times \left(\coprod_i \mathsf{Err}(Q_i) \right) \tag{5}$$

indicating that f may lead to e in the context α. Our only restriction is that $(e, *) \in \mathsf{Err}(\alpha)$ implies then $e = *$ as well; intuitively, specific failure modes can cause generic errors, but not vice versa. Formally, this data defines an functor from LSI to the monoidal category of relations $(\mathsf{Rel}, +, \emptyset)$; the $*$ elements and their restrictions can be introduced using a sort of "operadic monad" $A \mapsto A+1$ on Rel.

Next we would like to associate relative probabilities with these failure modes, and to do so in a way that is consistent with our existing model of component failure. To that end we construct an extension $\mathsf{LSIFail} \supseteq \mathsf{LSI}$.

Intuitively, $\mathsf{LSIFail}$ is organized into two layers, as shown in Figure 4. The lower layer consists of generic errors $(P, *)$ and forms a copy of the original LSI operad. The upper layer contains two types of failure modes: a priori failures (P, e) and relativized failures (α, e, f); these interact via refined versions of the LSI operations. Composition within each layer can be computed by reference

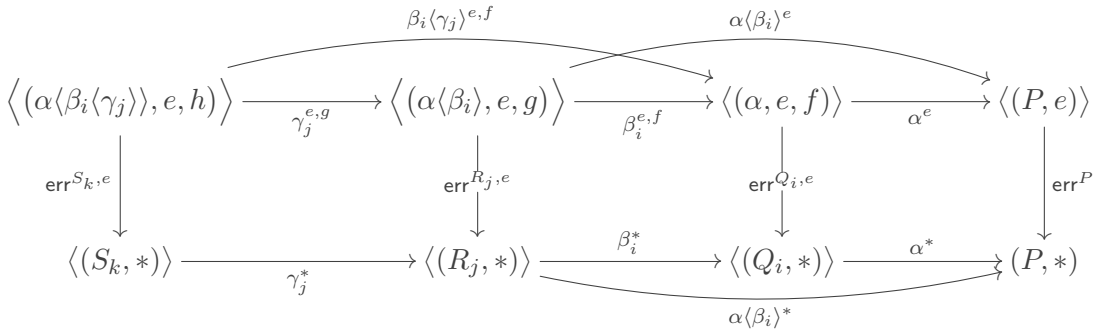


Figure 4: Objects, operations and equations of LSIFail.

to LSI. The two layers are connected by additional "vertical" operations that decompose generic errors into specific failure modes. "Diagonal" composites of horizontal and vertical operations are governed by a naturality condition $\mathsf{err}(\alpha) = \alpha(\mathsf{err})$ (suitably parameterized).

Definition 1. Given $\mathsf{Err} : \mathsf{LSI} \to \mathsf{Rel}$, the LSIFail operad is defined by the following data

$$\begin{aligned}
&\textbf{Objects:} && \Big\{(\alpha, e, f) \mid (e, f) \in \mathsf{Err}(\alpha)\Big\} \\
&\textbf{Operations:} && \beta^{e,f} : \Big\langle(\alpha\langle\beta\rangle, e, g) \mid (f, g) \in \mathsf{Err}(\beta)\Big\rangle \longrightarrow (\alpha, e, f) \\
&&& \mathsf{err}^{Q,e} : \Big\langle(\alpha, e, f) \mid (e, f) \in \mathsf{Err}(\alpha)\Big\rangle \longrightarrow (\mathrm{id}_Q, *, *) \\
&\textbf{Composition:} && \beta^{e,f}\langle\gamma^{e,g}\rangle = \Big(\beta\langle\gamma\rangle\Big)^{e,f} \\
&&& \beta^{*,*}\langle\mathsf{err}^{R,e}\rangle = \mathsf{err}^{Q,e}\langle\beta^{e,f}\rangle
\end{aligned}$$

Several special cases are indicated in Figure 4–generic errors $(P, *)$, a priori failure modes (P, e), operations α^*, α^e and err^P–all corresponding to the restriction $\alpha = \mathrm{id}_P$ (whence $e = f$ by functoriality of Err). These satisfy simplified versions of the composition relationships as indicated in the diagram.

Now consider a probabilistic model $\overline{\mathsf{fail}} : \mathsf{LSIFail} \to \mathsf{Prob}$. The map $P \mapsto (P, *)$ exhibits LSI as a full suboperad of LSIFail, so we can think of this as an extension of the original model fail. The operations and equations of LSIFail all have natural interpretations in these terms, shown in Table 2. Note that the need to represent relativized failures (α, e, f) is tied to the presence of summations in the last two equations.

6 Conclusion

We close by noting, briefly, some limitations of the present paper. Though our model is based on a real-world system, it is much too coarse to use in practice. A true predictive model would require a much more detailed decomposition of

Operations		Equations	
$\alpha^* \mapsto$	$\mathbf{pr}(Q_i\|P)$	$\alpha^*\langle\beta_i^*\rangle = \alpha\langle\beta_i\rangle^* \mapsto$	$\mathbf{pr}(R_j\|P) = \mathbf{pr}(R_j\|Q_i) \cdot \mathbf{pr}(Q_i\|P)$
$\alpha^e \mapsto$	$\mathbf{pr}(f\|e)$	$\alpha^e\langle\beta_i^{e,f}\rangle = \alpha\langle\beta_i\rangle^e \mapsto$	$\mathbf{pr}(g\|e) = \mathbf{pr}(g\|f,e) \cdot \mathbf{pr}(f\|e)$
$\beta^{e,f} \mapsto$	$\mathbf{pr}(g\|f,e)$	$\beta^{e,f}\langle\gamma^{e,g}\rangle = \Big(\beta\langle\gamma\rangle\Big)^{e,f} \mapsto$	$\mathbf{pr}(h\|f,e) = \mathbf{pr}(h\|g,e) \cdot \mathbf{pr}(g\|f,e)$
		$\mathsf{err}^P\langle\alpha^e\rangle = \alpha^*\langle\mathsf{err}^{Q_i,e}\rangle \mapsto$	
$\mathsf{err}^P \mapsto$	$\mathbf{pr}(e\|P)$	$\mathbf{pr}(f\|e) \cdot \mathbf{pr}(e\|P) = \sum_e \mathbf{pr}(f\|Q_i,e) \cdot \mathbf{pr}(Q_i\|P)$	
$\mathsf{err}^{Q,e} \mapsto$	$\mathbf{pr}(f\|Q,e)$	$\mathsf{err}^{Q,e}\langle\beta^{e,f}\rangle = \beta^*\langle\mathsf{err}^{R_j,e}\rangle \mapsto$	
		$\sum_e \mathbf{pr}(g\|f,e) \cdot \mathbf{pr}(f\|Q,e) = \sum_e \mathbf{pr}(g\|R_{ij},e) \cdot \mathbf{pr}(R_j\|Q)$	

Table 2: A probabilistic interpretation of the LSIFail operad.

the system. Similarly, the failure model presented in Section 4 is not based on real data, but rather selected to illustrate certain points. However, we intend to refine the model over time and, eventually, hope to produce a reference implementation for categorical systems modeling.

Second, the models presented here are purely static, but it would be preferable to incorporate sensor observations into our failure predictions. Here we should be able to leverage existing work on the interpretation of port-graphs as behavioral constraints on dynamical systems. Our next goal is to develop a dynamical model of system and component behaviors including both functioning and malfunctioning components. By substituting malfunctioning component models into the larger dynamical system, we can estimate the likely sensor readings for each failure mode and use these to assess the relativized failure probabilities discussed in Section 5.

Disclaimer: Commercial products are identified in this article to adequately specify the material. This does not imply recommendation or endorsement by the National Institute of Standards and Technology, nor does it imply the materials identified are necessarily the best available for the purpose.

References

[1] John S Beers and William B Penzes. The NIST length scale interferometer. *Journal of Research of the National Institute of Standards and Technology*, 104(3):225, 1999.

[2] David I Spivak. The operad of wiring diagrams: Formalizing a graphical language for databases, recursion, and plug-and-play circuits. *arXiv preprint arXiv:1305.0297*, 2013.

A LSI Operad

The appendix collects an explicit description of the LSI operad discussed throughout the paper. Interface types and boundaries are shown in Table 3. Operations, along with combinatorial descriptions of the associated port-graph architectures, and an equation are shown in Table 4.

LSI System Boundaries

Boundary	Ports
Intfr	laser, fringe
Chassis	laser, focus, drive
Optics	focus, intensity
Bath	heat, H_2O, setPt
Box	$heat_1$, $heat_2$, laser, temp
Lab	heat, temp
Mixer	mix
Resevoir	$heat_1$, $heat_2$, mix, H_2O
Heater	setPt, heat
TempSys	laser, H_2O, $temp_1$, $temp_2$, setPt
LengthSys	laser, intensity, fringe, drive
Sensors	intensity, fringe, $temp_1$, $temp_2$, focus, heat, laser
Actuators	H_2O, drive, setPt, focus, heat, laser
LSI	H_2O, intensity, fringe, $temp_1$, $temp_2$, setPt, drive

$$\mathsf{IType} = \left\{ \begin{array}{l} \texttt{heat}, \texttt{laser}, H_2O, \texttt{focus}, \texttt{mix}, \texttt{temp}, \\ \texttt{fringe}, \texttt{intensity}, \texttt{drive}, \texttt{setPt} \end{array} \right\}$$

Table 3: Objects and types for the LSI system operad, compiled from Figures 1, 2 and 3.

LSI System Architectures

φ:	$(ls : \texttt{LengthSys}, ts : \texttt{TempSys}) \longrightarrow \texttt{LSI}$
	$\ell.\texttt{laser} = t.\texttt{laser}$
λ:	$(ls : \texttt{Intfr}, op : \texttt{Optics}, ch : \texttt{Chassis}) \longrightarrow \texttt{LengthSys}$
	$ch.\texttt{focus} = op.\texttt{focus}$
	$ch.\texttt{laser} = ls.\texttt{laser}$
τ:	$(ba : \texttt{Bath}, bt : \texttt{Box}, rt : \texttt{Lab}) \longrightarrow \texttt{TempSys}$
	$bt.\texttt{heat}_1 = ba.\texttt{heat}$
	$bt.\texttt{heat}_2 = rt.\texttt{heat}$
κ:	$(sn : \texttt{Sensors}, ac : \texttt{Actuators}) \longrightarrow \texttt{LSI}$
	$s.\texttt{heat} = a.\texttt{heat}$
	$s.\texttt{laser} = a.\texttt{laser}$
	$s.\texttt{focus} = a.\texttt{focus}$
σ:	$(rt : \texttt{Lab}, bt : \texttt{Box}, op : \texttt{Optics}, ls : \texttt{Intfr}) \longrightarrow \texttt{LengthSys}$
	$bt.\texttt{heat}_2 = rt.\texttt{heat}$
	$bt.\texttt{laser} = ls.\texttt{laser}$
α:	$(ch : \texttt{Chassis}, ba : \texttt{Bath}) \longrightarrow \texttt{Actuators}$
β:	$(ht : \texttt{Heater}, mx : \texttt{Mixer}, rs : \texttt{Resevoir}) \longrightarrow \texttt{Bath}$
	$rs.\texttt{mix} = mx.\texttt{mix}$
	$rs.\texttt{heat}_2 = ht.\texttt{heat}$

Architectural Coherence Equation

$$\varphi(\lambda, \tau) = \kappa(\sigma, \alpha)$$

Table 4: Architectures (operations) from the LSI system operad. Each architecture can be described as a set of equations between component ports. The architectural coherence equation corresponds to the diagram in Figure 3.