arXiv:2010.08003v1 [cs.LO] 15 Oct 2020

Categorical Semantics of Cyber-Physical Systems Theory

Georgios Bakirtzis
University of Virginia
bakirtzis@virginia.edu

Cody H. Fleming
Iowa State University
flemingc@iastate.edu

Christina Vasilakopoulou
University of Patras
cvasilak@math.upatras.gr

ABSTRACT Cyber-physical systems require the construction and management of various models to assure their correct, safe, and secure operation. These various models are necessary because of the coupled physical and computational dynamics present in cyber-physical systems. However, to date the different model views of cyber-physical systems are largely related informally, which raises issues with the degree of formal consistency between those various models of requirements, system behavior, and system architecture. We present a category-theoretic framework to make different types of composition explicit in the modeling and analysis of cyber-physical systems, which could assist in verifying the system as a whole. This compositional framework for cyber-physical systems gives rise to unified system models, where system behavior is hierarchically decomposed and related to a system architecture using the systems-as-algebras paradigm. As part of this paradigm, we show that an algebra of (safety) contracts generalizes over the state of the art, providing more uniform mathematical tools for constraining the behavior over a richer set of composite cyber-physical system models, which has the potential of minimizing or eliminating hazardous behavior.

1 Introduction

In this paper we study the problem of unification between the disparate but necessary models used to assure correctness in cyber-physical systems (CPS), including requirements, system behaviors, and system architectures. Currently, these views are largely managed in an informal, piecemeal fashion, with no notion of formal traceability between different *types* of models, which could lead to designing and implementing systems that are ultimately unsafe due to inconsistencies between these three views. Model-based design attempts to address some of the aforementioned issues, but while it contains a notion of formal composition within each model view, it lacks a notion of formal composition *between* different

model types.

This need for formal compositional theories to support the design of CPS is a consistent theme in CPS literature both in the general modeling sense [12] and particularly in contract-based design [10, 51]. First, the model-based design of CPS can be greatly assisted by the composition of different *types* of models, which would provide traceability between the coupled physical and computational dynamics present in CPS [2]. Second, the application of formal composition makes precise abstraction and refinement, which are necessary in model-based design and analysis [60]. Third, by investing in a compositional modeling paradigm we are better able to identify unsafe or uncontrolled interactions between subsystems [62]. We posit that category theory and, specifically, the wiring diagram formalism [64] provide an appealing framework to build and analyze compositional models of CPS.

In the design and analysis of CPS, the word *composition* appears in many different contexts and may refer to different things. Category theory is one context where its meaning is formal and refers to something specific, namely the partial operation on morphisms of a category. However, we will herein also occasionally use the term *composite system* in its more relaxed sense, which we formalize categorically in this work using the systems-as-algebras framework. On the contrary, the term *compositionality* is not a formal one, rather the general characteristic of an analysis that ensures that the behavior of the whole is determined by the behavior of its building blocks.

Wiring diagrams are a particularly interesting example of the congruence between category theory and model-based design. Wiring diagrams have been independently created by category theorists [61, 66, 71] but surprisingly look and *feel* similar to engineering block diagrams used as the basis diagrammatic framework for modeling, for example, the unified modeling language (UML), the systems modeling language (SysML), and a variety of tools from Mathworks including Simulink. These types of diagrams are increasingly part of various research directions in CPS, for example the Ptolemy project [17] or Möbius [46]. Systems engineering is a discipline where diagrammatic reasoning has long been considered an important element in managing complexity. But several challenges persist, for example using SysML for the analysis of systems designs means a scarcity of simulation capabilities, an increased modeling effort to capture different views of the system, and the need to maintain all these differing views concurrently even as they evolve asynchronously. While the approach using wiring diagrams has little tool support currently, as an intellectual framework they overcome these limitations by augmenting this diagrammatic reasoning with stronger mathematical semantics.

In general, categorical semantics avoid modeling the internal structure of the objects they act upon. Instead, an object is perceived through its relationships with other objects and not – as is common with systems models – by what the object is individually. Indeed, in this context we focus on abstraction, which we see as determining *only* what is essential in each layer of a given model. This allows us to talk about how things are *related* instead

of focusing on how things *are*. This mindset as applied to systems theory gives rise to a circumspection of the system where we do not examine a system by its individual elements but by looking at the compositional structure of the system as a whole. This might sound familiar to safety experts where it is – arguably – accepted that we cannot examine how safe a system is by examining its individual constituents [44]. Instead, by modeling CPS in the wiring diagram framework we examine the system both by the individual constituents and their specific interconnections, compositionally.

Contributions. In this paper we use categories as a unifying modeling language for CPS:

- We develop a categorical semantics of compositional CPS theory that merges physical models with computational models for the design and analysis of CPS.[1]
- We formalize the general diagrammatic syntax of boxes and wires by adapting the systems-as-algebras model [64] for CPS, thereby producing a formal diagrammatic language for the design and analysis of CPS.
- We establish that the categorical and diagrammatic syntax equipped with a contracts algebra generalizes over the current state of the art [9].

As diagrammatic reasoning takes an increasingly central role in the modeling, simulation, and development of CPS, such relational semantics will become important in type checking, navigating different domains of abstraction, and ultimately assisting with providing evidence that CPS operate correctly during deployment. This requires an effort both from industry and academia to accept that visualization (usually the domain of industry) *and* mathematical rigor (usually the domain of academia) will be necessary to improve the current state of the art in system design. Wiring diagrams are one answer to this merger by implementing formal diagrammatic reasoning for CPS modeling and analysis.

2 Categorical background

In this section we present some essential categorical machinery that will be used to build up a formal compositional CPS theory.

2.1 A few basic categorical concepts

Briefly, a *category* $\mathbf{C}$ consists of a collection of objects $X, Y, \ldots, Z$ and a collection of arrows $f\colon X \to Y$, along with a composition rule

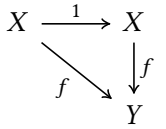
$$(f\colon X \to Y, g\colon Y \to Z) \mapsto g \circ f\colon X \to Z$$

and an identity arrow $1_X\colon X \to X$ for all objects, subject to associativity and unity conditions: $(f \circ g) \circ h = f \circ (g \circ h)$ and $f \circ 1_X = f = 1_Y \circ f$. This definition encompasses

[1] Compositional CPS theory is a flavor of what Lee calls computational dynamical systems theory [40].

a vast variety of structures in mathematics and other sciences: to name a few, **Set** is the category of sets and functions, whereas **Lin** is the category of k-linear (vector) spaces and k-linear maps between them, and we also have the category of states and transitions between them [23]. For a complete treatment of basic categorical concepts, consult Lawvere and Schanuel [38], Leinster [43], or Spivak [67].

A standard diagrammatic way to express composites is $X \xrightarrow{f} Y \xrightarrow{g} Z$ and equations via commutative diagrams of the following form



stands for $f \circ 1_X = f$

A morphism $f\colon X \to Y$ is called *invertible* or an *isomorphism* when there exists another $g\colon Y \to X$ such that $f \circ g = 1_Y$ and $g \circ f = 1_X$.

A *functor* $F\colon \mathbf{C} \to \mathbf{D}$ between two categories consists of a function between objects and a function between morphisms, where we denote $Ff\colon FX \to FY$, such that it preserves composition and identities: $F(f \circ g) = Ff \circ Fg$ and $F(1_X) = 1_{FX}$. A functor can informally be thought of as a structure preserving map between domains of discourse. Interestingly, categories and functors form a category on their own, denoted **Cat**, in the sense that functors compose and the rest of the axioms hold.

A *monoidal* category $\mathbf{V}$ is a category that comes equipped with a functor called 'tensor product'

$$\otimes\colon \mathbf{V} \times \mathbf{V} \to \mathbf{V}$$

which can be thought of as multiplication of objects and morphisms, or more broadly as doing operations in parallel. The tensor product comes with invertible morphisms $(X \otimes Y) \otimes Z \cong X \otimes (Y \otimes Z)$ meaning that it is associative up to isomorphism. There is also a distinguished object $I \in \mathbf{V}$ with $I \otimes X \cong X \cong X \otimes I$, acting like an identity for this multiplication. All this data satisfy certain axioms found, for example, in Joyal and Street [35], that are beyond the scope of this paper.

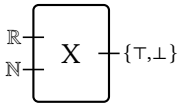
Widely used examples of monoidal categories include $(\mathbf{Set}, \times, \{*\})$ with the cartesian product of sets and the singleton, as well as $(\mathbf{Lin}, \otimes_k, k)$ with the tensor product of k-vector spaces. Moreover, $(\mathbf{Cat}, \times, \mathbf{1})$ with the cartesian product of categories (similarly to that of sets) and the unit category with a single object and single arrow forms a monoidal category. In fact, all these are examples of *symmetric* monoidal categories, which come further equipped with isomorphisms $X \otimes Y \cong Y \otimes X$, for example, for two sets $X \times Y \cong Y \times X$ via the mapping $(x, y) \mapsto (y, x)$.

A *lax monoidal* functor between two monoidal categories $F\colon (\mathbf{V}, \otimes_\mathbf{V}, I_\mathbf{V}) \to (\mathbf{W}, \otimes_\mathbf{W}, I_\mathbf{W})$ is a functor that preserves the monoidal structure in a lax sense (meaning not up to isomorphism). Explicitly, it comes equipped with collections of morphisms, the 'laxator' $\phi_{X,Y}\colon FX \otimes_\mathbf{W} FY \to F(X \otimes_\mathbf{V} Y)$ and the 'unitor' $\phi_0\colon F(I_\mathbf{V}) \to I_\mathbf{W}$ that express the relation

between the image of the tensor and the tensor of the images inside the target category **W**; these also adhere to certain axioms [35]. Monoidal categories and lax monoidal functors also form a category of their own, denoted $\mathbf{MonCat}_{\text{lax}}$.

2.2 The category W of wiring diagrams

The cornerstone of this work is the category **W** of *labeled boxes* and *wiring diagrams*. Informally, the objects of this category are to be thought of as empty placeholders for processes, so far only specifying the types of the input and output data that they may receive. For example, an object X is diagrammatically depicted as

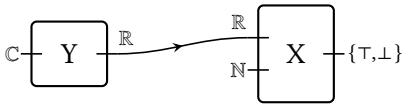


A process that can later be positioned inside this box is, for example, the function

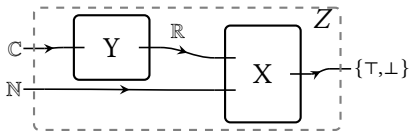
$$f(r,n) = \begin{cases} \top & \text{if } r = n \\ \bot & \text{if } r \neq n \end{cases}.$$

To begin with, however, these boxes are uninhabited: they merely represent the architecture of a possible system. The two input wires above can be represented by a single wire typed $\mathbb{R} \times \mathbb{N}$.

These interfaces, with finitely many input and output wires along with their associated types, are essentially the building blocks for forming larger interfaces from smaller ones, and this is what is captured by the morphisms in the category **W**. For example, suppose $\mathbb{C} \to \boxed{Y} \to \mathbb{R}$ is another box. Intuitively, since the type of the output wire of Y matches the type of one of the input wires of X, they could be linked along that wire



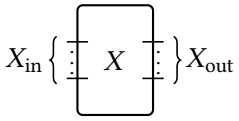
to provide a new interface that receives two inputs, one complex and one natural number, and outputs a true or false:



While combining interfaces together, we want to be able to express not only the new interface they form, which in the above example is $\begin{matrix}\mathbb{C}\\ \mathbb{N}\end{matrix} \to \boxed{Z} \to \{\top,\bot\}$, but also keep track of the

intermediate wires. In our envisioned category, this will be expressed as a morphism from 'X and Y' into Z.[2]

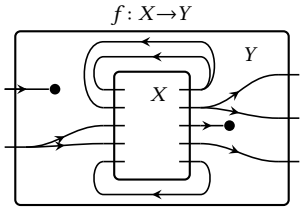
Definition 1. There is a category **W** with pairs of sets $X = (X_{\text{in}}, X_{\text{out}})$ as objects, thought of as the products of types of the input and output ports of an empty box as in



A morphism $f : X \to Y$ in this category is a pair of functions[3]

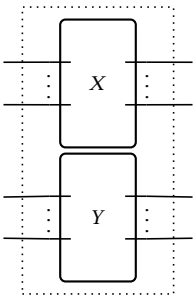
$$\begin{cases} f_{\text{in}} : X_{\text{out}} \times Y_{\text{in}} \to X_{\text{in}} & (a) \\ f_{\text{out}} : X_{\text{out}} \to Y_{\text{out}} & (b) \end{cases} \tag{1}$$

thought of as providing the flow of information in a picture as follows



which illustrates in diagrammatic view the system of equations 1 (where the forks correspond to duplication and black bullets correspond to discarding). Information going through those wires can be anything insofar as the types match between ports. The wires of the external input ports Y_{in} can only go to the internal input ports X_{in} (equation 1a), whereas the wires of the internal output ports X_{out} can either be directed to the external output ports Y_{out} (equation 1b) or fed back to the internal input ports X_{in} (equation 1a).

This is a monoidal category, where the tensor product of any two labelled boxes X and Y is $X \otimes Y = (X_{\text{in}} \times Y_{\text{in}}, X_{\text{out}} \times Y_{\text{out}})$ that represents the parallel placement of the two

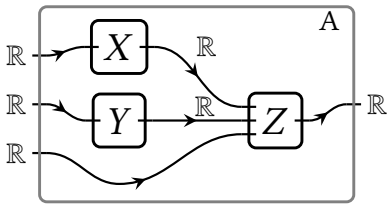


(2)

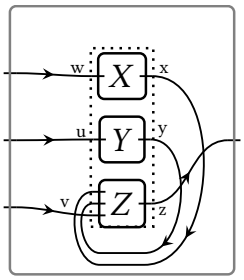
with input and output the (cartesian) product of the respective sets.

[2]This morphism is explicitly given later in Section 3.3.

[3]In reality, these are not just arbitrary functions, rather generated by projections, diagonals and switchings; for more details consult Spivak [65, Def. 3.3].



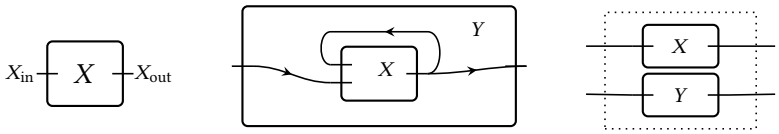
(a) A zoomed-in picture of a process A.



(b) View as a wiring diagram.

Figure 1: An example wiring diagram as a morphism in the category **W**.

For simplicity, we often abstract the pictures for objects, morphisms and tensor in **W** to



The composition in this category zooms two levels deep, and is formally defined as follows: for $f = (f_{\text{in}}, f_{\text{out}}) \colon X \to Y$ and $g = (g_{\text{in}}, g_{\text{out}}) \colon Y \to Z$ as in the systems of equations 1, the new wiring diagram $g \circ f \colon X \to Z$ consists of the functions $\Big((g \circ f)_{\text{in}} \colon X_{\text{out}} \times Z_{\text{in}} \to X_{\text{in}}, (g \circ f)_{\text{out}} \colon X_{\text{out}} \to Y_{\text{out}}\Big)$ given by

$$(g \circ f)_{\text{in}}(x', z) = f_{\text{in}}(x', g_{\text{in}}(f_{\text{out}}(x'), z))$$
$$(g \circ f)_{\text{out}}(x') = g_{\text{out}}(f_{\text{out}}(x')).$$

The identity morphism on X is $(\pi_2 \colon X_{\text{out}} \times X_{\text{in}} \to X_{\text{in}}, 1_{X_{\text{out}}} \colon X_{\text{out}} \to X_{\text{out}})$, and the axioms of a category hold.[4] Moreover, the monoidal unit is the box $\{*\} \to \boxed{I} \to \{*\}$ and the axioms of a monoidal category can also be verified to hold [71].

The category **W** as defined above is really **Set**-*typed* or *labeled*, namely the objects and morphisms are described using sets. However, the formalism allows to label the wires with any category equipped with finite products instead of $(\mathbf{Set}, \times, \{*\})$. For example, the types could be in linear spaces $\mathbb{R}^n$ or topological spaces (X, τ) or even more general time-related categories like lists of signals expressed as sheaves on real-time intervals [64, § 3]. Not only do these different types accommodate systems with such inputs and outputs, but also often provide a passage between different models on the same system by functorially changing the types.

The construction of this category allows us to formally give meaning to arbitrary wiring diagram pictures and as a result, coherently describe interconnections. As an example, consider three processes X, Y, and Z (Fig. 1a). The involved labelled boxes are $X = (\mathbb{R}, \mathbb{R})$, $Y = (\mathbb{R}, \mathbb{R})$ and $Z = (\mathbb{R}^3, \mathbb{R})$, which connected in the depicted way form

[4] There is a strong relation between **W** and the category of *lenses* [28], as well as the *Dialectica* category [21].

the composite interface $A = (\mathbb{R}^3, \mathbb{R})$. Although A's inputs and outputs are to the 'outside world', they could also potentially interconnect to other boxes themselves.

To implement the above as a morphism in the category **W**, we first 'align' the boxes such that the wires follow their input and output (Fig. 1b), which then forms a morphism from the tensor product of the three boxes $X \otimes Y \otimes Z$ (the dotted box) with input $\mathbb{R}^5$ and output $\mathbb{R}^3$, to the outside box $A = (\mathbb{R}^3, \mathbb{R})$ with explicit description

$$\begin{cases} f_{\text{in}} : \overbrace{\mathbb{R}\times\mathbb{R}\times\mathbb{R}}^{(X\otimes Y\otimes Z)_{\text{out}}} \times \overbrace{\mathbb{R}\times\mathbb{R}\times\mathbb{R}}^{A_{\text{in}}} \to \overbrace{\mathbb{R}\times\mathbb{R}\times\mathbb{R}\times\mathbb{R}\times\mathbb{R}\times\mathbb{R}}^{(X\otimes Y\otimes Z)_{\text{in}}}, & (x,y,z,w,u,v) \mapsto (w,u,x,y,v) \\ f_{\text{out}} : \underbrace{\mathbb{R}\times\mathbb{R}\times\mathbb{R}}_{(X\otimes Y\otimes Z)_{\text{out}}} \to \underbrace{\mathbb{R}}_{A_{\text{out}}}, & (x,y,z) \mapsto z \end{cases}. \tag{3}$$

The two functions, f_{in} and f_{out}, specify which wires are connected to which; f_{in} maps the three internal outputs x, y, z together with the external inputs w, u, v to the internal inputs, in the order determined by our alignment[5], and f_{out} projects out of the three internal outputs x, y, z the third one z.

To sum up, the category **W** provides a formal way of mathematically expressing any configuration at hand, with sole focus on the interconnection of vacant building blocks.

3 Compositional Cyber-Physical Systems Theory

Assessing the correct behavior of CPS requires several model views. Before discussing them, we must first clarify the meaning of the terminology that we will use. We choose to use the terminology of *requirements*, *system behavior*, and *system architecture* to describe the different diagrammatic abstractions of CPS models. We define requirements as constraints over system behavior and system architecture. By system behavior we mean models of the form of automata or state space models. By system architecture we mean models of candidate implementations that, in the case of CPS, include hardware and software for the embedded system portion of CPS and motors, control surfaces, and mechanical structure for the physical portion of the CPS. In the following formalism in general, we will view the individual diagram pictures as architecture, and the particular semantics that go into the boxes within this diagram as behavior, omitting the leading word 'system' when only discussing about the diagrammatic representation. Contracts that constraint both behavior and architecture in this sense will represent a subset of system safety requirements.

The categorical approach has the advantage of providing a *compositional* modeling and analysis, in which the composite system is completely and uniquely determined from its subsystems and their interconnections. This is achieved through the implementation of the formalism in two parts. The first is a behavior algebra that allows the hierarchical

[5]We could choose a different alignment of the internal boxes, which would result to a different, but essentially equivalent, pair of functions. This would not affect our analysis.

modeling between the abstraction of system behavior and system architecture, in a *zoom-in, zoom-out* approach [72], where each view may have distinct inputs and outputs. The second is a contract algebra that applies constraints as defined in requirements over the behavior algebra.

An analogously high-level approach using monoidal categories and compositional techniques has already found success in categorical quantum mechanics [1, 18], where it has become the de facto language to describe and manipulate quantum processes diagrammatically. We posit that a similar innovation should take place in the design and assessment of safety-critical CPS, due to the concerns raised by the intertwined nature of digital control with physical processes and the environment. We will view distinct but related system models, pertinent to assuring the correct behavior of CPS, as *algebras* of the monoidal category of wiring diagrams.

The wiring diagram approach diverges from input-output models. While the diagrammatic syntax looks similar to such models, what is contained within the boxes need not be a mathematical function. It can instead be any sort of process, from very concrete descriptions like automata, to more abstract processes which could be deterministic or nondeterministc, to mere requirements of a mathematically unknown formula. Similarly, the arrows do not need to contain one piece of information, for example the input and output of a function; rather, arrows can carry arbitrary objects of a chosen category of types. Previous compositional modeling methods for CPS are often limited to sets and functions or in the most general sense, relations. However, the state space of a controls system need not be the set $\mathbb{R}$, but could instead be a topological space like the line or circle $\mathbb{S}$. The rich interplay between topology and category theory positions category theory as a particularly good candidate for modeling dynamics, for example see Hansen and Ghrist [32] or earlier, in the more related area of hybrid systems, Ames [5] and Tabuada et al. [69].

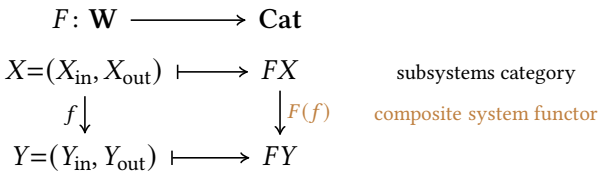
We now develop the formalism for the three system views necessary to assess the correct behavior of CPS: system behavior, system architecture, and (a subset of) requirements.

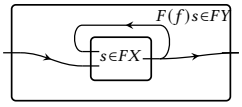
3.1 System behavior via algebras on the category W

The category of wiring diagrams does not populate the boxes with actual systems, for example, dynamical systems (Section 2.2). This is instead done by developing extra structure on top of it. By knowing the configuration of the component systems, the composite system can then be uniquely determined.

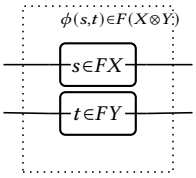
Categorically, this is described as an *algebra* on $\mathbf{W}$, namely a lax monoidal functor $F\colon (\mathbf{W}, \otimes, I) \to (\mathbf{Cat}, \times, \mathbf{1})$. The idea is that each algebra assigns to a box $X = (X_{\text{in}}, X_{\text{out}})$ a category FX of systems that can be placed in the box, and also assigns to a wiring diagram $f = (f_{\text{in}}, f_{\text{out}})$ a functor $Ff\colon FX \to FY$ that, given a system s inhabiting the internal box

of a wiring diagram, produces the *composite system* $F(f)(s)$ inhabiting the external box.





Intuitively, the object assignment FX and FY gives semantics to arbitrary boxes through the subsystems category while the composite system functor Ff assembles the composite operations of the overall system behavior. Moreover, the monoidal structure of the functor via the laxator $\phi_{X,Y}\colon FX \times FY \to F(X \otimes Y)$ ensures that for given systems inside parallely placed boxes, we can always determine a system inhabiting their tensor product



The categorical formulation allows us to use a number of algebras according to our purposes. Below we describe two such algebras of discrete dynamical systems, and later we will examine the algebra of contracts (Section 3.3). There exist also other algebras, describing systems behaviors that are not like difference equations. For example, algebras for abstract total or deterministic machines [64].

The diagrammatic representation via wiring diagrams for system modeling and analysis is rather straightforward, particularly because wiring diagrams are similar to engineering block diagrams and, hence, the visual syntax is equivalent to existing CPS design tools. However, the current diagrammatic representation is mathematically richer and more concrete – it also accounts for actual composition computations as we will see below. Another important factor specifically for CPS is the richness of other possible algebras or semantics that one can develop and assign in these boxes using as backing the notion of the monoidal category. As CPS become more complex, cooperative, and coordinated these functorial semantics can give formal relations between several concepts important in modeling and assurance of safe CPS [7].

3.1.1 Moore machines

As an illustrative example on how to develop and use the behavior algebra on an architecture in **W**, we will position the familiar Moore machines inside the boxes X, Y and Z of (Fig. 1a). This is a simple yet useful demonstration of the algebra machinery because Moore machines model discrete dynamical systems. To concretely describe the systems composite, we first need to verify that Moore machines form a **W**-algebra. Indeed, there

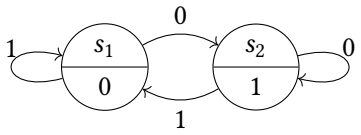
is a monoidal functor

$$\mathcal{M}\colon \mathbf{W} \to \mathbf{Cat}$$

which maps each $(X_{\text{in}}, X_{\text{out}})$ to the category $\mathcal{M}(X_{\text{in}}, X_{\text{out}})$ where

- objects are triples (S, u, r) where S is the *state space* set, $u\colon S \times X_{\text{in}} \to S$ is the *update function* and $r\colon S \to X_{\text{out}}$ is the *readout function*;
- morphisms $(S, u, r) \to (S', u', r')$ are functions $f\colon S \to S'$ between the state spaces that commute with the update and readout functions, namely $f(u(s, x)) = u'(fs, x)$ and $f(r(s)) = r'(fs)$.

Hence, $\mathcal{M}(X_{\text{in}}, X_{\text{out}})$ this is the category of Moore machines with fixed input and output alphabet X_{in} and X_{out} respectively. For example, an object of the category $M(\{0, 1\}, \{0, 1\})$ with inputs and outputs the booleans, is the '`not`' finite state machine



with state space $S = \{s_1, s_2\}$ and update and readout functions depicted in the above state diagram, for example, $u(s_1, 0) = s_2$ (middle top edge) and $r(s_2) = 1$ (bottom part of s_1-node).

Having defined the categories of systems that can inhabit boxes in wiring diagram pictures for this specific Moore machine model, we proceed to define the composite system functor $\mathcal{M}(f)\colon \mathcal{M}X \to \mathcal{M}Y$, given a wiring diagram $f = (f_{\text{in}}, f_{\text{out}})\colon X \to Y$. Explicitly, this functor maps a Moore machine (S, u, r) with input and output $X_{\text{in}}, X_{\text{out}}$ to a Moore machine (S, u', r') with input and output $Y_{\text{in}}, Y_{\text{out}}$ having the *same* state space S, but with new update and readout functions formed as follows

$$\begin{aligned} u'\colon Y_{\text{in}} \times S \to S, \qquad & u'(y, s) = u(f_{\text{in}}(y, r(s)), s) \\ r'\colon S \to Y_{\text{out}}, \qquad & r'(s) = f_{\text{out}}(r(s)) \end{aligned} \tag{4}$$

Finally, we need to specify the monoidal structure of $\mathcal{M}$ by providing functors $\mathcal{M}(X) \times \mathcal{M}(Y) \to \mathcal{M}(X \otimes Y)$. Explicitly, given two Moore machines $(S_X, u_X\colon X_{\text{in}} \times S_X \to S_X, r_X\colon S_X \to X_{\text{out}})$ and $(S_Y, u_Y\colon Y_{\text{in}} \times S_Y \to S_Y, r_Y\colon S_Y \to Y_{\text{out}})$, we construct a new Moore machine with space set $S_X \times S_Y$ and update and readout functions

$$\begin{aligned} u\colon X_{\text{in}} \times Y_{\text{in}} \times S_X \times S_Y \to S_X \times S_Y, \qquad & u(x, y, s, t) = (u_X(x, s), u_Y(y, t)) \\ r\colon S_X \times S_Y \to X_{\text{out}} \times Y_{\text{out}}, \qquad & r(s, y) = (r_X(s), r_Y(t)) \end{aligned} \tag{5}$$

It can been be verified that with the above assignments, Moore machines satisfy the axioms of a wiring diagram algebra [64, §2.3]. We can therefore arbitrarily interconnect

such systems, in particular as in Fig. 1a, and produce a new such system with a description only terms of its components and their wiring. Suppose we have Moore machines in the boxes $\boxed{X}$, $\boxed{Y}$, $\boxed{\boxed{Z}}$, all with $\mathbb{R}$-valued wires, with state spaces S_X, S_Y and S_Z and update and readout functions respectively as in

$$\begin{cases} S_X \times \mathbb{R} \xrightarrow{u_X} S_X \\ S_X \xrightarrow{r_X} \mathbb{R} \end{cases} \quad \begin{cases} S_Y \times \mathbb{R} \xrightarrow{u_Y} S_Y \\ S_Y \xrightarrow{r_Y} \mathbb{R} \end{cases} \quad \begin{cases} S_Z \times \mathbb{R}^3 \xrightarrow{u_Z} S_Z \\ S_Z \xrightarrow{r_Z} \mathbb{R}. \end{cases}$$

The algebra machinery (4) and (5) for the specific wiring diagram (3) produces the composite Moore machine which inhabits the outer box $\boxed{\boxed{A}}$ with state space $S_X \times S_Y \times S_Z$, readout function $r\colon S_X \times S_Y \times S_Z \to \mathbb{R}$ given by $(s,t,p) \mapsto r_Z(p)$ and update function $S_X \times S_Y \times S_Z \times \mathbb{R}^3 \to S_X \times S_Y \times S_Z$ given by

$$(s,t,p,w,u,v) \mapsto (u_X(s,w), u_Y(t,u), u_Z(p, r_X(s), r_Y(t), v))\,.$$

In general, the composite system is produced using the algebra machinery, no matter how complicated the systems or the wiring diagram is; given any interconnection, the monoidal functor will determine a result. Therefore, this functoriality alleviates some of the scalability issues present in other formalisms.

3.1.2 Linear time-invariant systems

There is a sub-algebra of the algebra of Moore machines, for *linear time-invariant systems* (LTIS) or linear discrete dynamical systems per Spivak [65]. In fact, the Moore machines model is an algebra of $\mathbf{W}_{\mathbf{Set}}$, where the types of wires are sets and the wiring diagrams are given by functions, whereas the LTIS model is an algebra of $\mathbf{W}_{\mathbf{Lin}}$, where the types are given by **Lin**, the category of linear spaces and linear maps.

Explicitly, there is a monoidal functor $\mathcal{L}\colon \mathbf{W}_{\mathbf{Lin}} \to \mathbf{Cat}$ that assigns to any box $X_{\text{in}}\ \boxed{\ }\ X_{\text{out}}$ a category $\mathcal{L}(X_{\text{in}}, X_{\text{out}})$ of systems $(S, u\colon S \times X_{\text{in}} \to S, r\colon S \to X_{\text{out}})$ like before, but where all $S, X_{\text{in}}, X_{\text{out}}$ are linear spaces and both update and readout functions u and r are linear functions expressed as

$$\begin{aligned} u(s,x) &= \mathscr{A} \cdot s + \mathscr{B} \cdot x = \begin{pmatrix} \mathscr{A} & \mathscr{B} \end{pmatrix} \begin{pmatrix} s \\ x \end{pmatrix} \\ r(s) &= \mathscr{C} \cdot s \end{aligned}$$

where $\mathscr{A}$, $\mathscr{B}$ and $\mathscr{C}$ are matrices of appropriate dimension. For example, if the input, out-

put and state spaces are $X_{\text{in}} = \mathbb{R}^k$, $X_{\text{out}} = \mathbb{R}^\ell$ and $S = \mathbb{R}^n$, then

$$\begin{cases} \mathscr{A} \in {}_nM_n & \text{represents a linear transformation } \mathbb{R}^n \to \mathbb{R}^n \\ \mathscr{B} \in {}_nM_k & \text{represents a linear transformation } \mathbb{R}^k \to \mathbb{R}^n \\ \mathscr{C} \in {}_\ell M_n & \text{represents a linear transformation } \mathbb{R}^n \to \mathbb{R}^\ell. \end{cases} \tag{6}$$

Now given an arbitrary wiring diagram $f = (f_{\text{in}}, f_{\text{out}}) \colon (X_{\text{in}}, X_{\text{out}}) \to (Y_{\text{in}}, Y_{\text{out}})$ as formalized in the system of equations (1), where for $Y_{\text{in}} = \mathbb{R}^{k'}$ and $Y_{\text{out}} = \mathbb{R}^{\ell'}$ both linear functions of the wiring diagram are also expressed as corresponding matrices $f_{\text{in}} = \begin{pmatrix} {}_k(\mathscr{A}^f)_\ell & {}_k(\mathscr{B}^f)_{k'} \end{pmatrix}$ and $f_{\text{out}} = {}_{\ell'}\mathscr{C}^f_\ell$, the functor $\mathcal{L}(f)$ maps some system $(S, \mathscr{A}, \mathscr{B}, \mathscr{C})$ in $\mathbb{R}^k \boxed{X} \mathbb{R}^\ell$ to the system

$$(S, \mathscr{A} + \mathscr{B} \cdot \mathscr{A}^f \cdot \mathscr{C},\ \mathscr{B} \cdot \mathscr{B}^f,\ \mathscr{C}^f \cdot \mathscr{C}) \tag{7}$$

in $\mathbb{R}^{k'} \boxed{Y} \mathbb{R}^{\ell'}$. The earlier-used term *sub-algebra* precisely means that this formula is a special case of equation (4) when the functions involved are of this specific form.

Finally, the monoidal structure of this assignment $\mathcal{L} \colon \mathbf{W}_{\mathbf{Lin}} \to \mathbf{Cat}$ is given by functors $\mathcal{L}(X) \times \mathcal{L}(Y) \to \mathcal{L}(X \otimes Y)$ that map any two such systems $(S_X, \mathscr{A}_X, \mathscr{B}_X, \mathscr{C}_X)$ and $(S_Y, \mathscr{A}_Y, \mathscr{B}_Y, \mathscr{C}_Y)$ inhabiting parallel boxes as in wiring diagram (2) give rise to a parallel composite system

$$\left(S_X \times S_Y, \begin{pmatrix} \mathscr{A}_X & 0 \\ 0 & \mathscr{A}_Y \end{pmatrix}, \begin{pmatrix} \mathscr{B}_X & 0 \\ 0 & \mathscr{B}_Y \end{pmatrix}, \begin{pmatrix} \mathscr{C}_X & 0 \\ 0 & \mathscr{C}_Y \end{pmatrix}\right).$$

3.1.3 Functions (as a non-example)

If we would like to populate the boxes of a wiring interconnection with mathematical functions, namely assign to some $X_{\text{in}} \boxed{X} X_{\text{out}}$ a function $h \colon X_{\text{in}} \to X_{\text{out}}$, there is no natural way to make this assignment into an algebra $\mathbf{W} \to \mathbf{Cat}$. The main reason this fails is the existence of the feedback loop.

However, we can incorporate functions into other existing models, for example Moore machines. It is possible to express a function $h \colon X_{\text{in}} \to X_{\text{out}}$ as an object of $\mathcal{M}(X_{\text{in}}, X_{\text{out}})$, with state space the domain X_{in} and update and readout functions $\pi_2 \colon X_{\text{in}} \times X_{\text{in}} \to X_{\text{in}}$ projecting the second variable and $h \colon X_{\text{in}} \to X_{\text{out}}$ applying the said function. The resulting finite state machine at each round replaces the old input with the new input, and outputs the function application on it. Analogously, a linear function can be viewed as a linear time-invariant system if we set $\mathscr{A} = 0$ the zero matrix, $\mathscr{B} = I$ the unit matrix and $\mathscr{C} = h$ the matrix represents the given linear transformation.

As a result, functions can be indeed used to populate boxes, and wired with other functions or Moore machines they produce a composite Moore machine using the algebra $M \colon \mathbf{W} \to \mathbf{Cat}$. It is also the case that sometimes, wiring two functions using the Moore

machine algebra machinery, we end up with another function and not a more general Moore machine – this usually happens in serial-like wirings without loops.

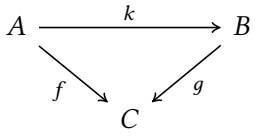
Summarizing this section, the starting point is the category of wiring diagrams $\mathbf{W}$ with no processes inside the boxes. We can then assign the behavior of Moore machines inside the boxes using the corresponding $\mathbf{W}$-algebra $\mathcal{M}$, or the behavior of linear discrete dynamical systems using the sub-algebra of linear time-invariant systems $\mathcal{L}$, which recovers the standard model of state-space representation in modern control albeit with a slightly different syntax. By composing behaviors using the latter, we recover a block-diagonal state space model, a useful representation for modeling the control portion of CPS.

To model state-space representations we had to develop all the above categorical machinery. However, the point of modelling algebraically is that now we can ensure composition and also caution when two boxes do not compose in the strict mathematical sense using only the diagrammatic syntax, which is familiar to develop and manipulate. At the moment we have developed the theory using pen and paper, but it is possible to produce an algorithmic implementation of this algebra machinery and then enforce these rules diagrammatically.

3.2 System architecture via hierarchical decomposition

Starting with a CPS from a designer point of view, we now might want to model a candidate system architecture. In general, decomposing a CPS in certain sub-components and using a specific wiring between them follows some choices based on the physical reality, experience, purpose and access to particular components at the time. Having formalized an agnostic process interface where various descriptions could live on as an object in the category of wiring diagrams $\mathbf{W}$, as well as arbitrary zoomed-in pictures of a system as a morphism in $\mathbf{W}$, we have now access to all necessary tools to realize the above system architecture design process using the general notion of a *slice category*.

For any category $\mathbf{C}$ and a fixed object $C \in \mathbf{C}$, the slice category $\mathbf{C}/C$ has as objects $\mathbf{C}$-morphisms with fixed target C, for example $f\colon A \to C, g\colon B \to C, \ldots$. The arrows in that category from some f to some g are $\mathbf{C}$-morphisms $k\colon A \to B$ between the domains, making the formed triangle



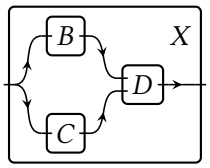
commute, namely $g \circ k = f$. This data forms a category, which also illustrates the abstract nature of the initial category definition in Section 2.1: objects and arrows can be of any sort (in this case objects are morphisms of a certain shape in some fixed category, and arrows are also morphisms that satisfy a property) as long as they satisfy the axioms of a category.

For our wiring diagram category $\mathbf{W}$, where a morphism $f\colon X \to Y$ can be thought of

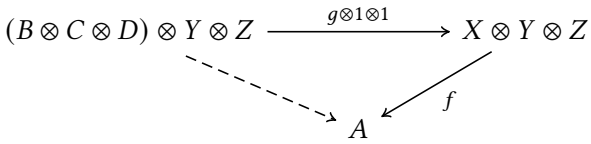
as an implementation of an interface Y into sub-interface(s) X wired in a specific manner, the slice category $\mathbf{W}/Y$ of all arrows mapping into the chosen object Y essentially contains all possible design choices available to a system engineer. This formally captures the possibility of implementing a system in multitudes of ways.

Concretely, suppose we have a system with $\mathbb{R}^3$-inputs and $\mathbb{R}$-outputs, namely inhabiting a box $\begin{smallmatrix}\mathbb{R}\\ \mathbb{R}\\ \mathbb{R}\end{smallmatrix}\!\!\boxed{A}\!\!-\mathbb{R}$. How can we decompose it into sub-processes, and how can they be interconnected to form the given system? All the possible decompositions can thus be thought of as the objects of the slice category $\mathbf{W}/A$. For example, (Fig. 1a) depicts one of these choices, namely the specific wiring diagram $f : X \otimes Y \otimes Z \to A$.

Now suppose we make another implementation choice to further decompose the box X as in



meaning we choose a specific wiring diagram $g : B \otimes C \otimes D \to X$. This constitutes another level of zoom-in for the process in A, at least for the subcomponent X, depicted as in (Fig. 2). Categorically, this is a picture of the composite morphism $f \circ (g \otimes 1 \otimes 1)$, the dashed arrow



where the top arrow employs the morphism g as the implementation of X and identity morphisms on Y and Z (as trivial implementations), and f is the earlier A-implementation of (Fig. 1a). In the end, we can disregard the borders of the interface X and map directly from the subcomponents $B \otimes C \otimes D \otimes Y \otimes Z$ to A without passing through X at all if desired. As a result, we are free to use hierarchical decomposition of processes for any sub-component (or for many simultaneously) and each time, these architectural choices add one more composite morphism to the resulting wiring diagram that expresses an implementation of the outmost system process.

3.3 System requirements via contracts

The concept of a *contract*, fundamental for this work, is another example of an algebra for the monoidal category of labeled boxes and wiring diagrams $\mathbf{W}$. In detail, for any labeled box $X = (X_{\text{in}}, X_{\text{out}})$, a contract is defined to be a relation

$$R \subseteq X_{\text{in}} \times X_{\text{out}}$$

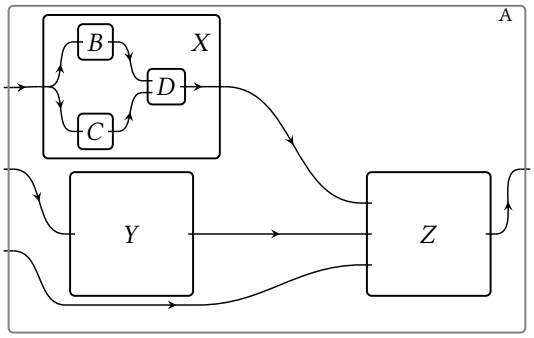


Figure 2: A two-level zoomed-in picture of a process A.

expressing the *allowable* tuples of input and output behaviors of the process. Such a description is one among the most widespread abstract systems modeling notions, see for example Mesarovic and Takahara [48, §2]. We make a distinction between the explicit defining process of a system; that is, the behavior assigned to a wiring diagram, and the system behavior. However, abstractly a system *is* its behavior and therefore modeling a system in the wiring diagram paradigm makes those two notions equivalent. The distinction is however useful for separating the *behavior* algebra from the *contracts* algebra, which are formally related but can be used independently of each other.

3.3.1 Static contracts

The algebra of *static contracts* is a slight variation of the algebra originally presented in [64, §4.5]. Explicitly, the functor $C\colon \mathbf{W} \to \mathbf{Cat}$ bound to express conditions on inputs and outputs in a time-less manner, assigns to a box $X_{\text{in}} \boxed{X} X_{\text{out}}$ the category $C(X_{\text{in}}, X_{\text{out}})$ of binary relations, that is, subsets $i\colon R \hookrightarrow X_{\text{in}} \times X_{\text{out}}$, with morphisms $f\colon R \to P$ being subset inclusions of the form

$$\begin{array}{ccc} R & \hookrightarrow & X_{\text{in}} \times X_{\text{out}} \\ {\scriptstyle f}\big\downarrow & \nearrow & \\ P & & \end{array}$$

For a given contract $R_X \subseteq X_{\text{in}} \times X_{\text{out}}$ and a wiring diagram ($f_{\text{in}}\colon X_{\text{out}} \times Y_{\text{in}} \to X_{\text{in}}$, $f_{\text{out}}\colon X_{\text{out}} \to Y_{\text{out}}$), the application of the functor $C(f)$ on R_X is the contract $R_Y \subseteq Y_{\text{in}} \times Y_{\text{out}}$ described by

$$R_Y = \{(y_1, y_2) \in Y_{\text{in}} \times Y_{\text{out}} \mid \exists x_2 \in X_{\text{out}} \text{ such that } (f_{\text{in}}(x_2, y_1), x_2) \in R_X \text{ and } f_{\text{out}}(x_2) = y_2\} \tag{8}$$

For an explicit description of how this formula arises categorically, see appendix B. In various examples, this composite contract may be expressed in more elementary terms depending on the form of the component contracts R_X and the wiring diagram at hand.

For the monoidal structure of the functor, suppose we have two parallel boxes (2) with

contracts $R_X \subseteq X_{\text{in}} \times X_{\text{out}}$ and $R_Y \subseteq Y_{\text{in}} \times Y_{\text{out}}$. The laxator $\phi_{X,Y} \colon C(X) \times C(Y) \to C(X \otimes Y)$ induces a contract on the box $(X_{\text{in}} \times Y_{\text{in}}, X_{\text{out}} \times Y_{\text{out}})$ which is merely the cartesian product

$$R_X \times R_Y \hookrightarrow X_{\text{in}} \times X_{\text{out}} \times Y_{\text{in}} \times Y_{\text{out}} \xrightarrow{\cong} X_{\text{in}} \times Y_{\text{in}} \times X_{\text{out}} \times Y_{\text{out}}$$

that essentially switches the two middle variables:

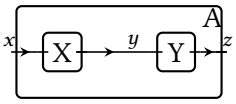
$$\phi_{X,Y}(R_X, R_Y) = \{(x_1, y_1, x_2, y_2) \mid (x_1, x_2) \in R_X \text{ and } (y_1, y_2) \in R_Y\}.$$

As an example, suppose in Fig. 1a we ask that some process in X satisfies the contract $R_X \subseteq \mathbb{R} \times \mathbb{R}$, some process in Y satisfies the contract $R_Y \subseteq \mathbb{R} \times \mathbb{R}$ and some process in Z satisfies the contract $R_Z \subseteq \mathbb{R}^3 \times \mathbb{R}$. The fact that contracts form an algebra on $\mathbf{W}$ ensures that the composite process in A will necessarily satisfy a contract formed only in terms of R_X, R_Y and R_Z and their interconnection $(f_{\text{in}}, f_{\text{out}})$, and specifically

$$R_A = \{(w, u, v, z) \in \mathbb{R}^4 \mid \exists (x, y) \in \mathbb{R}^2 \text{ such that } (w, x) \in R_X, (u, y) \in R_Y, (x, y, v, z) \in R_Z\}.$$

The algebra machinery produces a contract that matches our intuition: whenever the interconnected composite in Fig. 1a receives three real numbers (w, u, v) as inputs, it must produce an output z which is R_Z-allowable by (i.e. related to) (x, y, v), for some real x which is R_X-allowable by w and some real y which R_Y-allowable by u. Not all inputs of this composite A will have an allowable output, and that completely depends on the contracts of its components X, Y and Z.

As another example, which highlights the strong connection between the contract algebra machinery and the usual relation operators, consider a simple wiring diagram with $\mathbb{R}$-typed wires



This morphism $f \colon X \otimes Y \to A$ is described by $(f_{\text{in}}(y, z, x) = (x, y), f_{\text{out}}(y, z) = z)$, and given two contracts R_X and R_Y the formula (8) produces the composite contract

$$R_A = \{(x, z) \mid \exists y \text{ s.t. } (x, y) \in R_X \text{ and } (y, z) \in R_Y\}$$

which is the usual composition of binary relations.

What is particularly interesting about this algebra of contracts is that it is 'agnostic' to the exact specification of the systems. This means that although categorically it is expressed the same way as, for example, Moore machines, it is of a quite different flavor: we are not interested in giving explicit functions that describe the composite process, but in expressing all the possible (input,output) pairs that can be observed on it. This is very convenient especially when connecting systems of different models, for example, a Moore

machine with an 'an abstract machine' [64, §4]. Even if we cannot compose them in the previous sense, since they form distinct algebras (that is, they are described by different functors $\mathbf{W} \to \mathbf{Cat}$), we can still compose and examine the requirements the composite satisfies, in this relational sense.

3.3.2 Independent contracts

We will also be interested in a subclass of static contracts, called *independent*, of the form

$$I = I^1 \times I^2 \subseteq X_{\text{in}} \times X_{\text{out}}$$

These contracts capture cases like 'inputs are always in range I^1 and outputs are always in range I^2, independently from one another'.[6] Of course this is only a special case of arbitrary relations $R \subseteq X_{\text{in}} \times X_{\text{out}}$, since not all subsets of cartesian products are cartesian products of subsets, as a simple argument in the finite case shows: $|\mathcal{P}(X_{\text{in}} \times X_{\text{out}})| = 2^{n \cdot m}$ whereas $|\mathcal{P}(X_{\text{in}})| \cdot |\mathcal{P}(X_{\text{out}})| = 2^{n+m}$. For example, the contract $\{(x, y) \mid x < y\} \subseteq \mathbb{R} \times \mathbb{R}$ is not independent.

One could expect that these contracts form themselves an algebra, namely any wiring composite of independent contracts will also be an independent, rather than a general contract itself. However this is not the case in general: although the parallel placement of boxes with $I_X = I_X^1 \times I_X^2 \subseteq X_{\text{in}} \times X_{\text{out}}$ and $I_Y = I_Y^1 \times I_Y^2 \subseteq Y_{\text{in}} \times Y_{\text{out}}$ produces the independent contract $(I_X^1 \times I_Y^1) \times (I_X^2 \times I_Y^2)$ on $X \otimes Y$, closure under feedback fails. Explicitly, for an independent contract $I_X^1 \times I_X^2 \subseteq X_{\text{in}} \times X_{\text{out}}$ on X, and a wiring diagram $(f_{\text{in}}, f_{\text{out}}) \colon X \to Y$, the formula (8) produces the slightly simpler composite contract

$$R_Y = \{(y_1, y_2) \in Y_{\text{in}} \times Y_{\text{out}} \mid \exists x_2 \in I_X^2 \text{ s.t. } f_{\text{in}}(y_1, x_2) \in I_X^1 \text{ and } f_{\text{out}}(x_2) = y_2\} \tag{9}$$

which shows that y_1 and y_2 are not independent in general, hence R_Y is not of the form $I_Y^1 \times I_Y^2$.

Notice that in certain examples, R_Y can indeed be written as a product itself, for example, when (f_{in}, π_2) is of the form $k \times s$ for two functions k, s. Even more interestingly, due to the special form of morphisms in the wiring diagram category (where they are only made up from projections, diagonals and duplications) in our examples below we will be able to write R_Y as an independent contract itself.[7]

[6] These independent contracts in reality are even more special than that: not only are input restrictions separate from output restrictions, but also each individual *wire* has an associated subset of allowed values on it.

[7] It can be shown that independent contracts indeed form an algebra on $\mathbf{W}$ due to the special morphisms that generate it; the proof is beyond the scope of this paper.

3.3.3 Relation to assume-guarantee contracts

System theory and design has long recognized the need for a formal requirement engineering through mathematical models and formal analysis techniques [9]. As part of contract-based design, there have been multiple efforts to formalize and analyze *assume-guarantee* (AG) contracts [59] and incorporate them in the design as a fundamental concept. We here discuss such examples and how they fit to the previously described static contract model.

Given a box $\mathbb{R} \to \boxed{} \to \mathbb{R}$, an example of an assume-guarantee contract (adapted from Benveniste et al. [9, § IV]) is

$$R_1 : \begin{cases} \text{variables:} & \text{inputs } x, y;\ \text{outputs } z \\ \text{types:} & x, y, z \in \mathbb{R} \\ \text{assumptions:} & y \neq 0 \\ \text{guarantees:} & z = \dfrac{x}{y} \end{cases} \tag{10}$$

This explicitly makes the assumption that the environment (namely the inputs coming either from the external world or from other component systems) will never provide the input $y = 0$, essentially leaving the behavior for that input undefined. In our formalism, we can express this contract as

$$R_1 = \{(x, y, z) \mid y \neq 0 \wedge z = \frac{x}{y}\} \subseteq \mathbb{R} \times \mathbb{R} \times \mathbb{R}$$

indicating the fact that the input $y = 0$ will never occur on the input wire of the box; and if it did, the contract is violated. A different choice we could make, assuming the initial AG contract is really expressing an "if...then..." requirement, is

$$R_1' = \{(x, y, z) \mid y \neq 0 \Rightarrow z = \frac{x}{y}\} \subseteq \mathbb{R} \times \mathbb{R} \times \mathbb{R}$$

which is clearly a different subset of allowable values on the wires. For example, $(3, 0, 25) \in R_1'$ whereas $(3, 0, 25) \notin R_1$.

We now consider a standard AG contract operator called *contract composition and system integration*, and we realize it from the perspective of the wiring diagram algebra machinery – consequently a more general setting. Explicitly, the AG contract composition operator as described for example by Benveniste et al. [9, § IV.B] or Le et al. [39], takes two AG contracts $R_1 = (A_1, G_1)$ and $R_2 = (A_2, G_2)$ and produces a new AG contract $R_1 \otimes R_2$ (notice that this is a completely different use of our earlier monoidal product symbol $\otimes$) with

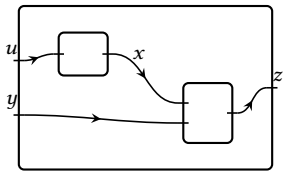
$$\begin{aligned} G_{R_1 \otimes R_2} &= G_1 \wedge G_2 \\ A_{R_1 \otimes R_2} &= \max\{A \mid A \wedge G_2 \Rightarrow A_1, A \wedge G_1 \Rightarrow A_2\} \end{aligned} \tag{11}$$

only when R_1 and R_2 are *compatible*, namely $A_{R_1 \otimes R_2} \neq \emptyset$. Since $A_{R_1 \otimes R_2}$ is the weakest assumption such that the two referred implications hold, if non-empty it ensures that there exists some environment in which the two contracts properly interact: when put in the context of a process that satisfies the first contract, the assumption of the second contract will be met and vice-versa. At first sight, this definition looks 'symmetric', since it considers a certain compatibility of output guarantee/input assumption in both directions, but in reality this is not quite the case.

One issue with the above AG contract composition is that the *names* of the variables and not only the types of the wires need to match, in order to connect along them [9, 52]. For example, the contract R_1 as in (10) can be composed with the contract on $\mathbb{R} \to \boxed{} \to \mathbb{R}$

$$R_2 : \begin{cases} \text{variables:} & \text{inputs } u;\ \text{outputs } x \\ \text{types:} & u, x \in \mathbb{R} \\ \text{assumptions:} & \top \\ \text{guarantees:} & x > u \end{cases}$$

not along any wire, as could be deduced by noticing that all wire types are $\mathbb{R}$, but specifically along the wire with variable name x. Pictorially, we can realize them as inhabiting boxes wired as



and using the formulas (11) we obtain

$$A_{R_1 \otimes R_2} = \max\{A \mid (A \wedge (x > u) \Rightarrow y \neq 0) \wedge (A \wedge (z = x/y) \Rightarrow \top)\} = (y \neq 0)$$
$$C_{R_1 \otimes R_2} = (x > u) \wedge (z = x/y).$$

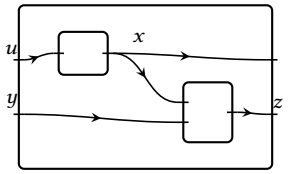
On the other hand, composing R_1 and R_2 using the static contract algebra (Section 3.3.1) for the above wiring diagram ($f_{\text{in}}(x, z, u, y) = (u, x, y)$, $f_{\text{out}}(x, z) = z$), we obtain the composite contract

$$R = \{(u, y, z) \in \mathbb{R}^3 \mid \exists x \in \mathbb{R} \text{ s.t. } y \neq 0 \wedge x > u \wedge z = x/y\},$$

which could be written in AG form as $A = \{(u, y) \mid y \neq 0\}$ and $G = \{z \mid \exists x > u \text{ s.t. } z = x/y\}$. Notice that the contract algebra machinery does not present this variable-match problem, since it does not prevent us from composing along the second input wire of X_1 or even do first X_1 and then X_2 in the opposite order, since all types of wires are real numbers. In all these cases, it would be possible to compute appropriate composite contracts in this uniform way.

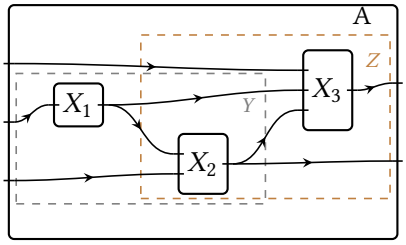
The second issue, which can also be noticed from the above calculation, is that the

assume and guarantee of the composite contract include information that mix the variables of the resulting input and output wires. For example, using the AG formalism, the variables of $R_1 \otimes R_2$ are set to be $\{u, y\}$ for inputs and $\{x, z\}$ for outputs, therefore this operation behaves as if the intermediate wires of a system composition can be extracted as extra output wires to the outside world:



This 'choice' does not agree with the wiring diagram formalism, and moreover is somewhat ad-hoc given that it could potentially add arbitrary many wires to the composite system, essentially according to the result of the contract composition. Adding extra wires is of course possible for the algebra formalism, but corresponds to a choice of architecture on how we decide to wire the subcomponents together, rather than a necessity that arises from dealing with contracts.

Finally, the AG formalism asks that compositions $(R_1 \otimes R_2) \otimes R_3$ and $R_1 \otimes (R_2 \otimes R_3)$ give equivalent contracts, and that so do $R_1 \otimes R_2$ and $R_2 \otimes R_1$. In the contract algebra formalism, the first statement follows for any $\mathbf{W}$-algebra: consider a possible wiring of three boxes, each inhabited with a contract (or a behavior)



First composing the contracts R_1 and R_2 and then the result with R_3 comes from the application of the functor $C\colon \mathbf{W} \to \mathbf{Cat}$ on a wiring diagram morphism

$$(X_1 \otimes X_2) \otimes X_3 \to Y \otimes X_3 \to A$$

whereas the other way around comes from the application of the functor C on the morphism

$$X_1 \otimes (X_2 \otimes X_3) \to X_1 \otimes Z \to A$$

which both express the same morphism $X_1 \otimes X_2 \otimes X_3 \to A$ in $\mathbf{W}$ as an implementation of A (Section 3.2).

Regarding the second statement about $R_1 \otimes R_2$ and $R_2 \otimes R_1$, in the AG formalism this can indeed be proved due to the symmetric formulation of composition (11) as observed

earlier. However, this refers more to the earlier variable-sharing clause (which would not allow the composition along arbitrary wires therefore in arbitrary order) and less to composition intuition: changing the order of two boxes and expecting the same behavior or requirements is something highly non expected, from a categorical but also a design point of view due to the input-output directionality. As a result, commutativity in this AG setting is slightly misleading, since it is just a technical term relevant to the constructed formula (it does not really have an effect on the operation) rather to an actually commuting composition which is not expected to hold – and does not, in the algebra formalism.

4 Compositional Cyber-Physical Systems Modeling and Analysis

In this section, we use the preceding algebraic formalism to illustrate a compositional CPS theory. We model an unmanned aerial vehicle (UAV), analyze it with respect to its control behavior, decompose it to a system architecture and constrain it using contracts. This process manifests the power, flexibility and further potential of the wiring diagram compositional framework in the concrete context of CPS analysis and design.

4.1 System Behavior

We algebraically recover a standard controls model *compositionally* in the behavior algebra (Section 3.1) of the familiar form

$$s_{k+1} = \mathscr{A} s_k + \mathscr{B} c_k,$$

where $s_k \in \mathbb{R}^n$ is the discrete time state, s_{k+1} (also denoted $\dot{s}$ or $u(s,c)$ using the earlier update function notation) is the subsequent time-step state and $c_k \in \mathbb{R}^n$ is the control signal/output, and

$$y_k = \mathscr{C} s_k + \mathscr{D} c_k$$

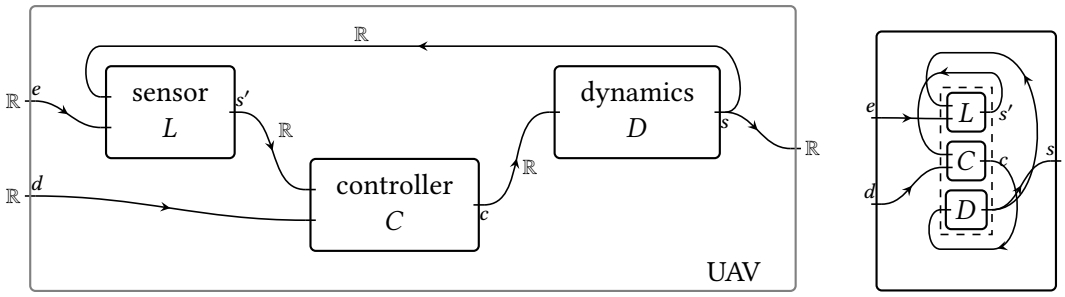
is the measurement, which is also in $\mathbb{R}^n$. We assume $\mathscr{D} = 0$.

We are going to illustrate the algebra machinery using longitudinal equations of motion for a fixed-winged aircraft represented in the following state-space model [49]

$$\begin{pmatrix} \dot{a} \\ \dot{q} \\ \dot{\theta} \end{pmatrix} = \begin{pmatrix} -0.313 & 56.7 & 0 \\ -0.0139 & -0.426 & 0 \\ 0 & 56.7 & 0 \end{pmatrix} \begin{pmatrix} a \\ q \\ \theta \end{pmatrix} + \begin{pmatrix} 0.232 \\ 0.0203 \\ 0 \end{pmatrix} \begin{pmatrix} \delta \end{pmatrix} \tag{12}$$

$$y = \begin{pmatrix} 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} a \\ q \\ \theta \end{pmatrix},$$

where a is the angle of attack, q is the pitch rate, θ is the pitch angle and δ is the elevator



$$f_{\text{in}} : \mathbb{R}^3 \times \mathbb{R}^2 \to \mathbb{R}^5,\ (s', c, s, e, d) \mapsto (s, e, s', d, c)$$
$$f_{\text{out}} : \mathbb{R}^3 \to \mathbb{R},\ (s', c, s) \mapsto s$$

Figure 3: The physical decomposition of the UAV, where d denotes the desired state, s' the predicted state, c the control action, s the current state, and e the environmental inputs.

deflection angle. This behavior is the *composite* one, built up from the subcomponents behavior and their wiring depicted in Fig. 3.

Working with the linear time-invariant system algebra $\mathcal{L} : \mathbf{W}_{\mathbf{Lin}} \to \mathbf{Cat}$ (Section 3.1.2), suppose $(S_L, \mathscr{A}_L, \mathscr{B}_L, \mathscr{C}_L)$, $(S_C, \mathscr{A}_C, \mathscr{B}_C, \mathscr{C}_C)$ and $(S_D, \mathscr{A}_D, \mathscr{B}_D, \mathscr{C}_D)$ are three linear systems inhabiting the respective boxes of Fig. 3, with

$$\begin{aligned} u_L(s_L, s, e) &= \mathscr{A}_L \cdot s_L + \mathscr{B}_L \cdot (s, e) & r_L(s_L) &= \mathscr{C}_L \cdot s_L \\ u_C(s_C, d, s') &= \mathscr{A}_C \cdot s_C + \mathscr{B}_C \cdot (d, s') & r_C(s_C) &= \mathscr{C}_C \cdot s_C \\ u_D(s_D, c) &= \mathscr{A}_D \cdot s_D + \mathscr{B}_D \cdot (c) & r_D(s_D) &= \mathscr{C}_D \cdot s_D. \end{aligned}$$

Using the algebra machinery for the specific wiring diagram (Fig. 3) given by matrix transformations

$$\begin{cases} f_{\text{in}} = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \end{pmatrix} = \begin{pmatrix} {}_5(\mathscr{A}^f)_3 & {}_5(\mathscr{B}^f)_2 \end{pmatrix} \\ f_{\text{out}} = \begin{pmatrix} 0 & 0 & 1 \end{pmatrix} = \mathscr{C}^f \end{cases}$$

we can compute the composite linear dynamical system that inhabits the box UAV from the formulas (7). Its state space is $S_L \times S_C \times S_D$, and its update and readout linear functions

are

$$u_{UAV} : S_L \times S_C \times S_D \times \mathbb{R}^2 \to S_L \times S_C \times S_D,$$
$$(s_L, s_C, s_D, d, e) \mapsto (\mathscr{A}_L s_L + \mathscr{B}_L \begin{pmatrix} \mathscr{C}_D s_D \\ e \end{pmatrix}, \mathscr{A}_C s_C + \mathscr{B}_C \begin{pmatrix} \mathscr{C}_L s_L \\ d \end{pmatrix}, \mathscr{A}_D s_D + \mathscr{B}_D \mathscr{C}_C s_C)$$
$$r_{UAV} : S_L \times S_C \times S_D \to \mathbb{R}$$
$$(s_L, s_C, s_D) \mapsto \mathscr{C}_D s_D.$$

We assume, for simplicity,[8] that the state spaces of the sensor and controller are in $\mathbb{R}^2$. Knowing that only the dynamics D actually relate to the triplet (a, q, θ), we deduce that S_D is in $\mathbb{R}^3$ which results in a composite state space S_{UAV} in $\mathbb{R}^2 \times \mathbb{R}^2 \times \mathbb{R}^3 \cong \mathbb{R}^7$. Moreover, from the shape of the boxes according to (6) we deduce that the matrices $\mathscr{A}_L$, $\mathscr{A}_C$, $\mathscr{B}_L$ and $\mathscr{B}_C$ are two-by-two, $\mathscr{C}_L$ and $\mathscr{C}_C$ are one-by-two, whereas $\mathscr{A}_D$ is three-by-three, $\mathscr{B}_D$ is three-by-one and $\mathscr{C}_D$ is one-by-three.

Unravelling the above update and readout functions of the composite linear time-invariant system denoted by UAV, the only output of the composite system behavior is that of the dynamics D, since by tuple (7)

$$\mathscr{C}_{\text{UAV}} = \mathscr{C}^f \cdot \mathscr{C}_{L \otimes C \otimes D} = \begin{pmatrix} 0 & 0 & {}_1(\mathscr{C}_D)_3 \end{pmatrix}.$$

Hence for obtaining equation (12), in the specific example we deduce that $\mathscr{C}_D = \begin{pmatrix} 0 & 0 & 1 \end{pmatrix}$ meaning only θ is outputted to the outside world as desired.

For an element of the state space $\mathbb{R}^7$ of the form $(\vec{s}_L, \vec{s}_C, \overbrace{a, q, \theta}^{\vec{s}_D})$, isolating the first two variables we obtain

$$\dot{\vec{s}}_L = \mathscr{A}_L \vec{s}_L + {}_2(\mathscr{B}_L)_2 \begin{pmatrix} \overbrace{\mathscr{C}_D \vec{s}_D}^{\theta} \\ e \end{pmatrix} \quad \text{and} \quad \dot{\vec{s}}_C = \mathscr{A}_C \vec{s}_C + {}_2(\mathscr{B}_C)_2 \begin{pmatrix} \mathscr{C}_L \vec{s}_L \\ d \end{pmatrix},$$

which could be viewed as some extra information of the composite system relating to the behaviors of the sensor and controller, not appearing in equation (12) but part of the total system's behavior.

Now isolating the last three variables we obtain a description

$$\begin{pmatrix} \dot{\alpha} \\ \dot{q} \\ \dot{\theta} \end{pmatrix} = {}_3(\mathscr{A}_D)_3 \begin{pmatrix} \alpha \\ q \\ \theta \end{pmatrix} + {}_3(\mathscr{B}_D)_1 \mathscr{C}_C \vec{s}_C.$$

[8] See end of this section for a concrete example where L and C are populated by linear functions, thus their state space matches their input linear space (Section 3.1.3).

Comparing with the desired equation (12), the elevator deflection angle δ is the output of the controller $\mathscr{C}_C s_C$ which matches the physical reality, and the $\mathscr{A}_D$, $\mathscr{B}_D$ are completely determined by the composite description, namely

$$\mathscr{A}_D = \begin{pmatrix} -0.313 & 56.7 & 0 \\ -0.0139 & -0.426 & 0 \\ 0 & 56.7 & 0 \end{pmatrix} \qquad \mathscr{B}_D = \begin{pmatrix} 0.232 \\ 0.0203 \\ 0 \end{pmatrix}.$$

The remaining data $\mathscr{A}_{L,C}, \mathscr{B}_{L,C}, \mathscr{C}_{L,C}$ depend on engineering and physical parameters.

We were thus able to partly reverse-engineering a given composite system behavior (12), where for the given system architecture (Fig. 3) we completely identified the behavior of the linear time-invariant system D by determining $S_D, \mathscr{A}_D, \mathscr{B}_D, \mathscr{C}_D$. We also obtained certain information about the other two subcomponents C and L: for example, two possible behaviors could be the linear functions (for example, signal concatenations) $s' = s + e$ for the sensor L and the linear function $c = s' + d$ for the controller C. Expressing those as linear time-invariant systems (Section 3.1.3), we obtain the following description

$$(S_L, \mathscr{A}_L, \mathscr{B}_L, \mathscr{C}_L) = \left(\mathbb{R}^2, \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}, \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 1 \end{pmatrix}\right), \qquad u_L(\vec{s}_L, s, e) = (s \;\; e),\; r_L(\vec{s}_L) = s_L^1 + s_L^2$$

$$(S_C, \mathscr{A}_C, \mathscr{B}_C, \mathscr{C}_C) = \left(\mathbb{R}^2, \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}, \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 1 \end{pmatrix}\right), \qquad u_C(\vec{s}_C, s', d) = (s' \;\; d),\; r_C(\vec{s}_C) = s_C^1 + s_C^2$$

Then the composite system's update function is explicitly computed, using (7), as

$$\begin{cases} \dot{\vec{s}}_L = \begin{pmatrix} \theta \\ e \end{pmatrix} \\ \dot{\vec{s}}_C = \begin{pmatrix} s_L^1 + s_L^2 \\ d \end{pmatrix} \\ \dot{a} = -0.313a + 56.7q + 0.232 s_C^1 + 0.232 s_C^2 \\ \dot{q} = -0.0139a - 0.426q + 0.0203 s_C^1 + 0.0203 s_C^2 \\ \dot{\theta} = 56.7q \end{cases}$$

where s_C^1 and s_C^2 are essentially the previous desired state s' and input d, producing the deflection angle δ that appears in (12). The first two equations give the functions of L and C (whose states are placeholders for their inputs at each instance), whereas the last three give the dynamics D as before. Informally, this shows the interplay between what the system is sensing, what its desired operating state is, and how it must react. If there were more information about the elevator deflection angle δ, that would restrict the possible behaviors for C appropriately.

From a more categorical perspective, the above process is summarized as follows: given

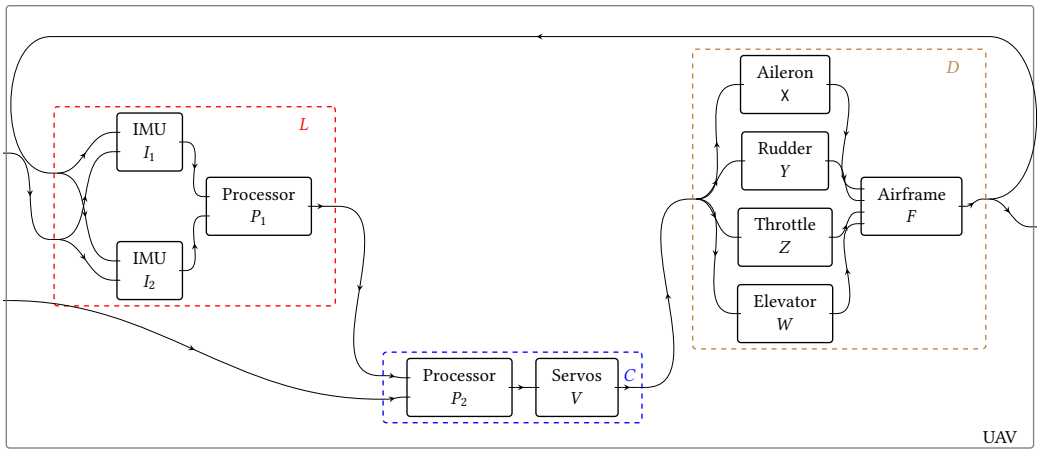


Figure 4: Any decomposition, including the previous one (Fig. 3) resides within the slice category $\mathbf{W}/\mathrm{UAV}$. In this case, the slice category contains all possible design decisions that adhere to the behavioral model; we pick one such design choice.

an algebra $\mathcal{L}$ and a wiring diagram $f\colon L \otimes C \otimes D \to UAV$ in $\mathbf{W}_{\mathbf{Lin}}$ (Fig. 3), as well as an object of the target category $\mathcal{L}(\mathrm{UAV})$, namely a specific linear system as in equation (12) inhabiting the outside box UAV, the goal is to find an object in the pre-image of the given system under the composite functor

$$\mathcal{L}(L) \times \mathcal{L}(C) \times \mathcal{L}(D) \xrightarrow{\phi_{L,C,D}} \mathcal{L}(L \otimes C \otimes D) \xrightarrow{\mathcal{L}(f)} \mathcal{L}(\mathrm{UAV}).$$

Such a problem certainly does not have a unique solution, namely a unique description of the three systems that form the composite, but for example in this specific case due to the form the wiring diagram, the component system

$$(S_D, \mathscr{A}_D, \mathscr{B}_D, \mathscr{C}_D)$$

was completely determined by the composite behavior. Further work would aim to shed light on possible shapes of wiring diagrams that have better identifiable solutions under algebras of interest.

4.2 System architecture

One of the important advantages of expressing system decompositions as a morphism in the category $\mathbf{W}$ is that we can perform further zoomed-in decompositions as desired in a *hierarchical* way (Section 3.2), and these are all realized as composite morphisms in the wiring diagram category.

For example, consider a possible UAV architecture (Fig. 3). We may further choose to implement the sensor box L using two IMU units I_1, I_2 and a processor P_1 in a certain interconnection. Expressing this as a morphism with target L (an object in the slice category

$\mathbf{W}/L$) namely $g\colon I_1 \otimes I_2 \otimes P \to L$ means that we can compose this with the original one-level implementation f to obtain a two-level zoomed-in decomposition

$$(I_1 \otimes I_2 \otimes P_1) \otimes C \otimes D \xrightarrow{g\otimes 1\otimes 1} L \otimes C \otimes D \xrightarrow{f} UAV$$

that only 'opens-up' the box L. We could moreover implement the control as well as the dynamics box, and decompose them in a choice of subcomponents and wires between them. An example where the control box is decomposed into P_2 followed by V in a serial composition, and the dynamics box is decomposed into four parallel boxes, X, Y, Z and W followed by F amounts to choosing a specific $h\colon P_2 \otimes V \to C$ in $\mathbf{W}/C$ and a specific $k\colon X \otimes Y \otimes Z \otimes W \otimes F \to D$ in $\mathbf{W}/D$. Combining all these morphisms we have the composition (Fig. 4):

$$(I_1 \otimes I_2 \otimes P_1) \otimes (P_2 \otimes V) \otimes (X \otimes Y \otimes Z \otimes W \otimes F) \xrightarrow{g\otimes h\otimes k} L \otimes C \otimes D \xrightarrow{f} \text{UAV}$$

that can be considered as a single morphism from the tensor of all second-level subcomponents to the box UAV. Pictorially, this would be realized by erasing the intermediate colored dashed boxes.

4.3 System requirements

We will use the algebra of static contracts $C\colon \mathbf{W} \to \mathbf{Cat}$ (Section 3.3.1), where all requirements are expressed as subsets of the cartesian product of input and output types. Consider the original system decomposition to sensor, controller, and dynamics boxes (Fig. 3) and suppose we have certain contracts on these components given by

$$R_L \subseteq \mathbb{R}^2 \times \mathbb{R}, \quad R_C \subseteq \mathbb{R}^2 \times \mathbb{R}, \quad R_D \subseteq \mathbb{R} \times \mathbb{R}.$$

These contracts could be any subsets, from the extreme case of equality which means that *all* combinations of inputs and outputs are allowed, to some specific requirement imposed to the example at hand, or in certain cases some *maximal* contract dictated by a discrete dynamical system (governed by a difference equation) that actually inhabits the box.

The contract algebra applies to the wiring diagram of Fig. 3 and based on the formula (8) produces a contract $R_{UAV} \subseteq \mathbb{R}^2 \times \mathbb{R}$ on the composite system, with the following explicit description

$$\begin{aligned}\mathbb{R}^3 \supseteq R_{UAV} = \{(a_1, a_2, a_3) \in \mathbb{R}^3 \mid \exists (x, y) \in \mathbb{R}^2 \text{ such that} \\ (a_3, a_1, x) \in R_L,\ (x, a_2, y) \in R_C,\ (y, a_3) \in R_D\}\end{aligned}$$

Further, we could assume that all contracts are independent as per Section 3.3.2, namely

they can be written as products of subsets of each wire type independently, like

$$R_L = R_L^1 \times R_L^2 \times R_L^3, \quad R_C = R_C^1 \times R_C^2 \times R_C^3, \quad R_D = R_D^1 \times R_D^2$$

where all components are subsets of $\mathbb{R}$ – i.e. the allowed values on each wire are completely unrelated to one another. Then the composite contract (9) takes the following, also independent contract form

$$R_{\text{UAV}} = \begin{cases} R_L^2 \times R_C^2 \times (R_L^1 \cap R_D^2) & \text{if } R_L^3 \cap R_C^1 \neq \emptyset \text{ and } R_C^3 \cap R_D^1 \neq \emptyset \\ \emptyset & \text{if } R_L^3 \cap R_C^1 = \emptyset \text{ or } R_C^3 \cap R_D^1 = \emptyset \end{cases} \tag{13}$$

The above formula expresses that the allowable tuples that can be observed on the composite system are the L- and C-external input contracts for the two input wires, along with an intersection of contracts for the output wire, subject to whether there exists a scenario where the contracts of the intermediate wires match: if their intersection is non-empty, there exist appropriate values that work for both contracts and the total system 'runs'. Otherwise the composed contracts are incompatible and the composite system fails to adhere to a contract, namely there is no guarantee about its observable input and output values (expressed by the empty set contract) and possibly the whole process fails.

We now proceed to a similar process to what we have seen before (Section 4.1), which in a sense reverse-engineered the behavior of the subcomponents, given a composite behavior of the total system using the system behavior algebra machinery. In this setting, given a specific desired requirement R_{UAV} on the composite system, we will identify possible contracts on the components that produce that specific composite; once again we do not expect unique solution to this problem.

Suppose the envisioned composite contract on the behavioral representation of our example UAV (Fig. 3) is

$$R_{\text{UAV}} = [0, 100] \times [-20, +20] \times [-35, +35]$$

This contract represents a possible requirement that the desired UAV pitch is no more or less than 20 degrees and the plane really must not pitch more or less than 35 degrees for a hypothetical safe flight. As hypothetical environmental conditions, we assume air speed does not exceed 100 km/h.

Comparing the above composite contract against equation (13), we can first of all deduce that

$$R_L^2 = [0, 100] \qquad R_C^2 = [-20, +20]$$

namely the external inputs for L and C are necessarily constrained by the ranges of the given composite contract on those wires. Moreover we have that $R_L^1 \cap R_D^2 = [-35, 35]$ and also that necessarily the intersections $R_L^3 \cap R_C^1$ and $R_C^3 \cap R_D^1$ are non-empty – since the

composite contract is indeed non-empty. Notice how all these intersections correspond to specific wiring connections or splittings we performed between subcomponents for the initial UAV's implementation.

Given these restrictions, we are free to choose contracts that satisfy them, for example

$$R_D^2 = [-35, +35],\ R_L^1 = R_L^3 = R_C^1 = R_C^3 = R_D^1 = \mathbb{R}$$

The above choices are made to also dispose of 'bad scenarios' for the given interconnection of the boxes. For example, choosing the opposite contracts for R_D^2 and R_L^1 would be mathematically correct since their intersection is still $[-35, 35]$, but could lead to a real value of, say, 40 degrees entering the sensor L which would then violate its contract (that said "all my inputs on the first wire will be less than 35"). Although in general, processes can be wired together as long as types match, in the contract algebra setting it is implied (by the algebra machinery) that the only values passing through an interconnected wire are those in the intersections of the individual (independent) contracts – so long as the composite system does not 'break'. It is important to realize that the contract algebra *describes* the observable inputs and outputs on a running composite machine, rather than *ensures* that the process runs: this has to be safeguarded by the designer also. This discussion relates to future work regarding 'total' or 'deterministic' contracts.

5 On unification

Having manifested the wiring diagram formalism for behavior, architecture and requirements of an UAV, we now summarize and further discuss how this categorical interpretation of CPS models leads to unification of these aspects of system design and analysis.

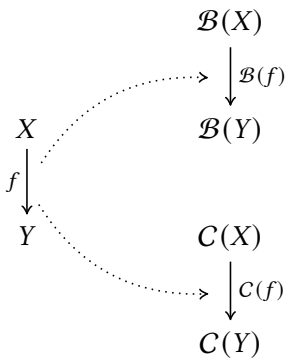
Starting with some cyber-physical process Y, we usually model its behavior, mathematically described for example via some equations, and also the requirements it satisfies or should satisfy. We earlier discussed Moore machines and linear time-invariant systems; there can be other algebras of system behavior,[9] so here we generically speak of the 'behavior algebra' which is any one of them, using the notation $\mathcal{B}$. As we saw, categorically these are certain objects $B_Y \in \mathcal{B}(Y)$ of the category of all the possible behaviors (Section 3.1), and similarly the requirements are objects $R_Y \in \mathcal{C}(Y)$ of the category of all contracts (Section 3.3) that could be associated to such a process, via lax monoidal functors

$$\mathcal{B}, \mathcal{C} : \mathbf{W} \to \mathbf{Cat}.$$

To formally discuss and capture the behavior and requirements in terms of subprocesses, the designer first chooses some valid architecture of Y which is categorically expressed by choosing a morphism $f : X \to Y$ in the category $\mathbf{W}$, namely an element of the slice

[9] For example, *machines* serve as an all-inclusive general system notion that allows us to compose systems of different description [64, § 4].

category $\mathbf{W}/Y$ (Section 3.2). Then the behavior algebra and requirements algebra, independently, produce assignments



(14)

The designer then decides on 'pre-image' objects $B_X \in \mathcal{B}(X)$ and $R_X \in \mathcal{C}(X)$ which, under these functors on the right-hand side, produce the original composite behavior and requirement on Y. As we saw, there could be multiple choices for B_X and R_X (Sections 4.1 and 4.3). Also, the designer can decompose even further to subprocesses, on which the analysis carries on in the same formal way (Section 4.2). Moreover, they may choose to go back and change the architecture to some alternative implementation $g\colon Z \to Y$, if that is physically sensible and allows to easier obtain the end results. Later on, using algorithms such tests could assist in deciding on the most optimal solutions.

On top of the above story, which summarizes the narrative of the current work, we now sketch some additional connections between these two independent algebras of behavior and requirements, which further clarify their formal relation.

First of all, there is an *algebra map*[10] $\alpha\colon \mathcal{B} \Rightarrow \mathcal{C}$ which assigns to each specific physical behavior of a process $B_Y \in \mathcal{B}(Y)$, the *maximally satisfied* contract by it, $\alpha_Y(B_Y) \in \mathcal{C}(Y)$; in [64, Prop. 5.2.15] this is done in an abstract setting. Informally, if a box $\mathbb{R} \to \boxed{X} \to \mathbb{R}$ is inhabited by the function $f(x) = 6x$, its maximally satisfied contract is in effect $\{(a, 6a) \mid a \in \mathbb{R}\} \subseteq \mathbb{R}^2$. However, the system also satisfies the contracts $\mathbb{R} \times 6\mathbb{R}$ or $\mathbb{R} \times 3\mathbb{R}$, or even $\mathbb{R} \times \mathbb{R}$ as the maximum such. The fact that the assignment $\mathcal{B}(Y) \ni B_Y \mapsto \alpha_Y(B_Y) \in \mathcal{C}(Y)$ is an algebra map signifies in particular that the above mappings (14) are part of a commutative square relating system behavior and requirements for a specific wiring diagram $f\colon X \to Y$

$$\begin{array}{ccc} \mathcal{B}(X) & \xrightarrow{\alpha_X} & \mathcal{C}(X) \\ {\scriptstyle \mathcal{B}(f)}\downarrow & & \downarrow{\scriptstyle \mathcal{C}(f)} \\ \mathcal{B}(Y) & \xrightarrow[\alpha_Y]{} & \mathcal{C}(Y) \end{array}$$

Intuitively, this says that for a given system decomposition into subcomponents, first composing the behaviors of the internal boxes using the behavior algebra and then talking about the contract that composite satisfies is the *same* as first computing the maximal con-

[10] Formally, this is a monoidal natural transformation between the two lax monoidal functors [45, § XI].

tracts the components satisfy individually and then composing using the contract algebra. This provides extra flexibility for passing between different models, not only for this specific algebra map example but also for other maps relating different algebras that may be established.

Another way to combine the behavior $\mathcal{B}$ and requirements C algebra is to construct a new algebra of *contracted behaviors* that, to each process placeholder Y assigns a pair of a physical behavior along with some contract it satisfies. This allows us to compose using both algebras simultaneously and choosing which information to look at; this abstract algebra is already defined [64, Prop. 4.5.5] for a specific behavior algebra and provides a tool that allows us to essentially relate two algebras via some desired condition inside their product.

The above sketched behavior and requirements formal connections, as well as the whole methodology presented in detail in this paper, shall be further developed to account for the crucial notion of time,[11] particularly a *compositional* model of real-time computing for CPS, which to this day raises several challenges [26, 41, 42, 68].

6 Related Work

Computational and physical modeling in the context of CPS is well-studied [22, 56, 73]. However, there is still a need for research in compositional methods for model-based system design [25, 70] and particularly for a compositional CPS theory that is able to model and simulate both the computational and the physical aspects of CPS [12, 20, 40], which can formally relate those necessary views.

Category theorists have worked extensively in the area of compositional systems. Among the primary results of that general program has been the *relation* of different types of models, for example, abstracting and unifying automata and dynamical systems [6]. For further discussion and references on alternative categorical approaches on systems theory, see Schultz et al. [64, § 1]. Furthermore, the application of algebraic structures has made seminal contributions in behavioral specification of programs [11] and modal logic [34], both of which show up later in control [27].

Other recent work in systems theory proposes category theory as the solution to model federation but lacks significant theoretical development. Hasuo [33], for example, provides much of the context and reasoning for using category theory in system design through coalgebras but the work represents a skeleton of what should be done. An older but significantly more fleshed out version of coalgebraic modeling for CPS was proposed by Matsikoudis and Lee [47], but only focuses on modeling the behavioral view of transition systems. Additionally, category theory lends itself as a possible *quasi*-formal approach to requirements management [29, 37]. Our framework is instead formal, in the strict sense, and we

[11] The abstract categorical framework where time is added to the wiring diagram model has been formally studied using *sheaves* on real-time intervals [64, § 3].

use the contracts algebra, which has shown to be effective in CPS.

The theory of contracts has had significant development, especially as applied to CPS [10, 63], including notions of contract composition [54]. Recently there are also concrete applications in the form, for example, of a toolkit on top of SysML [24], which will make contracts increasingly accessible to system designers. Contracts have been implemented as an end-to-end requirements engineering framework, but more importantly have also been merged with linear temporal logic (LTL) specifications that can compile down to contracts [53]; this idea could also be implemented into our compositional CPS theory. Our approach to contracts is more general than the often used AG formalism. Specifically, in AG contracts, names *and* types of variables need to match, while the categorical formulation only requires that types match. Examples of synthesis from a contract-based design specification [30], show that it is possible to use our generalized version of contracts to adapt control synthesis tools [36, 60] with our notion of modeling and simulation. Therefore, we would be able to not only have composition among requirements, system behaviors, and system architectures but we would also be able to produce a possible implementation that is compositionally constrained at any given level; this would represent an improvement over approaches that only consider the compositional verification of architecture models [19].

Hybrid systems is a well-established version of computational dynamical systems theory [3] (another being timed process algebras [16] or more recently model interfaces [57]). Ames (as well as Tabuada et al. to some extent [69]) did develop a categorical theory of hybrid systems [5], which could be used to relate or otherwise use results from the well-established formalism of hybrid systems in our proposed framework as future work, but also the opposite; potentially strengthening notions of composition [4, 15] in hybrid systems from category theory.

Compositional CPS theory can assist with model conformance [58] and model federation at large [31]. Complementary works using category theory have shown small but useful examples of categorical modeling of systems and how they can facilitate model conformance in CPS [8, 13, 14, 50].

7 Conclusion

The forthcoming SysML V2 standard is attempting to bridge the gap between requirement, behavioral, and structural models, showing an increasing need for unification and scalability of models in system design [55]. In this paper, we present a categorical framework to achieve such a unification of models and simulation tools as an alternative to current approaches, such as domain metamodeling and (semi-)manual model transformations. Through the categorical framework we also show that there is a functorial relationship between the architectural and behavioral modeling domains, which unifies what was previously a distinct difference between there domains. We show that there is a mul-

tidimensionality to modeling abstraction and manage it formally through the preceding formalism.

An additional benefit of category theory in this domain is its closeness to execution by means of dependent typed languages, which in the future could allow for a merge between modeling and code that traverses throughout the full lifecycle of the system. These stages might include requirements generation, control law simulation, and finally architectural design and deployment. In the domain of CPS we achieve that by unifying the controls and computation and requirement views without inventing a new formalism but rather by zooming in and out of different layers of abstraction with a formal composition rule. Furthermore, with this approach we are able to relate static views of the system with their dynamics, or otherwise executable, model representations.

The use of wiring diagrams already provides both an appealing and familiar syntax (that of boxes and arrows) as well as algebraic semantics; that is, the perspective of *systems as algebras*, which formalizes mathematically the diagrammatic reasoning already used in engineering. We posit that as systems become increasingly complex such semantics will be important to assess a system's dependable, safe, and secure operation. These semantics do not need to be visible to the practitioner but provide a flexible scaffolding for interchanging between modeling paradigms and metrics within a modeling language. Ultimately, the algebraic view of systems models has the potential of producing more scalable modeling efforts.

Acknowledgments

G. Bakirtzis and C.H. Fleming are partially supported through the Systems Engineering Research Center (SERC) under USDOD Contract HQ0034-13-D-0004, NASA under research grant NNX16AK47A, and National Science Foundation (NSF) under grant No. 1739333.

C. Vasilakopoulou is supported by the General Secretariat for Research and Technology (GSRT) and the Hellenic Foundation for Research and Innovation (HFRI).

References

[1] S. Abramsky and B. Coecke. Categorical quantum mechanics. In *Handbook of Quantum Logic and Quantum Structures*. Elsevier, 2009. doi:10.1016/B978-0-444-52869-8.50010-4.

[2] F. Allgöwer, J. B. de Sousa, J. Kapinski, P. Mosterman, J. Oehlerking, P. Panciatici, M. Prandini, A. Rajhans, P. Tabuada, and P. Wenzelburger. Position paper on the challenges posed by modern applications to cyber-physical systems theory. *Nonlinear Analysis: Hybrid Systems*, 2019. doi:10.1016/j.nahs.2019.05.007.

[3] R. Alur, T. A. Henzinger, G. Lafferriere, and G. J. Pappas. Discrete abstractions of hybrid systems. *Proceedings of the IEEE*, 2000. doi:10.1109/5.871304.

[4] R. Alur, R. Grosu, I. Lee, and O. Sokolsky. Compositional modeling and refinement for hierarchical hybrid systems. *Journal of Logical and Algebraic Methods in Programming*, 2006. doi:10.1016/j.jlap.2005.10.004.

[5] A. D. Ames. *A categorical theory of hybrid systems*. PhD thesis, University of California, Berkeley, 2006.

[6] M. A. Arbib and E. G. Manes. Machines in a category. *Journal of Pure and Applied Algebra*, 1980. doi:10.1016/0022-4049(80)90090-0.

[7] P. Asare, G. Bakirtzis, R. Bernard, D. Broman, E. Lee, G. Prinsloo, M. Torngren, and S. Sunder. Cyber-physical systems – a concept map. `https://cyberphysicalsystems.org`, 2020.

[8] G. Bakirtzis, C. Vasilakopoulou, and C. H. Fleming. Compositional cyber-physical systems modeling. In *Proceedings of the 2019 Applied Category Theory Conference (ACT 2020)*, Electronic Proceedings in Theoretical Computer Science. Open Publishing Association, 2020.

[9] A. Benveniste, B. Caillaud, D. Nickovic, R. Passerone, J.-B. Raclet, P. Reinkemeier, A. L. Sangiovanni-Vincentelli, W. Damm, T. A. Henzinger, and K. G. Larsen. Contracts for system design. *Research Report, RR-8147, INRIA*, 2012.

[10] A. Benveniste, B. Caillaud, D. Nickovic, R. Passerone, J.-B. Raclet, P. Reinkemeier, A. L. Sangiovanni-Vincentelli, W. Damm, T. A. Henzinger, and K. G. Larsen. Contracts for system design. *Foundations and Trends in Electronic Design Automation*, 2018. doi:10.1561/1000000053.

[11] M. Bidoit and R. Hennicker. Proving the correctness of behavioural implementations. In *Proceedings of the 1995 International Conference on Algebraic Methodology and Software Technology (AMAST 1995)*. Springer, 1995. doi:10.1007/3-540-60043-4_51.

[12] S. Bliudze, S. Furic, J. Sifakis, and A. Viel. Rigorous design of cyber-physical systems – linking physicality and computation. *Software and Systems Modeling*, 2019. doi:10.1007/s10270-017-0642-5.

[13] S. Breiner, R. D. Sriram, and E. Subrahmanian. Compositional models for complex systems. In *Artificial Intelligence for the Internet of Everything*. Elsevier, 2019. doi:10.1016/B978-0-12-817636-8.00013-2.

[14] S. Breiner, O. Marie-Rose, B. S. Pollard, and E. Subrahmanian. Operadic diagnosis in hierarchical systems. In *Proceedings of the 2nd Applied Category Theory Conference (ACT 2019)*, 2020. doi:10.4204/EPTCS.323.5.

[15] D. Bresolin, P. Collins, L. Geretti, R. Segala, T. Villa, and S. Z. Gonzalez. A computable and compositional semantics for hybrid automata. In *Proceedings of the 23rd ACM International Conference on Hybrid Systems: Computation and Control (HSCC 2020)*. ACM, 2020. doi:10.1145/3365365.3382202.

[16] M. Broy. Functional specification of time-sensitive communicating systems. *ACM Transactions on Software Engineering and Methodology*, 1993. doi:10.1145/151299.151302.

[17] J. Buck, S. Ha, E. A. Lee, and D. G. Messerschmitt. Ptolemy: A framework for simulating and prototyping heterogeneous systems. In *Readings in hardware/software co-design*. ACM, 2001. ISBN 1558607021.

[18] B. Coecke. Quantum picturalism. *Contemporary physics*, 2010. doi:10.1080/00107510903257624.

[19] Darren Cofer, Andrew Gacek, Steven Miller, Michael W Whalen, Brian LaValley, and Lui Sha. Compositional verification of architectural models. In *Proceedings of the 4th International Symposium on NASA Formal Methods (NFM 2012)*, Lecture Notes in Computer Science. Springer, 2012. doi:10.1007/978-3-642-28891-3_13.

[20] F. Cremona, M. Lohstroh, D. Broman, E. A. Lee, M. Masin, and S. Tripakis. Hybrid co-simulation: it's about time. *Software & Systems Modeling*, 2019. doi:10.1007/s10270-017-0633-6.

[21] V. C. V. De Paiva. *The Dialectica Categories*. PhD thesis, University of Cambridge, UK, 1990.

[22] P. Derler, E. A. Lee, and A. L. Sangiovanni-Vincentelli. Modeling cyber-physical systems. *Proc. IEEE*, 2012. doi:10.1109/JPROC.2011.2160929.

[23] Z. Diskin, T. Maibaum, and K. Czarnecki. A model management imperative: Being graphical is not sufficient, you have to be categorical. In *Proceedings of the 11th European Conference on Modelling Foundations and Applications (ECMFA@STAF 2015)*, 2015. doi:10.1007/978-3-319-21151-0_11.

[24] I. Dragomir, I. Ober, and C. Percebois. Contract-based modeling and verification of timed safety requirements within SysML. *Software & Systems Modeling*, 2017. doi:10.1007/s10270-015-0481-1.

[25] F. Durán, R. Heinrich, D. Pérez-Palacín, C. L. Talcott, and S. Zschaler. Composing Model-Based Analysis Tools (Dagstuhl Seminar 19481). *Dagstuhl Reports*, 2020. doi:10.4230/DagRep.9.11.97.

[26] S. A. Edwards and E. A. Lee. The case for the precision timed (PRET) machine. In *Proceedings of the 44th Annual Design Automation Conference (DAC)*, 2007.

[27] G. E. Fainekos, H. Kress-Gazit, and G. J. Pappas. Temporal logic motion planning for mobile robots. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation (ICRA 2005)*. IEEE, 2005. doi:10.1109/ROBOT.2005.1570410.

[28] J. N. Foster, M. B. Greenwald, J. T. Moore, B. C. Pierce, and A. Schmitt. Combinators for bidirectional tree transformations: A linguistic approach to the view-update problem. *ACM Transactions on Programming Languages and Systems*, 2007. doi:10.1145/1232420.1232424.

[29] S. Gebreyohannes, W. Edmonson, and A. Esterline. Formalization of the responsive and formal design process using category theory. In *Proceedings of the 2018 Annual IEEE International Systems Conference (SysCon 2018)*. IEEE, 2018. doi:10.1109/SYSCON.2018.8369508.

[30] K. Ghasemi, S. Sadraddini, and C. Belta. Compositional synthesis via a convex parameterization of assume-guarantee contracts. In *Proceedings of the 23rd ACM International Conference on Hybrid Systems: Computation and Control (HSCC 2020)*. ACM, 2020. doi:10.1145/3365365.3382212.

[31] F. R. Golra, F. Dagnat, J. Souquières, I. Sayar, and S. Guerin. Bridging the gap between informal requirements and formal specifications using model federation. In *Proceedings of the 16th International Conference on Software Engineering and Formal Methods (SEFM@STAF 2018)*. Springer, 2018. doi:10.1007/978-3-319-92970-5_4.

[32] J. Hansen and R. Ghrist. Opinion dynamics on discourse sheaves. *arXiv:2005.12798 [math.DS]*, 2020.

[33] I. Hasuo. Metamathematics for systems design: Comprehensive transfer of formal methods techniques to cyber-physical systems. *New Generation Computing*, 2017. doi:10.1007/s00354-017-0023-1.

[34] J. Hughes. Modal operators for coequations. In *Proceedings of the 2001 Coalgebraic Methods in Computer Science (CMCS 2001)*, Electronic Notes in Theoretical Computer Science. Elsevier, 2001. doi:10.1016/S1571-0661(04)80909-5.

[35] A. Joyal and R. Street. Braided tensor categories. *Advances in Mathematics*, 1993. doi:10.1006/aima.1993.1055.

[36] M. Mazo Jr., A. Davitian, and P. Tabuada. PESSOA: A tool for embedded controller synthesis. In *Proceedings of the 22nd International Conference on Computer Aided Verification (CAV 2010)*, Lecture Notes in Computer Science. Springer, 2010. doi:10.1007/978-3-642-14295-6_49.

[37] N. Kibret, W. W. Edmonson, and S. Gebreyohannes. Category theoretic based formalization of the verifiable design process. In *Proceedings of the 2019 IEEE International Systems Conference (SysCon 2019)*. IEEE, 2019. doi:10.1109/SYSCON.2019.8836804.

[38] F. W. Lawvere and S. H. Schanuel. *Conceptual mathematics: a first introduction to categories*. Cambridge University Press, 2009.

[39] T. T. H. Le, R. Passerone, U. Fahrenberg, and A. Legay. Contract-based requirement modularization via synthesis of correct decompositions. *ACM Transactions on Embedded Computing Systems*, 2016. doi:10.1145/2885752.

[40] E. A. Lee. Cyber-physical systems – Are computing foundations adequate. Position Paper for NSF Workshop on Cyber-Physical Systems: Research Motivation, Techniques and Roadmap, 2006.

[41] E. A. Lee. Fundamental limits of cyber-physical systems modeling. *ACM Transactions on Cyber-Physical Systems*, 2016. doi:10.1145/2912149.

[42] E. A. Lee. What is real time computing? a personal view. *IEEE Design & Test*, 2018. doi:10.1109/MDAT.2017.2766560.

[43] T. Leinster. *Basic Category Theory*. Cambridge University Press, 2014. doi:10.1017/CBO9781107360068.

[44] N. G. Leveson. *Engineering a safer world: Systems thinking applied to safety*. MIT Press, 2011. ISBN 9780262016629.

[45] S. Mac Lane. *Categories for the working mathematician*. Springer, 1998.

[46] G. Masetti, S. Chiaradonna, F. Di Giandomenico, B. Feddersen, and W. H. Sanders. An efficient strategy for model composition in the möbius modeling environment. In *Proceedings of the 14th European Dependable Computing Conference (EDCC 2018)*. IEEE Computer Society, 2018. doi:10.1109/EDCC.2018.00029.

[47] E. Matsikoudis and E. A. Lee. From transitions to executions. In *Revised Selected Papers from the 11th International Workshop Coalgebraic Methods in Computer Science (CMCS 2012), Colocated with ETAPS 2012*, Lecture Notes in Computer Science. Springer, 2012. doi:10.1007/978-3-642-32784-1_10.

[48] Mihailo D. Mesarovic and Yasuhiko Takahara. *Abstract Systems Theory*. Springer, 1989. ISBN 978-3-540-46038-1.

[49] B. Messner, D. Tilbury, R. Hill, and J. D. Taylor. Control tutorials for Matlab and Simulink: Aircraft pitch. https://web.archive.org/web/20200509164711/http://ctms.engin.umich.edu/CTMS/index.php?ex 2020.

[50] J. S. Nolan, B. S. Pollard, S. Breiner, D. Anand, and E. Subrahmanian. Compositional models for power systems. In *Proceedings of the 2nd Applied Category Theory Conference (ACT 2019)*, Electronic Proceedings in Theoretical Computer Science, 2020. doi:10.4204/EPTCS.323.10.

[51] P. Nuzzo and A. L. Sangiovanni-Vincentelli. Hierarchical system design with vertical contracts. In *Principles of Modeling - Essays Dedicated to Edward A. Lee on the Occasion of His 60th Birthday*, Lecture Notes in Computer Science. Springer, 2018. doi:10.1007/978-3-319-95246-8_22.

[52] P. Nuzzo, A. L. Sangiovanni-Vincentelli, D. Bresolin, L. Geretti, and T. Villa. A platform-based design methodology with contracts and related tools for the design of cyber-physical systems. *Proceedings of the IEEE*, 2015. doi:10.1109/JPROC.2015.2453253.

[53] P. Nuzzo, M. Lora, Y. A. Feldman, and A. L. Sangiovanni-Vincentelli. CHASE: contract-based requirement engineering for cyber-physical system design. In *Proceedings of the 2018 Design, Automation & Test in Europe Conference & Exhibition (DATE 2018)*. IEEE, 2018. doi:10.23919/DATE.2018.8342122.

[54] C. Oh, E. Kang, S. Shiraishi, and P. Nuzzo. Optimizing assume-guarantee contracts for cyber-physical system design. In *Proceedings of the Design, Automation & Test in Europe Conference & Exhibition (DATE 2019)*. IEEE, 2019. doi:10.23919/DATE.2019.8715284.

[55] OMG Systems Modeling Language. Systems modeling language (SysML®) v2 request for proposal (RFP). Technical report, Object Management Group (OMG), 2019.

[56] A. Platzer. The logical path to autonomous cyber-physical systems. In *Proceedings of the 16th International Conference on Quantitative Evaluation of Systems (QEST 2019)*, Lecture Notes in Computer Science. Springer, 2019. doi:10.1007/978-3-030-30281-8_2.

[57] J.-B. Raclet, É. Badouel, A. Benveniste, B. Caillaud, A. Legay, and R. Passerone. A modal interface theory for component-based design. *Fundamenta Informaticae*, 2011. doi:10.3233/FI-2011-416.

[58] H. Roehm, J. Oehlerking, M. Woehrle, and M. Althoff. Model conformance for cyber-physical systems: A survey. *ACM Transactions on Cyber-Physical Systems*, 2019. doi:10.1145/3306157.

[59] Í. Í. Romeo, A. L. Sangiovanni-Vincentelli, C.-W. Lin, and E. Kang. Quotient for assume-guarantee contracts. In *Proceedings of the 16th ACM/IEEE International Conference on Formal Methods and Models for System Design (MEMOCODE 2018)*. IEEE, 2018. doi:10.1109/MEMCOD.2018.8556872.

[60] M. Rungger and P. Tabuada. Abstracting and refining robustness for cyber-physical systems. In *Proceedings of the 17th International Conference on Hybrid Systems: Computation and Control*, 2014.

[61] Dylan Rupel and David I. Spivak. The operad of temporal wiring diagrams: formalizing a graphical language for discrete-time processes, 2013.

[62] J. Rushby. Composing safe systems. In *Revised Papers from the International Workshop on Formal Aspects of Component Software (FACS 2011)*. Springer, 2011. doi:10.1007/978-3-642-35743-5_2.

[63] A. L. Sangiovanni-Vincentelli, W. Damm, and R. Passerone. Taming Dr. Frankenstein: Contract-based design for cyber-physical systems. *European Journal of Control*, 2012. doi:10.3166/ejc.18.217-238.

[64] P. Schultz, D. I. Spivak, and C. Vasilakopoulou. Dynamical systems and sheaves. *Applied Categorical Structures*, 2019. doi:DOI:10.1007/s10485-019-09565.

[65] D. Spivak. The steady states of coupled dynamical systems compose according to matrix arithmetic. arXiv:1512.00802 [math.CT], 2016.

[66] D. I. Spivak. The operad of wiring diagrams: formalizing a graphical language for databases, recursion, and plug-and-play circuits. arXiv:1305.0297 [cs.DB], 2013.

[67] D. I. Spivak. *Category theory for the sciences*. MIT Press, 2014.

[68] J. A. Stankovic. Misconceptions about real-time computing: A serious problem for next-generation systems. *Computer*, 1988. doi:10.1109/2.7053.

[69] P. Tabuada, G. J. Pappas, and P. U. Lima. Composing abstractions of hybrid systems. In *Proceedings of the 5th International Workshop on Hybrid Systems: Computation and Control (HSCC 2002)*, Lecture Notes in Computer Science. Springer, 2002. doi:10.1007/3-540-45873-5_34.

[70] S. Tripakis. Compositionality in the science of system design. *Proceedings of the IEEE*, 2016. doi:10.1109/JPROC.2015.2510366.

[71] D. Vagner, D. I. Spivak, and E. Lerman. Algebras of open dynamical systems on the operad of wiring diagrams. *Theory and Applications of Categories*, 2015.

[72] J. C. Willems. The behavioral approach to open and interconnected systems. *IEEE Control Systems Magazine*, 2007. doi:10.1109/MCS.2007.906923.

[73] Q. Zhu and A. L. Sangiovanni-Vincentelli. Codesign methodologies and tools for cyber-physical systems. *Proc. IEEE*, 2018. doi:10.1109/JPROC.2018.2864271.

A Nomenclature

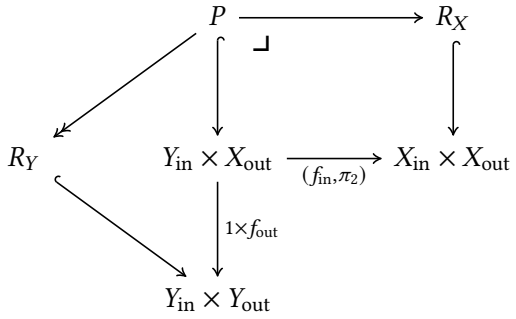
Here we summarize some of the symbols we use and their meaning in category theory as a quick guide for working engineers to effectively navigate the preceding formalism.

$\mathbf{C}$	a generic category
1	identity morphism
f or $\xrightarrow{f}$	morphism in a category
$g \circ f$	composition (right to left)
$\begin{array}{ccc} A & \xrightarrow{f} & B \\ {\scriptstyle h}\downarrow & & \downarrow{\scriptstyle g} \\ C & \xrightarrow{k} & D \end{array}$	commutative diagram standing for equation $g \circ f = k \circ h$
$\cong$	isomorphism
$(\mathbf{V}, \otimes, I)$	a generic monoidal category
F or $\xrightarrow{F}$	functor
$F(A)$ or FA	functor application on objects
$F(f)$ or Ff	functor application on morphisms
$\mathbf{C}/C$	slice category over object C
Set	the category of sets and functions
Lin	the category of linear spaces and linear maps
$\times$	cartesian product of sets (or linear spaces)
$\Delta\colon X \to X \times X$	duplication function
Cat	the category of categories and functors
$\mathbf{W}$	the category of labelled boxes and wiring diagrams (with types in **Set**)
$\mathbf{W}_{\mathbf{Lin}}$	the category of labelled boxes and wiring diagrams (with types in **Lin**)
$\mathcal{M}$	the algebra of Moore machines; a lax monoidal functor $\mathbf{W} \to \mathbf{Cat}$
$\mathcal{L}$	the algebra of linear time-invariant systems; a lax monoidal functor $\mathbf{W}_{\mathbf{Lin}} \to \mathbf{Cat}$

$\mathcal{C}$	the algebra of (static) contracts
$\mathcal{B}$	a generic behavior algebra; could be $\mathcal{M}$ or $\mathcal{L}$ (among others)

B Contract Pullback

Regarding the static contract algebra (Section 3.3.1), the functor $\mathcal{C}f : \mathcal{C}(X) \to \mathcal{C}(Y)$ for a wiring diagram $f : X \to Y$ (1) assigns a contract $R_X \subseteq X_{\text{in}} \times X_{\text{out}}$ on the inside box to a contract $R_X \subseteq X_{\text{in}} \times X_{\text{out}}$ on the outside box, following a two-step procedure:



(15)

First, we compute the pullback – a limit of a diagram of two morphisms with common codomain [43, 5.1.16] – of the relation R_X along the function (f_{in}, π_2) which is defined by $Y_{\text{in}} \times X_{\text{out}} \ni (y, x') \mapsto (f_{\text{in}}(y, x'), x') \in X_{\text{in}} \times X_{\text{out}}$. The explicit description of that pullback in **Set** is

$$P = \{(y, x') \mid (f_{\text{in}}(y, x'), x') \in R_X\}$$

namely those pairs of Y-inputs and X-outputs which the bottom function actually maps to elemenets of the contract R_X. Second, we take the image factorization of the inclusion $P \subseteq Y_{\text{in}} \times X_{\text{out}}$ post-composed with the function $1 \times f_{\text{out}}$ that maps some (y, x') to the pair $(y, f_{\text{out}}(x'))$. Recall that the image of a function is the subset of its codomain where all elements of the domain get mapped to, namely for an arbitrary $g : A \to B$, $Im(g) = \{b \in B \mid \exists a \in A \text{ such that } g(a) = b\}$. In the end, using the above constructions of the two-step process exhibited in (15), the explicit description of the resulting contract is precisely equation (8).