

Figure 1: (a) This equation guarantees that the preconditions of **Pick** are satisfied.

(b) This equation guarantees the desired specification (ball at goal).

A few questions arise when trying to formalize this. First, how are informational resources supposed to behave in general? To model such resources, we borrow the notion of a *data service* from [9]; we summarize this in Definition 1. Data services are algebraic structures (defined internally to SMCs) which formalize the operations of *filtering*, *copying*, and *deleting* pieces of data.

Furthermore, what does it mean for an entity to have a datum as an attribute (e.g. in the way that the robot above “has a location”)? We argue that this question can be answered by requiring that the data service associated to the informational object “acts on” the other object in a suitable sense. To provide more control over the behavior of these attributes, we define the notion of an *attribute structure* on a SMC. These concepts are discussed in Subsection 2.1.

When considering the semantics of these categories, it is often the case that certain morphisms turn out to be “partially defined.” This manifests itself through a partial order structure on hom-sets, or more precisely an enrichment over Poset, the category of partially-ordered sets (posets) and monotone increasing functions. Due to similarities between these partial orders across semantics, we find it useful and illuminating to enrich the *syntax* categories over Poset as well. By doing so we are able to impose extra conditions on these attribute structures that mimic the ways users expect informational resources to behave. The details of this enrichment are spelled out in Subsection 2.2.

In our motivating example, we use these categories with attributes and their diagrams to coordinate two semantic models at different levels of abstraction, based on the principle of functorial semantics. First, we define a Boolean semantics based on the Planning Domain Definition Language (PDDL) [7]; a presentation of the categorical syntax can be used to generate a PDDL domain, and the solution to a PDDL problem can be used to define an associated string diagram. Once a high-level plan has been identified, a second mapping to a category of geometric semantics can be used to parameterize a path planning algorithm. We give sketches of these categories and mappings in Section 3.

We seek to achieve two principal goals in this paper. First, we aim to develop rigorous foundations for the forthcoming paper [2], which explores similar ideas with a greater focus on the engineering than on the mathematics. Second, we seek to present interesting examples of categorical modeling in engineering. It is our hope that the applications presented in this paper will motivate other researchers to investigate the connections between category theory and engineering models.

## 2 Categories with Attributes

Our goal in this section is to develop a definition of a *category with attributes* – that is, a (symmetric monoidal) category in which objects can have “information” (defined internally to the category) attached to them in some way. We develop one notion of a category with attributes in Subsection 2.1. In Subsection 2.2, we adapt this notion to the case where the categories involved are Poset-enriched. This additional structure allows us to impose conditions on the relationships between certain morphisms, allowing us to connect categories with attributes to our intuition about partially-defined morphisms (e.g. partial functions).

### 2.1 General Case

For a formal categorical notion of information, we borrow the definition of a *data service* from [9].<sup>1</sup> This definition is restated as follows. As hinted above, we will depict the underlying objects of data services with *dashed lines* to contrast them with other objects.

**Definition 1** ([9]). A *data service* in a SMC  $(\mathcal{C}, \otimes, I)$  consists of

- An object  $D$  in  $\mathcal{C}$ ;
- A *multiplication morphism*  $\mu : D \otimes D \rightarrow D$ ;
- A *comultiplication morphism*  $\delta : D \rightarrow D \otimes D$ ; and
- A *counit morphism*  $\varepsilon : D \rightarrow I$ .

The morphisms of a data service are depicted as follows:

$$\mu := \begin{array}{c} D \quad D \\ \text{---} \quad \text{---} \\ \text{---} \quad \text{---} \\ \circ \\ \text{---} \\ D \end{array} ; \quad \delta := \begin{array}{c} D \\ \text{---} \\ \circ \\ \text{---} \quad \text{---} \\ D \quad D \end{array} ; \quad \varepsilon := \begin{array}{c} D \\ \text{---} \\ \circ \end{array} \tag{2}$$

These morphisms are also required to satisfy several axioms:

- $(V, \mu)$  is a commutative semigroup object in  $\mathcal{C}$ ;
- $(V, \delta, \varepsilon)$  is a commutative comonoid object in  $\mathcal{C}$ ;
- $\mu$  and  $\delta$  satisfy the *Frobenius laws*:

$$\begin{array}{c} D \quad D \\ \text{---} \quad \text{---} \\ \text{---} \quad \text{---} \\ \circ \\ \text{---} \quad \text{---} \\ D \quad D \end{array} = \begin{array}{c} D \quad D \\ \text{---} \quad \text{---} \\ \text{---} \quad \text{---} \\ \circ \\ \text{---} \quad \text{---} \\ D \quad D \end{array} = \begin{array}{c} D \quad D \\ \text{---} \quad \text{---} \\ \text{---} \quad \text{---} \\ \circ \\ \text{---} \quad \text{---} \\ D \quad D \end{array} \tag{3}$$

---

<sup>1</sup>Readers familiar with Frobenius algebras will observe that a data service is the same as a special commutative Frobenius algebra except for the fact that data services are not required to have units. Non-unitality can be unavoidable in certain applications; for example, every object in PartFn, the category of sets and partial functions, has a canonical data service structure, but this structure typically does not admit a unit.

- $\mu$  and  $\delta$  satisfy the *special law*:

$$\begin{array}{c}
 D \\
 \vdots \\
 \text{---} \circ \text{---} \\
 \text{---} \circ \text{---} \\
 \vdots \\
 D
 \end{array}
 =
 \begin{array}{c}
 \vdots \\
 D \\
 \vdots
 \end{array}
 \quad (4)$$

When we have no reason to explicitly reference the morphisms  $\mu$ ,  $\delta$ , and  $\varepsilon$ , we will refer to the data service  $(D, \mu, \delta, \varepsilon)$  simply as  $D$ .

Some discussion of the intuition behind data services is in order. As mentioned above, an object with a data service structure can be thought of as a value or datum. From this perspective, the comultiplication morphism can be viewed as creating a copy of the input value (where the copy is indistinguishable from the original), while the counit morphism can be viewed as deleting or forgetting about the input value. Meanwhile, the multiplication morphism can be thought of as filtering for equality: given two values, the multiplication morphism checks whether or not they're the same and returns the common value of both (if they're the same) or nothing (if they aren't). The Frobenius, special, semigroup, and comonoid axioms formalize this intuition.

For any SMC  $\mathcal{C}$ , we can define a category  $\text{Data}(\mathcal{C})$  of data services in  $\mathcal{C}$ . The “obvious” definition of a morphism of data services  $D \rightarrow D'$  (i.e. a morphism  $D \rightarrow D'$  in  $\mathcal{C}$  that is both a semigroup homomorphism and a comonoid homomorphism) will turn out to be too restrictive for our eventual goals. In this general case, we are unable to find any nontrivial useful ways to weaken this “obvious” definition, so we opt for the trivial solution, allowing any morphism between the underlying objects of two data services to be considered a “morphism of data services.” (Contrast this with the discussion in Subsection 2.2, where we are able to formulate acceptable conditions). This is recorded in the following definition.

**Definition 2.** Let  $\mathcal{C}$  be a SMC. The *category of data services* in  $\mathcal{C}$ , denoted  $\text{Data}(\mathcal{C})$ , is the category where:

- Objects are data services in  $\mathcal{C}$
- $\text{Data}(\mathcal{C})(D, D') = \mathcal{C}(D, D')$ , i.e. morphisms in  $\text{Data}(\mathcal{C})$  are arbitrary morphisms between the underlying objects of the domain / codomain.

There exists an obvious (fully faithful) forgetful functor  $U : \text{Data}(\mathcal{C}) \rightarrow \mathcal{C}$ .

In order to define categories with attributes, we still need to determine a proper definition for “attributes” themselves. To an initial approximation, we consider an attribute of an object  $M$  to be a data service  $D$  associated to  $M$  in some way. Given an instance of  $M$ , we should be able to retrieve the associated instance of  $D$ . This notion can be formalized by the morphism  $\gamma : M \rightarrow M \otimes D$  depicted in (5).<sup>2</sup> Furthermore, we should be able to “filter attributes for equality,” checking whether or not the instance of  $D$  associated to the instance of  $M$  agrees with some arbitrary instance of  $D$ . This is formalized by the

<sup>2</sup>An attribute is not determined by the pair  $(M, D)$ , so our depiction here is lacking insofar as it does not represent the specific choice of attribute for  $(M, D)$ . Despite this shortcoming, the depiction here is still useful for brevity, and we have no need to explicitly consider different attributes with the same  $D$  and  $M$ .



Remarkably, a data service action  $(\gamma, \phi)$  is entirely determined by the comonoid action  $\gamma$ , as the following proposition shows.<sup>3</sup>

**Proposition 1.** *Let  $(D, \mu, \delta, \varepsilon)$  be a data service and let  $\gamma : M \rightarrow M \otimes D$  (depicted as above) be a (right) action of the comonoid object  $(D, \delta, \varepsilon)$  on the object  $M$ . Then there exists a unique morphism  $\phi : M \otimes D \rightarrow M$  such that  $(\gamma, \phi)$  forms a data service action of  $D$  on  $M$ . This  $\phi$  is defined as in (10).*

$$\phi := \begin{array}{c} M \\ | \\ \triangleleft \\ | \\ M \end{array} \begin{array}{c} D \\ \text{---} \\ \circ \\ | \\ \circ \\ | \\ \circ \end{array} \begin{array}{c} D \\ \text{---} \\ \circ \\ | \\ \circ \end{array} \quad (10)$$

The proof that such a  $\phi$  is unique is fairly straightforward; in fact, one only needs to use the equality between the first and third terms of (8) and the fact that  $D$  is a data service. A series of immediate diagram chases then establishes all of the other laws of Definition 3 (provided that  $\gamma$  gives a comonoid action of  $(D, \delta, \varepsilon)$  on  $M$ ). In particular, the data service action axioms are overdetermined, so that some of the less immediately intuitive laws (e.g. the equality between the first and second terms of (8)) follows from the other assumptions

Given a data service action  $(\gamma, \phi)$  of some data service  $D$  on some object  $M$ , we can define several other interesting morphisms. For example, because  $\mathcal{C}$  is symmetric and  $D$  is a commutative semigroup / comonoid object, we obtain a pair of morphisms  $(\gamma', \phi')$  giving a “left data service action” of  $D$  on  $M$ , defined by (post- or pre-)composing  $\gamma$  and  $\phi$  with the appropriate braiding. We depict these  $\gamma'$  and  $\phi'$  with the (horizontal) mirror images of our usual notation for  $\gamma$  and  $\phi$  (respectively); the morphisms  $\gamma'$  and  $\phi'$  then satisfy the mirror images of the laws in Definition 3.

Furthermore, we can extend the “filtering” operation from data services to objects equipped with data service actions. This enables us to take instances of two entities  $M$  and  $M'$ , each equipped with a data service action by a common data service  $D$ , and ensure that the corresponding instances of  $D$  are equal. We define this via the morphism  $\chi : M \otimes M' \rightarrow M \otimes M'$  defined in (12).

$$\chi := \begin{array}{c} M \quad M' \\ | \quad | \\ \triangleleft \quad \triangleleft \\ \text{---} \\ | \quad | \\ M \quad M' \end{array} \quad := \quad \begin{array}{c} M \quad M' \\ | \quad | \\ \triangleleft \quad \triangleleft \\ \text{---} \\ \circ \\ | \\ \circ \\ | \\ \circ \end{array} \quad (11)$$

Our depiction of  $\chi$  is *a priori* ambiguous, as it could easily be interpreted as referring to other morphisms, for example:

$$\begin{array}{c} M \quad M' \\ | \quad | \\ \triangleleft \quad \triangleleft \\ \text{---} \\ | \quad | \\ M \quad M' \end{array} \quad \stackrel{?}{=} \quad \begin{array}{c} M \quad M' \\ | \quad | \\ \triangleleft \quad \triangleleft \\ \text{---} \\ | \quad | \\ M \quad M' \end{array} \quad (12)$$

<sup>3</sup>This proposition is closely related to results obtained in [1] for Frobenius algebras. Note that the asymmetry of the data service axioms prevents us from recovering  $\gamma$  from  $\phi$  alone in a data service, as in [1].

However, there is no real ambiguity here, as in fact this equation (and other obvious equations arising from our depiction) do hold, e.g. by Proposition 1. Proposition 1 can also be used to show that the diagrams in Figure 1 (and other well-formed diagrams with the action morphisms displayed horizontally) represent well-defined morphisms.

From this setup, the definition of a category with attributes (or equivalently an attribute structure on a SMC) is straightforward.

**Definition 4.** An *attribute structure*  $(\mathcal{A}, E, V, \gamma)$  on a SMC  $\mathcal{C}$  consists of:

- A category  $\mathcal{A}$ , called the *category of attributes*;
- A functor  $E : \mathcal{A} \rightarrow \mathcal{C}$ ;
- A functor  $V : \mathcal{A} \rightarrow \text{Data}(\mathcal{C})$ ;
- A natural transformation<sup>4</sup>  $\gamma : E \rightarrow E \otimes (U \circ V)$

such that:

- Each  $\gamma_A : E(A) \rightarrow E(A) \otimes V(A)$  gives a action of the comonoid  $(V(A), \delta_{V(A)}, \epsilon_{V(A)})$  on  $E(A)$ .

When we have an attribute structure on  $\mathcal{C}$  in mind, we refer to  $\mathcal{C}$  as a *category with attributes*.

An attribute structure on  $\mathcal{C}$  be viewed as a “collection of distinguished attributes in  $\mathcal{C}$ .” The naturality condition on  $\gamma$  ensures that, given  $f : A \rightarrow A'$  in  $\mathcal{A}$ , the map  $V(f)$  allows us to “predict” the effects of  $E(f)$  on the relevant informational objects. Note that we require no naturality condition on the  $\phi_A$  maps associated to each  $\gamma_A$  through Proposition 1. This is because we cannot anticipate that the “processing” performed by  $V(f)$  will produce meaningful results when applied to an arbitrary input.

## 2.2 Poset-Enriched Case

When modeling attributes, we are often interested in SMCs which are enriched in Poset. This enriched structure can be used to model the notion that a morphism or process is partially defined, i.e. that it could fail to execute on certain inputs. Heuristically, morphisms  $f$  and  $g$  satisfy  $f \leq g$  iff the results of  $f$  are the same as those of  $g$  when both exist, and whenever  $f$  successfully executes given a certain input, so too does  $g$ . We spell out the details of this enrichment for ease of reference.

**Definition 5.** A Poset-enrichment of a category  $\mathcal{C}$  is an assignment of a partial order (uniformly denoted  $\leq$ ) to each hom-set  $\mathcal{C}(A, B)$  such that:

- If  $f \leq f'$  and  $g \leq g'$ , and  $g \circ f$  and  $g' \circ f'$  exist, then  $g \circ f \leq g' \circ f'$ .

If  $(\mathcal{C}, \otimes, I)$  is a SMC, we require furthermore:

- If  $f \leq f'$  and  $g \leq g'$ , then  $f \otimes g \leq f' \otimes g'$ .

A Poset-enriched functor between Poset-enriched categories  $\mathcal{C}, \mathcal{D}$  is a functor  $F : \mathcal{C} \rightarrow \mathcal{D}$  such that the associated functions  $\mathcal{C}(A, B) \rightarrow \mathcal{D}(FA, FB)$  are all monotone increasing.

From this viewpoint, a certain condition on the data services in a Poset-enriched SMC seems desirable. Namely, filtering two values for equality and then returning both values should yield the same results as simply returning both values whenever the former operation is defined. This is best formalized in the following axiom, which appears in [3].

---

<sup>4</sup>Recall  $U : \text{Data}(\mathcal{C}) \rightarrow \mathcal{C}$  is the forgetful functor.

**Definition 6.** A data service  $D$  in a Poset-enriched SMC  $\mathcal{C}$  is said to be *well-behaved with respect to the enrichment* (for brevity, “well-behaved”) if it satisfies:

$$\begin{array}{c} D & D \\ \text{---} & \text{---} \\ \circ & \\ \text{---} & \text{---} \\ D & \\ \text{---} & \text{---} \\ \circ & \\ \text{---} & \text{---} \\ D & D \end{array} \leq \begin{array}{c} | & | \\ D & D \end{array} . \quad (13)$$

One is led to wonder whether or not an analogous property to that satisfied by well-behaved data services holds for data service actions by such data services. This is indeed the case, as shown in the following proposition.

**Proposition 2.** Let  $\mathcal{C}$  be a Poset-enriched SMC,  $D$  a well-behaved data service in  $\mathcal{C}$ , and  $M$  an object of  $\mathcal{C}$  together with a data service action by  $D$ . Then:

$$\begin{array}{c} M & D \\ | & \text{---} \\ \triangleright & \\ | & \text{---} \\ M & D \\ | & \\ M & \end{array} \leq \begin{array}{c} | & | \\ M & D \end{array} \quad (14)$$

We can also prove a useful proposition about the morphism  $\chi$  described above.

**Proposition 3.** Let  $\mathcal{C}$  be a Poset-enriched SMC,  $D$  a well-behaved data service in  $\mathcal{C}$ , and  $M$  an object of  $\mathcal{C}$  together with a data service action by  $D$ . Then:

$$\begin{array}{c} M & M' \\ | & | \\ \triangleright & \text{---} & \triangleleft \\ | & | \\ M & M' \end{array} \leq \begin{array}{c} | & | \\ M & M' \end{array} \quad (15)$$

As indicated above, well-behaved data services enable us to create a useful variation on the category  $\text{Data}(\mathcal{C})$  defined previously.

**Definition 7.** Let  $(\mathcal{C}, \otimes, I, \leq)$  be a Poset-enriched SMC. Define  $\text{Data}'(\mathcal{C})$ , the *category of well-behaved data services* in  $\mathcal{C}$ , to be the category where:

- Objects are well-behaved data services in  $\mathcal{C}$ , and
- Morphisms  $D \rightarrow D'$  are *lax data service homomorphisms* from  $D$  to  $D'$ ; that is, morphisms  $f : D \rightarrow D'$  satisfying the axioms (see [3]):

$$\begin{array}{c} D \\ | \\ \boxed{f} \\ | \\ D' \\ \text{---} & \text{---} \\ \circ & \\ \text{---} & \text{---} \\ D' & D' \end{array} \leq \begin{array}{c} D \\ \text{---} & \text{---} \\ \circ & \\ \text{---} & \text{---} \\ \boxed{f} & \boxed{f} \\ | & | \\ D' & D' \end{array} \quad (16)$$

$$\begin{array}{c} D \\ \boxed{f} \\ D' \circlearrowleft \end{array} \leq \begin{array}{c} D \\ \vdots \\ \circlearrowleft \end{array} \quad (17)$$

$$\begin{array}{c} D \quad D \\ \circlearrowleft \\ D \\ \boxed{f} \\ D' \end{array} \leq \begin{array}{c} D \quad D \\ \boxed{f} \quad \boxed{f} \\ D' \quad \circlearrowleft \quad D' \\ \vdots \\ D' \end{array} \quad (18)$$

A morphism  $f$  in  $\text{Data}'(\mathcal{C})$  is *deterministic* if it is a (strong) comonoid homomorphism, i.e. if both (16) and (17) are equalities.

The inequalities in Definition 7 are described in [3] as (some of) those which hold between relations and (Set-based) monoids. That is, each object of the categories  $\text{Rel}$  or  $\text{PartFn}$  comes equipped with a natural data service structure (with respect to the Cartesian product of sets as tensor product) such that the inequalities described above hold for all morphisms  $f$  in the respective category. In fact, (16) is always an equality in  $\text{PartFn}$ , though it's only an inequality in  $\text{Rel}$  and other non-deterministic contexts. Since we are using Poset-enrichments to describe partial definition of operations, requiring these inequalities in  $\text{Data}'(\mathcal{C})$  helps us to accurately reflect the desired behavior.

**Definition 8.** A *Poset-enriched attribute structure* on a Poset-enriched SMC  $\mathcal{C}$  is an attribute structure  $(\mathcal{A}, E, V, \gamma)$  such that:

- $\mathcal{A}$ ,  $E$ , and  $V$  are all Poset-enriched;
- $V$  factors through the inclusion  $\text{Data}'(\mathcal{C}) \hookrightarrow \text{Data}(\mathcal{C})$ .

### 2.3 An Example

A key motivating example for our definition of categories with attributes is the modeling of planning in robotics. This context provides a collection of excellent simple (and not-so-simple) examples of the interaction of physical and informational resources. We walk through a toy example in this context in the hopes of clarifying our definitions above. This example will be described in the non-enriched case, though readers can easily extend it to the Poset-enriched case as desired.

It should be noted that the purpose of this example is to provide intuition for the rest of the paper and inspiration for future research into these themes. As such, this example may need modifications before being used in practical applications or being connected to the functorial semantics below.

**Example 1.** Suppose that we have a strict SMC  $\mathcal{C}$  with:

- generator objects  $R$  (robot),  $B$  (ball),  $R_B$  (robot holding ball), and  $L$  (location);
- morphisms  $\mu, \delta, \varepsilon$  such that  $(L, \mu, \delta, \varepsilon)$  is a data service;
- for each generator object  $X$  other than  $L$ , a morphism  $\gamma_X$  comprising an action of  $(L, \delta, \varepsilon)$  on  $X$ ;
- and morphisms **MoveTo** :  $R \otimes L \rightarrow R$ , **MoveTo'** :  $R_B \otimes L \rightarrow R_B$ , **Pick** :  $R \otimes B \rightarrow R_B$ , and **Place** :  $R_B \rightarrow R \otimes B$ .



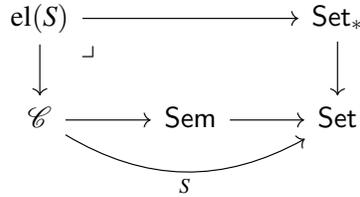


Figure 2: Schematic of the discussion at the beginning of Section 3. Special cases of interest are  $\text{Sem} = \text{BoolAlg}^{\text{op}}$  and  $\text{Sem} = \text{Geom}$ .

mathematical sense of intuitive terms like “instance of an entity” that were used above. See Figure 2 for a depiction of this setup.

In this section, we examine the semantics of categories with attributes by way of two key examples (Boolean and geometric semantics) relevant to applications in engineering. We also illustrate the above construction involving categories of elements in the case of geometric semantics.

### 3.1 Boolean Semantics

The Planning Domain Definition Language (PDDL) is a standard encoding for “classical” planning problems involving collections of Boolean variables and discrete actions that operate on them [7]. It extends the earlier STRIPS<sup>5</sup> language with features including typing and object equality. PDDL aligns quite well with the diagrammatic syntax presented earlier, providing both a graphical interface to define and report PDDL problems and solutions, and an approach to string-diagram generation based on existing PDDL solvers.

PDDL can be divided into three fundamental units: *domains*, *problems* and *solutions*. Domains define the relevant (Boolean) features of the situation of interest, as well as the actions that manipulate these features. A problem is posed relative to a domain; it specifies a collection of elements, their initial state, and a desired goal state. A solution gives a sequence of applied actions which transform the initial state into the goal.

Before proceeding we briefly recall a few facts about Boolean algebras [6], which will provide the substance for our high-level semantics. The free Boolean algebra over a finite set of atomic statements  $S$  is given by  $\text{Bool}(S) := 2^{2^S}$ ; here we think of a map  $s : S \rightarrow 2$  as a truth function defining a global state, while a proposition defines a map  $q : 2^S \rightarrow 2$  picking out a subset of global states.

Any element of a Boolean algebra  $q \in B$  defines an associated quotient algebra  $B \twoheadrightarrow B/q$ ; logically speaking,  $B/q$  represents the theory  $B$  extended by the axiom  $q$ . A homomorphism  $b : B \rightarrow B'$  descends to the quotient  $B/q$  if and only if  $b(q) = \top$ , in which case we write  $b \models q$ .

A *point* of a Boolean algebra  $B$  (also called a valuation) is a homomorphism  $b : B \rightarrow 2$ . For a free, finite algebra  $\text{Bool}(S)$  it is easy to show that this is equivalent to a truth function  $S \rightarrow 2$ . Stone duality establishes a contravariant relationship between algebra homomorphisms and (continuous) functions between points. Here, where everything is finite, the Stone topologies are discrete and every function between points induces a homomorphism in the opposite direction.

A PDDL domain consists of two main elements, a set of *predicates*  $P$  and a set of *actions*  $A$ . Each predicate has a list of variables which are (optionally) typed from a fixed set or hierarchy  $T$ , corresponding to an arity function  $\text{ar} : P \rightarrow \text{List}(T)$ . Given a set of typed variables  $\text{tp} : X \rightarrow T$ , the collection of

<sup>5</sup>Stanford Research Institute Problem Solver

atomic statements  $\{p(\bar{x})\}$  can be identified with the pullback  $\text{Atom}(P/X) := P \times_{\text{List}(T)} \text{List}(X)$ . We write  $P(X)$  for the free algebra over  $\text{Atom}(P/X)$ , yielding a functor  $P : \text{Set}/T \rightarrow \text{BoolAlg}$ .

A PDDL action  $a \in A$  consists of a set of typed *parameters*  $X_a$  along with pre- and post-conditions  $q_0^a, q_1^a \in P(X_a)$ . Because  $P(X_a)$  is free, this induces an associated homomorphism  $a : P(X_a)/q_1^a \rightarrow P(X_a)/q_0^a$ . This is easiest to see by defining a dual function on points, regarded as truth functions  $s : \text{Atom}(P/X_a) \rightarrow 2$ :

$$a(s) : p(\bar{x}) \mapsto \begin{cases} \neg s(p(\bar{x})) & \text{if } p(\bar{x}) \wedge q_1^a = \perp \\ s(p(\bar{x})) & \text{otherwise} \end{cases} \quad (24)$$

In other words, we flip any bits that are inconsistent with the post-condition, and leave everything else alone.

From the actions  $A$ , we construct a pair of categories  $\mathcal{B}$  and  $\mathcal{C}$  along with a span of functors  $\mathcal{C} \leftarrow \mathcal{B} \xrightarrow{I} \text{el}(\text{Pt})$ , where  $\text{el}(\text{Pt})$  is the category of elements of the functor  $\text{Pt} : \text{BoolAlg}^{\text{op}} \rightarrow \text{Set}$  sending a Boolean algebra to its set of points. Moreover, the functor  $\mathcal{B} \rightarrow \mathcal{C}$  is a “partial opfibration,” in that lifts of arrows are unique if they exist.

Both  $\mathcal{B}$  and  $\mathcal{C}$  are free symmetric monoidal. In  $\mathcal{B}$ , the objects are Boolean points, and we include one generator  $a_s : s \rightarrow a(s)$  for each point  $s \models q_0^a$ . By contrast,  $\mathcal{C}$  is generated by the objects  $X_a$  (for  $a \in A$ ) and contains a generator morphism  $a : X_a \rightarrow X_a$  for each  $a \in A$ . The functor  $\mathcal{B} \rightarrow \mathcal{C}$  forgets about the pre- and post-conditions that appear in  $\mathcal{B}$ .

There is an obvious functor  $I : \mathcal{B} \rightarrow \text{el}(\text{Pt})$  sending  $a_s$  to the point function defined in (24). We note that  $I$  does not preserve the monoidal structure because a global state over two collections  $X$  and  $Y$  will include predicates with variables from both collections. However,  $I$  is *oplax* monoidal: a global state over  $X$  and  $Y$  entails global states on  $X$  and  $Y$  individually, corresponding to a function  $I(X \otimes Y) \rightarrow IX \times IY$ .

A PDDL problem is defined relative to a domain, and specifies a set of typed variables (called *objects*)  $O$ . In addition, a problem provides initial and goal states, formulated as propositions  $q_0^*, q_1^* \in P(O)$ . We can regard these as a pair of (category-theoretic) objects in  $\mathcal{B}$ .

Finally, a PDDL solution is given as a sequence of (validly) applied actions which, when applied to any initial state  $s_0 \models q_0^*$ , result in a final state  $s_1 \models q_1^*$ . Here an “applied action” for a state  $s \in P(O)$  consists of an action  $a$  and a function  $j : X_a \rightarrow O$ . This induces a function  $j^* : \text{Pt}(P(O)) \rightarrow \text{Pt}(P(X_a))$ , and for an application to be valid we should have  $j^*s \models q_0^a$ .

Because  $P(O)$  is free, we can use the principle of minimal modification to lift the local transformation  $a_{j^*s}$  to a global map  $a_s^j : \text{Pt}(P(O)) \rightarrow \text{Pt}(P(O))$  satisfying  $j^*(a_s^j(s)) \models q_1^a$ ; i.e., the new global state satisfies the local post-condition for  $a$ . Given a sequence of applied actions  $(a_i, j_i)$  we can iteratively construct a sequence of states  $s_i$  and, if each application is valid for the previous state, this will define a string diagram in  $\mathcal{B}$ .

We can relate our Boolean semantics and the attribute syntax introduced earlier by associating any free attribute category with a PDDL domain. The types of the domain are the atomic entities and data services of the syntax. Predicates are defined from the attributes. Given two attributes over a shared data service, say with underlying comonoid actions  $\phi : M \rightarrow M \otimes D$  and  $\psi : N \rightarrow N \otimes D$ , we introduce a binary predicate  $D^{\phi, \psi}(M, N)$  indicating whether or not their data values agree. (Based on our semantic intuition from before,  $D^{\phi, \psi}(M, N)$  should be true precisely when the  $\chi$  morphism associated to these attributes is defined.)

Actions of the PDDL domain correspond to generating morphisms of the syntax, and the equational axioms attached to a morphism define the pre- and post-conditions of the action. For example, the equation shown in (1) corresponds to a post-condition  $L^{\phi, \delta}(R, L)$  on the **MoveTo** action, equating the location attribute of the robot with the copy “attribute” of the target location.

### 3.2 Geometric Semantics

Here we describe a “geometric semantics category”  $\text{Geom}$  and its connections to categories with attributes and modeling. The category  $\text{Geom}$  provides a lower-level semantic counterpart to the Boolean approach outlined above. In addition, it yields a “physical” approach to understanding the behaviors specified by a category with attributes. Informally, objects in  $\text{Geom}$  are physical objects together with data values, while morphisms in  $\text{Geom}$  are (partially defined) paths of the physical objects and update rules for the corresponding values.

Formally, an object  $X = (\{X_i\}, P_X, \theta_X)$  in  $\text{Geom}$  consists of a sequence  $\{X_i\}_{i=1}^{k_X}$  of subsets of  $\mathbb{R}^3$  where each  $X_i$  is called a *simple object*, a topological space  $P_X$  called the *parameter space*, and a continuous function  $\theta_X = (\theta_{X,i}) : P_X \rightarrow \text{SE}(3)^{k_X}$ , called the *structure map*,<sup>6</sup> such that for all  $p \in P_X$ , the sets  $\theta_{X,i}(p) \cdot U_i$  are pairwise disjoint. In general, if  $X$  is an object in  $\text{Geom}$ , we will write  $\{X_i\}_{i=1}^{k_X}$  for the simple objects of  $X$ ,  $P_X$  for the parameter space of  $X$ , and  $\theta_X$  for the structure map of  $X$ . We use similar notation when the object is instead called  $Y$  or anything else.

A good example object to keep in mind is a multi-jointed robot arm. Such an arm can be modeled as a finite collection of rods, the positions of which vary based on parameters (e.g. Euler angles). The condition imposed on the structure map ensures that, regardless of the parameters, no two rods ever occupy the same point in space at once.

A morphism  $f : X \rightarrow Y$  in  $\text{Geom}$ , where  $X$  and  $Y$  have the same simple objects (in the same order), consists of a continuous partial function  $\Phi_f : P_X \rightarrow P_Y$  and a continuous partial function  $\phi_f : P_X \times [0, T_f] \rightarrow \text{SE}(3)^{k_X}$  (for some  $T_f \geq 0$ ), such that:

- For all  $p \in P$  and  $t \in [0, T_f]$ ,  $\phi_f(p, t)$  is defined if and only if  $\Phi_f(p)$  is defined;
- For all  $p \in P$  such that  $\Phi_f(p)$  is defined:
  - $\phi_f(p, 0) = \theta_X(p)$ ;
  - $\phi_f(p, T_f) = \theta_Y(\Phi_f(p))$ ;
  - For all  $t \in [0, T_f]$ , the sets  $\pi_i(\phi_f(p, t)) \cdot U_i$  are pairwise disjoint.

If  $X$  and  $Y$  are two objects of  $\text{Geom}$  such that  $X$  and  $Y$  do not have the same sequence of simple objects, then there are no morphisms between  $X$  and  $Y$ .

Morphisms  $f : X \rightarrow Y$  and  $g : Y \rightarrow Z$ , specified using the pattern established above, can be composed as follows. We set  $\Phi_{g \circ f} = \Phi_g \circ \Phi_f$  (composition of partial functions) and define  $\phi_{g \circ f}$  by:

$$\phi_{g \circ f}(p, t) = \begin{cases} \phi_f(p, t) & t \in [0, T_f] \text{ and } \Phi_{g \circ f}(p) \text{ is defined} \\ \phi_g(\Phi_f(p), t - T_f) & t \in [T_f, T_f + T_g] \text{ and } \Phi_{g \circ f}(p) \text{ is defined.} \end{cases}$$

It is clear that identity morphisms exist and that composition is associative, so  $\text{Geom}$  is in fact a category. In fact,  $\text{Geom}$  is a Poset-enriched category, where  $f \leq g$  if and only if  $T_f = T_g$  and  $\Phi_f \leq \Phi_g$  and  $\phi_f \leq \phi_g$  as partial functions.

The category  $\text{Geom}$  admits a useful notion of “tensor product of objects,” defined as follows. Let  $X, Y \in \text{Ob}(\text{Geom})$ ; then  $X \otimes Y$  is defined to be the object  $(\{X_i\}_{i=1}^{k_X} \sqcup \{Y_j\}_{j=1}^{k_Y}, P_X \otimes P_Y, (\theta_X \times \theta_Y)|_{P_X \otimes P_Y})$ , where  $P_X \otimes P_Y$  is the subspace of  $P_X \times P_Y$  given by the set:

$$\{(p_x, p_y) \in P_X \times P_Y : \forall i, j, (\theta_{X,i}(p_x) \cdot X_i) \cap (\theta_{Y,j}(p_y) \cdot Y_j) = \emptyset\}.$$

---

<sup>6</sup>Here  $\text{SE}(3)$  is the group of rigid motions in  $\mathbb{R}^3$ .

In other words,  $P_X \otimes P_Y$  is the largest subset  $P$  of  $P_X \times P_Y$  for which  $(\{U_i\}_{i=1}^{k_X} \cup \{V_j\}_{j=1}^{k_Y}, P, (\theta_X \times \theta_Y)|_P)$  is a valid object of  $\text{Geom}$ . It is not clear how to extend or adapt this notion to allow for “tensor products of morphisms,” which would equip  $\text{Geom}$  with a symmetric monoidal structure or something of that sort.<sup>7</sup>

We can specify the geometric semantics of a given category with attributes  $\mathcal{C}$  by defining a functor  $F : \mathcal{C} \rightarrow \text{Geom}$ . (This  $F$  may be Poset-enriched if the attribute structure on  $\mathcal{C}$  is.) In order to construct such an  $F$ , we would need models of all the relevant objects and processes in  $\mathcal{C}$ . Hence, instead of outright defining any such  $F$  here, we describe properties  $F$  should satisfy.

We expect that  $F$  sends an entity  $A \in \text{Ob } \mathcal{C}$  to an object  $F(A) \in \text{Geom}$  with simple objects consisting of the component parts of some physical model of  $A$ , parameter space describing the possible states of  $A$ , and structure map attaching to each state a physical configuration of the component parts. This approach works even if  $A$  is a value; in this case,  $F(A)$  has no simple objects ( $k_{F(A)} = 0$ ) and so is described entirely by its parameter space.

The functor  $F$  sends a morphism  $f : A \rightarrow B$  to a physical model of the process specified by  $f$ . Specifically, given a state  $p \in P_{F(A)}$ , the map  $\Phi_{F(f)}$  sends  $p$  to the state achieved by applying  $f$  to a system in the state  $p$  (if such a state exists), while  $\phi_{F(f)}$  describes the physical motion needed to accomplish this change in state.

When  $A$  is a value / data service, the morphisms  $\mu_A, \delta_A, \varepsilon_A$  making  $A$  into a data service would typically be sent to morphisms much like those of the canonical data service structures on objects of  $\text{PartFn}$ . For example,  $\Phi_{F(\delta_A)}$  might be the partial function  $P_A \rightarrow P_A \times P_A$  given by  $p \mapsto (p, p)$ ; in this case, we would have  $T_{F(\delta_A)} = 0$  and  $\phi_{F(\delta_A)}(p, 0) = \Phi_{F(\delta_A)}(p) = (p, p)$ . The image of the natural transformation  $\gamma$  from the attribute structure could be defined in a similar way.

We can instantiate the discussion at the beginning of this section in the context of geometric semantics as follows. We have a canonical functor  $\text{Geom} \rightarrow \text{PartFn}$  (the category of sets and partial functions) defined by  $X \mapsto P_X$  and  $f \mapsto \Phi_f$ . Composing this functor with the standard equivalence  $\text{PartFn} \simeq \text{Set}_*$  (the category of pointed sets and point-preserving maps) and the forgetful functor  $\text{Set}_* \rightarrow \text{Set}$  yields a forgetful functor  $G : \text{Geom} \rightarrow \text{Set}$ .

Given  $F$  and  $G$  as above, the functor  $S = F \circ G : \text{Geom} \rightarrow \text{Set}$  sends  $X \in \text{Ob } \mathcal{C}$  to the parameter space of some model of  $X$  and a morphism to its action on such parameter spaces. If we consider the state of an object to be entirely determined by a point in its parameter space (and also allow the existence of an “undefined” state for each object of  $\mathcal{C}$ ), then the category of elements  $\text{el}(S)$  consists of states of objects in  $\mathcal{C}$  and processes in  $\mathcal{C}$  transforming the domain state into the codomain state.

## 4 Conclusion

In this work, we have presented a categorical interpretation of the notion of an “attribute” of an object. We have also discussed example applications of this interpretation to the categorical modeling of robotics, illustrating how categorical perspectives can be used for the benefit of engineering and other applied fields.

Several interesting questions remain about categorical notions of “attributes.” For example, it appears that several of the axioms for Poset-enriched categories with elements break down when considered with respect to subprobabilistic semantics, e.g. semantics valued in the category  $\text{SRel}$  of [8], leading one to wonder how the approach here might be modified for compatibility with such semantics. In addition, it

---

<sup>7</sup>The difficulty lies in the interchange law  $(f_1 \otimes g_1) \circ (f_2 \otimes g_2) = (f_1 \cdot f_2) \otimes (g_1 \circ g_2)$ ; when morphisms represent paths (as above), rather than paths up to some suitable notion of homotopy or reparametrization, this law does not hold for “obvious” definitions of  $\otimes$ .

is sometimes useful to think of time as an attribute of an object, although there are enough differences between time and the other attributes discussed here that it seems that a separate formalism might work better for modeling time. We hope that future work in this area will be able to resolve some of these questions.

## Acknowledgments

The authors would like to thank Angeline Aguinaldo, Blake Pollard, Fred Proctor, and Eswaran Subrahmanian for helpful comments and discussions. Thanks are also due to the anonymous reviewers for comments that helped the authors improve and clarify aspects of this paper.

The second named author would also like to thank the organizers of the Applied Category Theory 2020 Adjoint School for providing an excellent environment in which to gain a deeper understanding of category theory. In particular, special thanks are due to Paolo Perrone, whose references to categories of elements during meetings of the School helped the second author to realize the role that such constructions played in the semantics of categories with attributes.

## Disclaimer

This paper includes contributions from the U. S. National Institute of Standards and Technology, and is not subject to copyright in the United States. Commercial products are identified in this article to adequately specify the material. This does not imply recommendation or endorsement by the National Institute of Standards and Technology, nor does it imply the materials identified are necessarily the best available for the purpose.

## References

- [1] Lowell Abrams (1999): *Modules, comodules, and cotensor products over Frobenius algebras*. *Journal of Algebra* 219(1), pp. 201–213, doi:10.1006/jabr.1999.7901.
- [2] Angeline Aguinaldo, Spencer Breiner, John S Nolan & Blake S Pollard: *Robot planning with string diagrams*. In preparation.
- [3] Filippo Bonchi, Dusko Pavlovic & Paweł Sobociński (2017): *Functorial semantics for relational theories*. *arXiv preprint arXiv:1711.08699*. Available at <https://arxiv.org/abs/1711.08699>.
- [4] Bob Coecke & Aleks Kissinger (2017): *Picturing quantum processes*. Cambridge University Press, doi:10.1017/9781316219317.
- [5] Brendan Fong & David I. Spivak (2019): *Seven Sketches in Compositionality: An Invitation to Applied Category Theory*. Cambridge University Press, doi:10.1017/9781108668804. Available at <https://math.mit.edu/~dspivak/teaching/sp18/7Sketches.pdf>.
- [6] Peter T Johnstone (1982): *Stone spaces*. *Cambridge studies in advanced mathematics* 3, Cambridge University Press.
- [7] Drew McDermott, Malik Ghallab, Adele Howe, Craig Knoblock, Ashwin Ram, Manuela Veloso, Daniel Weld & David Wilkins (1998): *PDDL-the planning domain definition language*. Technical Report CVC TR-98-003/DCS TR-1165, Yale Center for Computational Vision and Control. Available at <http://icaps-conference.org/ipc2008/deterministic/data/mcdermott-et-al-tr-1998.pdf>.
- [8] Prakash Panangaden (1998): *Probabilistic relations*. *School of Computer Science Research Reports - University of Birmingham CSR*, pp. 59–74.

- [9] Dusko Pavlovic (2013): *Monoidal computer I: Basic computability by string diagrams*. *Information and computation* 226, pp. 94–116, doi:10.1016/j.ic.2013.03.007.