A Categorical Account for the Specification of Replicated Data Type

Fabio Gadducci
Dipartimento di Informatica, Università di Pisa, Italia
fabio.gadducci@unipi.it

Hernán Melgratti
Departamento de Computación, Universidad de Buenos Aires, Argentina
ICC-CONICET-UBA, Argentina
hmelgra@dc.uba.ar

Christian Roldán
Departamento de Computación, Universidad de Buenos Aires, Argentina
croldan@dc.uba.ar

Matteo Sammartino
Department of Computer Science, University College London, UK
m.sammartino@ucl.ac.uk

Abstract

Replicated Data Types (RDTs) have been introduced as suitable abstractions for dealing with weakly consistent data stores, which may (temporarily) expose multiple, inconsistent views of their state. In the literature, RDTs are commonly specified in terms of two relations: visibility, which accounts for the different views that a store may have, and arbitration, which states the logical order imposed on the operations executed over the store. Different flavours, e.g., operational, axiomatic and functional, have recently been proposed for the specification of RDTs. In this work, we propose a categorical characterisation of RDT specifications. We define categories of visibility relations and arbitrations, show the existence of relevant limits and colimits, and characterize RDT specifications as functors between such categories that preserve these additional structures.

2012 ACM Subject Classification General and reference → General literature; General and reference

Keywords and phrases Replicated data type, Specification, Functorial characterisation

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2019.23

1 Introduction

The CAP theorem establishes that a distributed data store can simultaneously provide two of the following three properties: consistency, availability, and tolerate network partitions [8]. A weakly consistent data store prioritises availability and partition tolerance over consistency. As a consequence, a weakly consistent data store may (temporarily) expose multiple, inconsistent views of its state; hence, the behaviour of operations may depend on the particular view over which they are executed. Replicated data types (RDT) have been proposed as suitable data type abstractions for weakly consistent data stores. The specification of such data types usually takes into account the particular views over which operations are executed. A view is usually represented by a *visibility* relation, which is a binary, acyclic relation over executed operations (a.k.a. *events*). The state of a store is described instead as total order over events, called *arbitrations*, which describes the way in which conflicting concurrent operations are resolved. Different specification approaches for RDTs build on the notions of visibility and arbitration [2, 3, 4, 5, 7, 9, 11, 13, 14]. A purely functional approach for the specification of RDTs has been presented in [7], where an RDT is associated with a function that maps visibility into set of arbitrations.

$$\mathcal{S}_{Ctr}\left(\begin{array}{c}\langle \texttt{inc},\texttt{ok}\rangle\\ \downarrow\\ \langle\texttt{rd},1\rangle\end{array}\right) = \left\{ \begin{array}{cc} \langle \texttt{inc},\texttt{ok}\rangle & \langle \texttt{rd},1\rangle \\ | & | \\ \langle \texttt{rd},1\rangle & \langle \texttt{inc},\texttt{ok}\rangle\end{array}\right\}$$

(a) Specification of a `Counter`

$$\mathcal{S}_{Ctr}\left(\begin{array}{c}\langle \texttt{inc},\texttt{ok}\rangle\\ \downarrow\\ \langle\texttt{rd},0\rangle\end{array}\right) = \emptyset$$

(b) Non-admissible arbitrations

Figure 1 Counter specifications

For illustration purposes, consider the RDT `Counter`, which mantains an integer value and provides two operations for incrementing and reading its current value. The specification of `Counter` states that every increment is always successful, while the expected result for a read operation is the number of increments seen by that read, regardless of the order in which such operations are arbitrated. Following the approach in [7], the RDT `Counter` is specified as a function $\mathcal{S}_{Ctr}$ that maps visibility relations into sets of arbitrations. For instance, Figure 1a illustrates the case for a visibility relation that involves an increment event seen by a read event: events are depicted by pairs $\langle \texttt{operation}, \texttt{expected\_result}\rangle$, and `inc` stands for increment and `rd` for read. Note that the expected result for the read event is `1`, which coincides with the quantity of events labelled by $\langle \texttt{inc}, \texttt{ok}\rangle$ seen by that read. The function $\mathcal{S}_{Ctr}$ maps that visibility relation into a set containing two arbitrations of the events, i.e., two total orders, for the events of the visibility relation. We remark that arbitration does not mean real time ordering, but just a way in which a store can totally order events, which may not respect the causal order of operations. In fact, the second arbitration orders the read event before the increment one, despite the first event causally depends on the second one. Figure 1b shows instead a case in which the specification maps a visibility relation into an empty set of arbitrations, which means that such visibility relation is not accepted by the specification. Basically, that visibility is rejected because there is a read event that sees an increment event but returns 0 instead of the expected result 1.

This work develops the approach suggested in [7] for the categorical characterisation of RDT specifications. We consider the category $\mathbf{PIDag}(\mathcal{L})$ of labelled, directed acyclic graphs and *pr-morphisms*, i.e., label-preserving morphisms that reflect directed edges, and the category $\mathbf{SPath}(\mathcal{L})$ of sets of labelled, total orders and *ps-morphisms*, i.e., morphisms between set of paths. A ps-morphism $\mathtt{f} : \mathcal{X}_1 \to \mathcal{X}_2$ from a set of paths $\mathcal{X}_1$ to a set of path $\mathcal{X}_2$ states that any total order in $\mathcal{X}_2$ can be obtained by extending some total order in $\mathcal{X}_1$. In this work we show that a large class of specifications, dubbed *iso-coherent*, can be characterised functorially. Roughly, a coherent specification accounts for those RDTs such that the arbitrations associated with a visibility relation can be obtained by extending arbitrations associated with "smaller" visibilities. An iso-coherent specification is a coherent specification that maps isomorphic graphs into isomorphic sets of paths. We establish a bijection between functors and specifications, showing that an iso-coherent specification induces a functor from $\mathbf{PIDag}(\mathcal{L})$ into $\mathbf{SPath}(\mathcal{L})$ that preserves colimits and binary pullbacks and vice versa.

The paper has the following structure. Section 2 offers some preliminaries on categories of relations, which are used for proposing some basic results on categories of graphs and paths in Section 3. Section 4 recalls the set-theoretical presentation of RDTs introduced in [7]. Section 5 introduces our semantical model, the category of set of paths, describing some of its basic properties with respect to limits and colimits. On Section 6 we presents some categorical operators for RDTs, which are used in Section 7 to present our main characterisation results. The paper is closed with some final remarks and some hints towards future works.

2 Preliminaries on Relations

Relations. Given a set $\mathtt{E}$, a (binary) *relation* over $\mathtt{E}$ is a sub-set $\rho \subseteq \mathtt{E} \times \mathtt{E}$ of the cartesian product of $\mathtt{E}$ with itself. We write $\langle \mathtt{E}, \rho \rangle$ for a relation over $\mathtt{E}$, and $\emptyset$ to denote the empty relation. The downward closure of $\mathtt{E}' \subseteq \mathtt{E}$ is a set such that $\forall \mathtt{e} \in \mathtt{E}, \mathtt{e}' \in \mathtt{E}'. \mathtt{e} \; \rho \; \mathtt{e}'$ implies $\mathtt{e} \in \mathtt{E}'$.

In addition, we write $\lfloor \mathtt{e} \rfloor$ to stand for the downward closure of a single element e.

▶ **Definition 1** ((Binary Relation) Morphisms). *A (binary relation) morphism* $\mathtt{f} : \langle \mathtt{E}, \rho \rangle \to \langle \mathtt{T}, \gamma \rangle$ *is a function* $\mathtt{f} : \mathtt{E} \to \mathtt{T}$ *such that*

$$\forall \mathtt{e}, \mathtt{e}' \in \mathtt{E}. \mathtt{e} \; \rho \; \mathtt{e}' \textit{ implies } \mathtt{f}(\mathtt{e}) \; \gamma \; \mathtt{f}(\mathtt{e}')$$

A morphism $\mathtt{f} : \langle \mathtt{E}, \rho \rangle \to \langle \mathtt{T}, \gamma \rangle$ *is* past-reflecting *(shortly, pr-morphism) if*

$$\forall \mathtt{e} \in \mathtt{E}, \mathtt{t} \in \mathtt{T}. \; \mathtt{t} \; \gamma \; \mathtt{f}(\mathtt{e}) \textit{ implies } \exists \mathtt{e}' \in \mathtt{E}. \mathtt{e}' \; \rho \; \mathtt{e} \; \wedge \; \mathtt{t} = \mathtt{f}(\mathtt{e}')$$

Note that both classes of morphisms are closed under composition: we denote as **Rel** the category of relations and their morphisms and **PRel** the sub-category of pr-morphisms.

▶ **Lemma 2** (Characterising pr-morphisms). *Let* $\mathtt{f} : \langle \mathtt{E}, \rho \rangle \to \langle \mathtt{T}, \gamma \rangle$ *be a morphism. If it is order-reflecting and downward closed, that is*

1. $\mathtt{f}(\mathtt{e}) \; \gamma \; \mathtt{f}(\mathtt{e}')$ *implies* $\mathtt{e} \; \rho \; \mathtt{e}'$
2. $\bigcup_{\mathtt{e} \in \mathtt{E}} \mathtt{f}(\mathtt{e})$ *is downward closed,*

then it is a pr-morphism. If $\mathtt{f}$ *is injective, then the vice-versa holds.*

Clearly, **Rel** has both finite limits and finite colimits, which are computed point-wise as in **Set**. The structure is largely lifted to **PRel**.

▶ **Proposition 3** (Properties of **PRel**). *The inclusion functor* **PRel** $\to$ **Rel** *reflects finite colimits and binary pullbacks.*

In other words, since **Rel** has finite limits and finite colimits, finite colimits and binary pullbacks in **PRel** always exist and are computed as in **Rel**. There is no terminal object, since morphisms in **Rel** into the singleton are clearly not past-reflecting.

Monos in **Rel** are just morphisms whose underlying function is injective, and similarly in **PRel**, so that the inclusion functor preserves (and reflects) them.

▶ **Lemma 4** (Monos under pushouts). *Pushouts in* **Rel** *(and thus in* **PRel***) preserve monos.*

We now introduce labelled relations. Consider the forgetful functors $\mathtt{U_r} : \mathbf{Rel} \to \mathbf{Set}$ and $\mathtt{U_p} : \mathbf{PRel} \to \mathbf{Set}$, the latter factoring through the inclusion functor **PRel** $\to$ **Rel**. Chosen a set $\mathcal{L}$ of labels, we consider the comma categories $\mathbf{Rel}(\mathcal{L}) = \mathtt{U_r} \downarrow \mathcal{L}$ and $\mathbf{PRel}(\mathcal{L}) = \mathtt{U_p} \downarrow \mathcal{L}$: it is well known that all the relevant structure is preserved in such comma categories.

Explicitly, an object in $\mathtt{U_r} \downarrow \mathcal{L}$ is a triple $(\mathtt{E}, \rho, \lambda)$ for a labeling function $\lambda : \mathtt{E} \to \mathcal{L}$. A label-preserving morphism $(\mathtt{E}, \rho, \lambda) \to (\mathtt{E}', \rho', \lambda')$ is a morphism $\mathtt{f} : (\mathtt{E}, \rho) \to (\mathtt{E}', \rho')$ such that $\forall \mathtt{s} \in \mathtt{E}. \lambda(\mathtt{s}) = \lambda'(\mathtt{f}(\mathtt{s}))$. Moreover, finite limits and finite colimits are computed as in **Rel**. The same characterisation also holds for the objects and the morphisms of $\mathtt{U_p} \downarrow \mathcal{L}$.

3 Categories of Graphs and Paths

We now move to introduce specific sub-categories that are going to be used for both the syntax and the semantics of specifications.

▶ **Definition 5** (Directly acyclic graphs category). **PDag** *is the full sub-category of* **PRel** *whose objects are directed acyclic graphs.*

In other terms, objects are relations whose transitive closure is a *strict* partial order.

▶ Remark 6. The category whose arrows are morphisms is not that interesting, categorically speaking, because, e.g., it does not admit pushouts, not even along monos. The one with pr-morphisms is much more so, still remaining computationally simple.

▶ **Proposition 7** (Properties of **PDag**). *The inclusion functor* **PDag** $\to$ **PRel** *reflects finite colimits and binary pullbacks.*

We now move to consider *paths*, i.e., relations that are total orders.

▶ **Definition 8** (Paths Category). **Path** *is the full sub-category of* **Rel** *whose objects are paths.*

Note that the sub-category of just pr-morphisms is not so relevant, since there exists a pr-morphism between two paths if and only if one path is a prefix of the other.

▶ **Proposition 9** (Properties of **Path**). *The inclusion functor* **Path** $\to$ **Rel** *reflects finite colimits.*

As for relations, we consider suitable comma categories in order to capture labelled paths and graphs. In particular, we use the forgetful functors $\mathtt{U_{rp}} : \mathbf{Path} \to \mathbf{Set}$ and $\mathtt{U_{pd}} : \mathbf{PDag} \to \mathbf{Set}$: for a set of labels $\mathcal{L}$ we denote $\mathbf{PDag}(\mathcal{L}) = \mathtt{U_{rp}} \downarrow \mathcal{L}$ and $\mathbf{Path}(\mathcal{L}) = \mathtt{U_{pd}} \downarrow \mathcal{L}$. Once more, the relevant categorical structure is preserved and computed as in **Rel**.

4 Replicated Data Type Specification

We briefly recall the set-theoretical model of replicated data types (RDT), introduced in [7]. Our main result is its categorical characterisation, which is given in the following sections.

First, some notation. We denote a graph as $\langle \mathcal{E}, \prec, \lambda \rangle$ and a path as $\langle \mathcal{E}, \leq, \lambda \rangle$, in order to distinguish them. Moreover, given a graph $\mathtt{G} = \langle \mathcal{E}, \prec, \lambda \rangle$ and a subset $\mathcal{E}' \subseteq \mathcal{E}$, we denote by $\mathtt{G}|_{\mathcal{E}'}$ the obvious restriction (and the same for a path $\mathtt{P}$).

We now define a product operation on a set of paths $\mathcal{X} = \{\langle \mathcal{E}_i, \leq_i, \lambda_i \rangle\}_i$. We require that paths in $\mathcal{X}$ are *compatible*, i.e., $\forall \mathsf{e}, i, j. \mathsf{e} \in \mathcal{E}_i \cap \mathcal{E}_j$ implies $\lambda_i(\mathsf{e}) = \lambda_j(\mathsf{e})$.

▶ **Definition 10** (Product). *Let* $\mathcal{X}$ *be a set of compatible paths. The product of* $\mathcal{X}$ *is*

$$\bigotimes \mathcal{X} = \{\mathtt{P} \mid \mathtt{P} \;\; \textit{is a path over} \; \bigcup_i \mathcal{E}_i \;\textit{and}\; \mathtt{P}|_{\mathcal{E}_i} \in \mathcal{X} \;\}$$

Intuitively, the product of paths is analogous to the synchronous product of transition systems, in which common elements are identified and the remaining ones can be freely interleaved, as long as the original orders are respected. A set of sets of paths $\mathcal{X}_1, \mathcal{X}_2, \ldots$ is compatible if $\bigcup_i \mathcal{X}_i$ is so. In such case we can define the product $\bigotimes_i \mathcal{X}_i$ as $\bigotimes \bigcup_i \mathcal{X}_i$.

Now, let us further denote with $\mathbb{G}(\mathcal{L})$ and $\mathbb{P}(\mathcal{L})$ the sets of graphs and paths, respectively, labelled over $\mathcal{L}$ and with ϵ the empty graph. Also, when the set of labels $\mathcal{L}$ is chosen, we let $\mathbb{G}(\mathcal{E}, \lambda)$ and $\mathbb{P}(\mathcal{E}, \lambda)$ the sets of graphs and paths, respectively, whose elements are those in $\mathcal{E}$ and are labelled by $\lambda : \mathcal{E} \to \mathcal{L}$.

▶ **Definition 11** (Specifications). *A* specification $\mathcal{S}$ *is a function* $\mathcal{S} : \mathbb{G}(\mathcal{L}) \to 2^{\mathbb{P}(\mathcal{L})}$ *such that* $\mathcal{S}(\epsilon) = \{\epsilon\}$ *and* $\forall \mathtt{G}.\ \mathcal{S}(\mathtt{G}) \in 2^{\mathbb{P}(\mathcal{E}_\mathtt{G}, \lambda_\mathtt{G})}$.

In other words, a specification $\mathcal{S}$ maps a graph (interpreted in terms of the visibility relation of a RDT) to a set of paths (that is, the admissible arbitrations of the RDT). Indeed, note that $\mathtt{P} \in \mathcal{S}(\mathtt{G})$ is a path over $\mathcal{E}_\mathtt{G}$, hence a total order of the events in $\mathtt{G}$.

As shown in [7], Definition 11 offers an alternative characterisation of RDTs [4] for a suitable choice of the set of labels. In particular, an RDT boils down to a specification labelled over pairs $\langle operation, value\rangle$ that is *saturated* and *past-coherent*. The former property is a technical one: roughly, if $\mathtt{G}'$ is an extension of $\mathtt{G}$ with an fresh event $\mathtt{e}$, then the admissible arbitrations that a saturated specification $\mathcal{S}$ assigns to $\mathtt{G}'$ (i.e., the set of paths $\mathcal{S}(\mathtt{G}')$) are the admissible arbitrations of $\mathtt{G}$ saturated with respect to $\mathtt{e}$, i.e., all the paths that extends a path in $\mathcal{S}(\mathtt{G})$ with $\mathtt{e}$ inserted at an arbitrary position. Coherence instead is fundamental and expresses that admissible arbitrations of a visibility graph can be obtained by composing the admissible arbitrations of smaller visibilities.

▶ **Definition 12** ((Past-)Coherent Specification). *Let $\mathcal{S}$ be a specification. We say that $\mathcal{S}$ is past-coherent (briefly, coherent) if*

$$\forall \mathtt{G} \neq \epsilon.\ \mathcal{S}(\mathtt{G}) = \bigotimes_{\mathtt{e}\in\mathcal{E}_\mathtt{G}} \mathcal{S}(\mathtt{G}|_{\lfloor \mathtt{e}\rfloor})$$

Explicitly, in a coherent specification $\mathcal{S}$ the arbitrations of a configuration $\mathtt{G}$ (i.e., the set of paths $\mathcal{S}(\mathtt{G})$) are the composition of the arbitrations associated with its sub-graphs $\mathtt{G}|_{\lfloor \mathtt{e}\rfloor}$.

Next example illustrates a saturated and coherent specification for the `Counter` RDT.

▶ **Example 13** (Counter). Fix the following set of labels: $\mathcal{L} = \{\langle \mathtt{inc}, \mathtt{ok}\rangle\} \cup (\{\mathtt{rd}\} \times \mathbb{N})$. Then, the specification of the RDT `Counter` is given by the function $\mathcal{S}_{Ctr}$ defined such that

$$\begin{array}{c} \mathtt{P} \in \mathcal{S}_{Ctr}(\mathtt{G}) \\ \text{iff} \\ \forall \mathtt{e} \in \mathcal{E}_\mathtt{G}.\forall \mathtt{k}.\lambda(\mathtt{e}) = \langle \mathtt{rd}, \mathtt{k}\rangle \ \textit{implies}\ \mathtt{k} = \#\{\mathtt{e}' \mid \mathtt{e}' \prec_\mathtt{G} \mathtt{e} \text{ and } \lambda(\mathtt{e}') = \langle \mathtt{inc}, \mathtt{ok}\rangle\} \end{array}$$

Intuitively, a visibility graph $\mathtt{G}$ is mapped to a non-empty set of arbitrations (i.e., $\mathcal{S}_{Ctr}(\mathtt{G}) \neq \emptyset$) only when each event $\mathtt{e}$ in $\mathtt{G}$ associated with a read operation has a return value $\mathtt{k}$ that matches the number of increments preceding $\mathtt{e}$ in $\mathtt{G}$. We remark that this specification is *coherent* and *saturated*. Saturation follows immediately because the definition of $\mathcal{S}_{Ctr}$ does not impose any constraint on the ordering of events for the arbitrations $\mathtt{P}$ in $\mathcal{S}_{Ctr}(\mathtt{G})$. Coherence can be shown as follows. By definition of $\mathcal{S}_{Ctr}$, $\mathtt{P} \in \mathcal{S}_{Ctr}(\mathtt{G})$ implies $\mathtt{P}|_{\lfloor \mathtt{e}\rfloor} \in \mathcal{S}_{Ctr}(\mathtt{G}|_{\lfloor \mathtt{e}\rfloor})$ for all $\mathtt{e} \in \mathcal{E}_\mathtt{G}$. Consequently, $\mathtt{P} \in \bigotimes_{\mathtt{e}\in\mathcal{E}_\mathtt{G}} \mathcal{S}(\mathtt{G}|_{\lfloor \mathtt{e}\rfloor})$. On the contrary, take $\mathtt{P} \in \bigotimes_{\mathtt{e}\in\mathcal{E}_\mathtt{G}} \mathcal{S}(\mathtt{G}|_{\lfloor \mathtt{e}\rfloor})$. Then, $\mathtt{e} \in \mathcal{E}_\mathtt{G}$ implies $\mathtt{e} \in \mathcal{E}_\mathtt{P}$. Moreover, $\mathtt{e} \in \mathcal{E}_\mathtt{P}$ implies $\lambda(\mathtt{e}) = \langle \mathtt{rd}, \mathtt{k}\rangle$ iff $\mathtt{k} = \#\{\mathtt{e}' \mid \mathtt{e}' \prec_\mathtt{G} \mathtt{e}$ and $\lambda(\mathtt{e}') = \langle \mathtt{inc}, \mathtt{ok}\rangle\}$. Hence, $\mathtt{P} \in \mathcal{S}_{Ctr}(\mathtt{G})$. Therefore, the equality in Definition 12 holds.

5 The model category

In order to provide a categorical characterisation of coherent specifications, we must first define precisely the model category. So far, we know that its objects have to be sets of compatible paths. We fix a set of labels $\mathcal{L}$, and we start looking at morphisms.

▶ **Definition 14** (Saturation). *Let $\mathtt{P}$ be a path and $\mathtt{f} : (\mathcal{E}_\mathtt{P}, \lambda_\mathtt{P}) \to (\mathcal{E}, \lambda)$ a function preserving labels. The saturation of $\mathtt{P}$ along $\mathtt{f}$ is defined as*

$$\mathtt{sat}(\mathtt{P}, \mathtt{f}) = \{\mathtt{Q} \mid \mathtt{Q}\ \in \mathbb{P}(\mathcal{E}, \lambda) \textit{ and } \mathtt{f} \textit{ induces a path morphism } \mathtt{f} : \mathtt{P} \to \mathtt{Q}\}$$

The notion of saturation is extended to sets of paths $\mathcal{X} \subseteq \mathbb{P}(\mathcal{E}, \lambda)$ as $\bigcup_{\mathtt{P}\in\mathcal{X}} \mathtt{sat}(\mathtt{P}, \mathtt{f})$.

Note that, should $\mathtt{f}$ not be injective, it could be that $\mathtt{sat}(\mathtt{P}, \mathtt{f}) = \emptyset$.

▶ **Example 15.** Consider the (injective, label-preserving) function $\mathtt{f}$ mapping two events with labels $\{\mathtt{a}, \mathtt{b}\}$ to three events with labels $\{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$. Then we have

$$\mathtt{sat}\left(\begin{array}{c}\mathtt{a}\\ |\\ \mathtt{b}\end{array}, \mathtt{f}\right) = \left\{\begin{array}{c}\mathtt{a}\\ |\\ \mathtt{b}\\ |\\ \mathtt{c}\end{array}, \begin{array}{c}\mathtt{a}\\ |\\ \mathtt{c}\\ |\\ \mathtt{b}\end{array}, \begin{array}{c}\mathtt{c}\\ |\\ \mathtt{a}\\ |\\ \mathtt{b}\end{array}\right\}$$

Intuitively, saturation adds $\mathtt{c}$ – and in general events not in the image of $\mathtt{f}$ – to the original path in all possible ways, preserving the order of original events.

We can exploit saturation to get a simple definition of our model category.

▶ **Definition 16** (ps-morphism). *Let $\mathcal{X}_1 \subseteq \mathbb{P}(\mathcal{E}_1, \lambda_1)$ and $\mathcal{X}_2 \subseteq \mathbb{P}(\mathcal{E}_2, \lambda_2)$ be sets of paths. A path-set morphim (shortly, ps-morphism) $\mathtt{f} : \mathcal{X}_1 \to \mathcal{X}_2$ is a function $\mathtt{f} : (\mathcal{E}_1, \lambda_1) \to (\mathcal{E}_2, \lambda_2)$ preserving labels such that*

$$\mathcal{X}_2 \subseteq \mathtt{sat}(\mathcal{X}_1, \mathtt{f})$$

Intuitively, there is a ps-morphism from the set of paths $\mathcal{X}_1$ to the set of path $\mathcal{X}_2$ if any path in $\mathcal{X}_2$ can be obtained by adding events to some path in $\mathcal{X}_2$. This notion captures the idea that arbitrations of larger visibilities are obtained as extensions of smaller visibilities.

▶ **Example 17.** Consider the following three sets and the function $\mathtt{f}$ from Example 15.

$$\mathcal{X}_1 = \left\{\begin{array}{c}\mathtt{a}\\ |\\ \mathtt{b}\end{array}\right\} \qquad \mathcal{X}_2 = \left\{\begin{array}{c}\mathtt{a}\\ |\\ \mathtt{b}\\ |\\ \mathtt{c}\end{array}, \begin{array}{c}\mathtt{a}\\ |\\ \mathtt{c}\\ |\\ \mathtt{b}\end{array}\right\} \qquad \mathcal{X}_3 = \left\{\begin{array}{c}\mathtt{a}\\ |\\ \mathtt{b}\\ |\\ \mathtt{c}\end{array}, \begin{array}{c}\mathtt{b}\\ |\\ \mathtt{c}\\ |\\ \mathtt{a}\end{array}\right\}$$

$\mathtt{f}$ induces a ps-morphism $\mathtt{f} : \mathcal{X}_1 \to \mathcal{X}_2$ because $\mathcal{X}_2 \subseteq \mathtt{sat}(\mathcal{X}_1, \mathtt{f})$ ($\mathtt{sat}(\mathcal{X}_1, \mathtt{f})$ is depicted in Example 15). On the contrary, there is no ps-morphism from $\mathcal{X}_1$ to $\mathcal{X}_3$ because the second path of $\mathcal{X}_3$ cannot be obtained by extending some path of $\mathcal{X}_1$ with an event labelled by $\mathtt{c}$.

▶ **Definition 18** (Retraction). *Let $\mathtt{Q}$ be a path and $\mathtt{f} : \mathcal{E} \to \mathcal{E}_{\mathtt{Q}}$ a function. The retraction of $\mathtt{Q}$ along $\mathtt{f}$ is defined as*

$$\mathtt{ret}(\mathtt{Q}, \mathtt{f}) = \{\mathtt{P} \mid \mathtt{P} \in \mathbb{P}(\mathcal{E}, \lambda) \textit{ and } \mathtt{f} \textit{ induces a path morphism } \mathtt{f} : \mathtt{P} \to \mathtt{Q}\}$$

The notion of retraction is extended to sets of paths $\mathcal{X} \subseteq \mathbb{P}(\mathcal{E}, \lambda)$ as $\bigcup_{\mathtt{Q} \in \mathcal{X}} \mathtt{ret}(\mathtt{Q}, \mathtt{f})$.

Note that λ is fully characterised as the restriction of $\lambda_{\mathtt{Q}}$ along the mapping. Should $\mathtt{f}$ be injective, $\mathtt{ret}(\mathtt{Q}, \mathtt{f})$ would be a singleton, and if $\mathtt{f}$ is an inclusion, then $\mathtt{ret}(\mathtt{Q}, \mathtt{f}) = \mathtt{Q}|_{\mathcal{E}}$.

We may now start considering the relationship between the two notions.

▶ **Lemma 19.** *Let $\mathcal{X}_1 \subseteq \mathbb{P}(\mathcal{E}_1, \lambda_1)$ be a set of paths and $\mathtt{f} : (\mathcal{E}_1, \lambda_1) \to (\mathcal{E}_2, \lambda_2)$ a function preserving labels. Then $\mathcal{X}_1 \subseteq \mathtt{ret}(\mathtt{sat}(\mathcal{X}_1, \mathtt{f}), \mathtt{f})$. If $\mathtt{f}$ is injective, then the equality holds.*

▶ **Lemma 20.** *Let $\mathcal{X}_2 \subseteq \mathbb{P}(\mathcal{E}_2, \lambda_2)$ be a set of paths and $\mathtt{f} : \mathcal{E}_1 \to \mathcal{E}_2$ a function. Then $\mathcal{X}_2 \subseteq \mathtt{sat}(\mathtt{ret}(\mathcal{X}_2, \mathtt{f}), \mathtt{f})$.*

We say that an injective function $\mathtt{f}$ is *saturated* with respect to $\mathcal{X}_2$ if the equality holds.

▶ **Example 21.** Consider the ps-morphism

$$\mathtt{f}: \left\{ \begin{array}{c} \mathtt{a} \\ | \\ \mathtt{b} \end{array} \right\} \to \left\{ \begin{array}{c} \mathtt{a} \\ | \\ \mathtt{b} \\ | \\ \mathtt{c} \end{array} \right\}$$

whose underlying function is $\mathtt{f}$ from Example 15. This is *not* saturated. In fact, we have

$$\left\{ \begin{array}{c} \mathtt{a} \\ | \\ \mathtt{b} \\ | \\ \mathtt{c} \end{array} \right\} \neq \mathtt{sat}(\mathtt{ret}(\left\{ \begin{array}{c} \mathtt{a} \\ | \\ \mathtt{b} \\ | \\ \mathtt{c} \end{array} \right\}, \mathtt{f}), \mathtt{f}) = \mathtt{sat}(\left\{ \begin{array}{c} \mathtt{a} \\ | \\ \mathtt{b} \end{array} \right\}, \mathtt{f}) = \left\{ \begin{array}{c} \mathtt{a} \\ | \\ \mathtt{b} \\ | \\ \mathtt{c} \end{array}, \begin{array}{c} \mathtt{a} \\ | \\ \mathtt{c} \\ | \\ \mathtt{b} \end{array}, \begin{array}{c} \mathtt{c} \\ | \\ \mathtt{a} \\ | \\ \mathtt{b} \end{array} \right\}$$

▶ **Definition 22** (Sets of Paths Category)**.** *We define* $\mathbf{SPath}(\mathcal{L})$ *as the category whose objects are sets of paths* $\mathcal{X} \subseteq \mathbb{P}(\mathcal{E}, \lambda)$ *and morphisms are ps-morphisms.*

▶ **Proposition 23** (Properties of $\mathbf{SPath}$)**.** *The category* $\mathbf{SPath}(\mathcal{L})$ *has finite colimits along monos and binary pullbacks.*

Proof. *(Strict) initial object.* The choice is $\langle \emptyset, \{\epsilon\}, \emptyset \rangle$, with $\epsilon \in \mathbb{P}(\emptyset, \emptyset)$ the empty path. Let $\mathcal{X} \subseteq \mathbb{P}(\mathcal{E}, \lambda)$ and $! \colon \emptyset \to \mathcal{E}$ the unique function. We have a function $! \colon (\emptyset, \emptyset) \to (\mathcal{E}, \lambda)$ such that $\mathcal{X} \subseteq \mathtt{sat}(\{\epsilon\}, !) = \mathbb{P}(\mathcal{E}, \lambda)$.

Binary Pushouts. Let $\mathcal{X}, \mathcal{X}_1$, and $\mathcal{X}_2$ be sets of paths and $\mathtt{f}_i : \mathcal{X} \to \mathcal{X}_i$ ps-morphisms. Consider the underlying functions $\mathtt{f}_i : \mathcal{E} \to \mathcal{E}_i$ and their pushout $\mathtt{f}'_i : \mathcal{E}_i \to \mathcal{E}_1 +_{\mathcal{E}} \mathcal{E}_2$ in the category of sets. This induces a pushout $\mathtt{f}'_i : \mathcal{X}_i \to \mathtt{sat}(\mathcal{X}_1, \mathtt{f}'_1) \cap \mathtt{sat}(\mathcal{X}_2, \mathtt{f}'_2)$ in $\mathbf{SPath}(\mathcal{L})$.

Binary Pullbacks. Let $\mathcal{X}, \mathcal{X}_1$, and $\mathcal{X}_2$ be sets of paths and $\mathtt{f}_i : \mathcal{X}_i \to \mathcal{X}$ ps-morphisms. Consider the underlying functions $\mathtt{f}_i : \mathcal{E}_i \to \mathcal{E}$ and their pullback $\mathtt{f}'_i : \mathcal{E}_1 \times_{\mathcal{E}} \mathcal{E}_2 \to \mathcal{E}$ in the category of sets. This induces a pullback $\mathtt{f}'_i : \mathtt{ret}(\mathcal{X}_1, \mathtt{f}'_1) \cup \mathtt{ret}(\mathcal{X}_2, \mathtt{f}'_2) \to \mathcal{X}_i$ in $\mathbf{SPath}(\mathcal{L})$. ◀

The characterisation of pushouts might not work, should not be on a span of injective functions. To help intuition, we now instantiate the constructions above to suitable inclusions.

▶ **Lemma 24.** *Let* $\mathtt{f}_i : \mathcal{X} \to \mathcal{X}_i$ *be ps-morphisms such that the underlying functions* $\mathtt{f}_i : \mathcal{E} \to \mathcal{E}_i$ *are inclusions and* $\mathcal{E} = \mathcal{E}_1 \cap \mathcal{E}_2$*. Then the pushout is given by* $\mathtt{f}'_i : \mathcal{X}_i \to \mathcal{X}_1 \otimes \mathcal{X}_2$*.*

Proof. By definition $\mathcal{X}_1 \otimes \mathcal{X}_2 = \{\mathtt{P} \mid \mathtt{P}$ is a path over $\bigcup_i \mathcal{E}_i$ and $\mathtt{P}|_{\mathcal{E}_i} \in \mathcal{X}_i\}$. Note also that $\mathtt{sat}(\mathcal{X}_i, \mathtt{f}'_i) = \bigcup_{\mathtt{Q} \in \mathcal{X}_i} \{\mathtt{P} \mid \mathtt{P} \in \mathbb{P}(\bigcup_i \mathcal{E}_i, \bigcup_i \lambda_i)$ and $\mathtt{f}'_1$ induces a path morphism $\mathtt{f}'_\mathtt{i} : \mathtt{P} \to \mathtt{Q}\}$. Since $\mathtt{f}'_\mathtt{i}$ is an inclusion, the latter condition equals to $\mathtt{P}|_{\mathcal{E}_i} = \mathtt{Q}$, thus the property holds. ◀

▶ **Example 25.** Consider the following ps-morphisms

$$\mathtt{f}_1 : \left\{ \begin{array}{c} \mathtt{a} \\ | \\ \mathtt{b} \end{array}, \begin{array}{c} \mathtt{b} \\ | \\ \mathtt{a} \end{array} \right\} \to \left\{ \begin{array}{c} \mathtt{a} \\ | \\ \mathtt{b} \\ | \\ \mathtt{c} \end{array} \right\} \qquad \mathtt{f}_2 : \left\{ \begin{array}{c} \mathtt{a} \\ | \\ \mathtt{b} \end{array}, \begin{array}{c} \mathtt{b} \\ | \\ \mathtt{a} \end{array} \right\} \to \left\{ \begin{array}{c} \mathtt{a} \\ | \\ \mathtt{b} \\ | \\ \mathtt{d} \end{array}, \begin{array}{c} \mathtt{b} \\ | \\ \mathtt{a} \\ | \\ \mathtt{d} \end{array} \right\}$$

then, the pushout is given by the following two morphisms

$$\mathtt{g}_1 : \left\{ \begin{array}{c} \mathtt{a} \\ | \\ \mathtt{b} \\ | \\ \mathtt{c} \end{array} \right\} \to \left\{ \begin{array}{c} \mathtt{a} \\ | \\ \mathtt{b} \\ | \\ \mathtt{c} \\ | \\ \mathtt{d} \end{array}, \begin{array}{c} \mathtt{b} \\ | \\ \mathtt{a} \\ | \\ \mathtt{d} \\ | \\ \mathtt{c} \end{array} \right\} \qquad \mathtt{g}_2 : \left\{ \begin{array}{c} \mathtt{a} \\ | \\ \mathtt{b} \\ | \\ \mathtt{d} \end{array}, \begin{array}{c} \mathtt{b} \\ | \\ \mathtt{a} \\ | \\ \mathtt{d} \end{array} \right\} \to \left\{ \begin{array}{c} \mathtt{a} \\ | \\ \mathtt{b} \\ | \\ \mathtt{c} \\ | \\ \mathtt{d} \end{array}, \begin{array}{c} \mathtt{b} \\ | \\ \mathtt{a} \\ | \\ \mathtt{d} \\ | \\ \mathtt{c} \end{array} \right\}$$

An analogous property holds for pullbacks. Let $\mathtt{f}_i : \mathcal{X}_i \to \mathcal{X}$ be pr-morphisms such that the underlying functions are inclusions: the pullback is given as $\mathtt{f}'_i : \bigcup_i \mathcal{X}_i|_{\mathcal{E}_1 \cap \mathcal{E}_2} \to \mathcal{X}_i$. In particular, the square below is both a pullback and a pushout.

$$\begin{array}{ccc} \bigcup_i \mathcal{X}_i|_{\mathcal{E}_1\cap\mathcal{E}_2} & \hookrightarrow & \mathcal{X}_1 \\ \downarrow & & \downarrow \\ \mathcal{X}_2 & \hookrightarrow & \mathcal{X}_1 \otimes \mathcal{X}_2 \end{array}$$

6 Operators for Visibility

We now introduce a family of operations that will be handy for our categorical characterisation. First, we provide a new operation on visibility relations.

▶ **Definition 26** (Extension). *Let* $\mathtt{G} = \langle \mathcal{E}, \prec, \lambda \rangle$ *and* $\mathcal{E}' \subseteq \mathcal{E}$. *We define the* extension *of* $\mathtt{G}$ *over* $\mathcal{E}'$ *with* ℓ *as the graph* $\mathtt{G}^{\ell}_{\mathcal{E}'} = \langle \mathcal{E}_{\top}, \prec \cup (\mathcal{E}' \times \{\top\}), \lambda[\top \mapsto \ell] \rangle$.

Intuitively, $\mathtt{G}^{\ell}_{\mathcal{E}'}$ is obtained by adding to the visibility relation $\mathtt{G}$ one additional event which sees the events in $\mathcal{E}'$. We will just write $\mathtt{G}^{\ell}$ whenever $\mathcal{E}'$ is the set of top elements of $\mathtt{G}$ – i.e., the additional event may see *all* the events of $\mathtt{G}$ – and we call it *top* extensions. Note how top extensions can be lifted to endofunctors (and actually, monads) on $\mathbf{PDag}(\mathcal{L})$. Extension allows us to characterise *saturated* specifications.

▶ **Definition 27** (Saturated specification). *Let* $\mathcal{S}$ *be a specification. It is saturated if for all graphs* $\mathtt{G}$ *the inclusion* $\mathtt{f} : \mathcal{E}_{\mathtt{G}} \to \mathcal{E}_{\mathtt{G}^{\ell}}$ *is saturated with respect to* $\mathcal{S}(\mathtt{G}^{\ell}_{\mathcal{E}})$ *(see Lemma 20), that is*

$$\forall \mathtt{G}.\ \mathcal{S}(\mathtt{G}^{\ell}_{\mathcal{E}}) = \mathtt{sat}(\mathtt{ret}(\mathcal{S}(\mathtt{G}^{\ell}_{\mathcal{E}}), \mathtt{f}), \mathtt{f})$$

We now show that all graphs are generated from suitable top extensions via pushout contructions. We consider *tree* extensions $\mathtt{T} \to \mathtt{T}^{\ell}$ for a tree $\mathtt{T}$, i.e., a graph such that each event has a unique successor. Intuitively, trees represent the simplest visibility relations, and can be seen as "generators" for $\mathbf{PDag}(\mathcal{L})$. We first show that trees are freely generated via pushouts and tree extensions.

▶ **Lemma 28.** *The sub-category of* $\mathbf{PIDag}(\mathcal{L})$ *of trees if freely generated from the empty tree via coproduct and tree extensions.*

Now we can show that trees and monic arrows between them generate the whole $\mathbf{PIDag}(\mathcal{L})$ via pushouts.

▶ **Lemma 29.** *Every monic arrow* $\mathtt{f} \colon \mathtt{G}' \to \mathtt{G}$ *of* $\mathbf{PIDag}(\mathcal{L})$ *is given by a pushout in* $\mathbf{PDag}(\mathcal{L})$ *of the form*

$$\begin{array}{ccc} \mathtt{T}' & \xhookrightarrow{\mathtt{f}'} & \mathtt{T} \\ \downarrow & & \downarrow \\ \mathtt{G}' & \xhookrightarrow{\mathtt{f}} & \mathtt{G} \end{array}$$

where $\mathtt{f}' : \mathtt{T}' \to \mathtt{T}$ *is a monic arrow between trees.*

Proof. Given a graph $\mathtt{G}$, we proceed by induction on the set $\mathcal{E}$ of the events of $\mathtt{G}$.

For the base case let us now consider a graph $\mathtt{G} = \mathtt{G}|_{\lfloor \mathtt{e} \rfloor}$ for a (necessarily unique) $\mathtt{e} \in \mathcal{E}$. Note that we can find an epic pr-morphism $\mathtt{f} : \mathtt{T} \to \mathtt{G}$, for a tree $\mathtt{T}$. This induces another epic pr-morphism $\mathtt{T}|_{\mathcal{E}_\mathtt{T}\setminus\{\mathtt{f}^{-1}(\mathtt{e})\}} \to \mathtt{G}|_{\mathcal{E}_\mathtt{G}\setminus\{\mathtt{e}\}}$. Since $(\mathtt{T}|_{\mathcal{E}_\mathtt{T}\setminus\{\mathtt{f}^{-1}(\mathtt{e})\}})^{\ell}$ is isomorphic to $\mathtt{T}$, $\mathtt{G}$ is now obtained as the obvious pushout.

The inductive step is immediate. In fact, note that $\mathtt{G} = \bigcup_{\mathtt{e}\in\mathcal{E}} \mathtt{G}|_{\lfloor \mathtt{e} \rfloor}$ for $\mathcal{E}$ the set of top elements and let $\mathtt{e}_1 \in \mathcal{E}$ and $\mathcal{E}_1 = \mathcal{E} \setminus \{\mathtt{e}_1\}$. Then, $\mathtt{G} = \mathtt{G}|_{\mathcal{E}_1} \cup \mathtt{G}|_{\lfloor \mathtt{e}_1 \rfloor}$ is the obvious pushouts of two inclusions.

◀

7 A categorical correspondence

It is now time to move towards our categorical characterisation of specifications. In this section we will show that coherent specifications induce functors preserving relevant structure (soundness) and, viceversa, that a certain class of functors induce coherent specifications (completeness). Finally, we show that these functors are "mutually inverse".

We first provide a simple technical result for coherent specifications.

▶ **Lemma 30.** *Let $\mathcal{S}$ be a coherent specification and $\mathcal{E} \subseteq \mathcal{E}_\mathtt{G}$. If $\mathcal{E} = \bigcup_{e\in\mathcal{E}} \lfloor \mathtt{e} \rfloor$ (that is, if $\mathcal{E}$ is downward closed in $\mathtt{G}$), then $\mathcal{S}(\mathtt{G})|_{\mathcal{E}} \subseteq \mathcal{S}(\mathtt{G}|_{\mathcal{E}})$.*

Proof. Since $\mathcal{E}$ is downward closed, for all $\mathtt{e} \in \mathcal{E}$ we have that $(\mathtt{G}|_{\mathcal{E}})|_{\lfloor \mathtt{e} \rfloor} = \mathtt{G}|_{\lfloor \mathtt{e} \rfloor}$. Now, by the latter and by coherence we have that $\mathcal{S}(\mathtt{G})|_{\mathcal{E}} = (\bigotimes_{\mathtt{e}\in\mathcal{E}_\mathtt{G}} \mathcal{S}(\mathtt{G}|_{\lfloor \mathtt{e} \rfloor}))\Big|_{\mathcal{E}}$ and $\mathcal{S}(\mathtt{G}|_{\mathcal{E}}) = \bigotimes_{\mathtt{e}\in\mathcal{E}} \mathcal{S}(\mathtt{G}|_{\lfloor \mathtt{e} \rfloor})$. Note that $(\bigotimes_{\mathtt{e}\in\mathcal{E}_\mathtt{G}} \mathcal{S}(\mathtt{G}|_{\lfloor \mathtt{e} \rfloor}))\Big|_{\mathcal{E}} \subseteq \bigotimes_{\mathtt{e}\in\mathcal{E}} \mathcal{S}(\mathtt{G}|_{\lfloor \mathtt{e} \rfloor})$, because a path $\mathtt{P}$ can always be restricted to a suitable path on fewer events (the viceversa in general does not hold). This concludes the proof. ◀

Our second step is to further curb the arrows in our syntax category to *monic* ones. Intuitively, we are only interested in what happens if we add further events to visibility relations. Note that a morphism in $\mathbf{PDag}(\mathcal{L})$ is a mono if and only if the underlying function is injective. We thus consider the sub-category $\mathbf{PIDag}(\mathcal{L})$ of direct acyclic graphs and monic pr-morphisms.

We now give our soundness results. We assume that specifications are *iso-coherent*, i.e, they map isomorphic graphs to isomorphic sets of paths (along the same isomorphism on events).

▶ **Proposition 31** (functors induced by specifications)**.** *An* iso-coherent *specification $\mathcal{S}$ induces a functor $\mathbb{M}(\mathcal{S}) : \mathbf{PIDag}(\mathcal{L}) \to \mathbf{SPath}(\mathcal{L})$.*

Proof. We define $\mathbb{M}(\mathcal{S})(\mathtt{G}) = \mathcal{S}(\mathtt{G})$ and $\mathbb{M}(\mathcal{S})(\mathtt{f})$ as the ps-morphism with underlying injective function $\mathtt{f} \colon (\mathcal{E}_\mathtt{G}, \lambda_\mathtt{G}) \hookrightarrow (\mathcal{E}_{\mathtt{G}'}, \lambda_{\mathtt{G}'})$. The proof boils down to show that $\mathtt{f}$ really is a ps-morphism from $\mathcal{S}(\mathtt{G})$ into $\mathcal{S}(\mathtt{G}')$, i.e., $\mathcal{S}(\mathtt{G}') \subseteq \mathtt{sat}(\mathcal{S}(\mathtt{G}), \mathtt{f})$ and, since we are considering specifications preserving isos, we can restrict our attention to the case where $\mathtt{f}$ is an inclusion.

Since $\mathtt{f}$ is a pr-morphism, $\bigcup_{\mathtt{e}\in\mathcal{E}_\mathtt{G}} \mathtt{f}(\mathtt{e})$ is downward-closed in $\mathtt{G}'$ and thus by Lemma 30 we have $\mathcal{S}(\mathtt{G}')|_{\mathcal{E}_\mathtt{G}} \subseteq \mathcal{S}(\mathtt{G}'|_{\mathcal{E}_\mathtt{G}}) = \mathcal{S}(\mathtt{G})$, the latter equality by iso-coherence. Now, consider a path $\mathtt{P} \in \mathcal{S}(\mathtt{G}')$. Since $\mathtt{P}|_{\mathcal{E}_\mathtt{G}} \in \mathcal{S}(\mathtt{G})$, we have $\mathtt{P} \in \mathtt{sat}(\mathcal{S}(\mathtt{G}), \mathtt{f})$, because saturation adds missing events – namely those in $\mathcal{E}_{\mathtt{G}'} \setminus \mathcal{E}_\mathtt{G}$ – to $\mathtt{P}|_{\mathcal{E}_\mathtt{G}}$ in all possible ways. Therefore we can conclude $\mathcal{S}(\mathtt{G}') \subseteq \mathtt{sat}(\mathcal{S}(\mathtt{G}), \mathtt{f})$.

◀

A simple corollary instantiates the result to saturated specifications. So, let $\mathbf{SSPath}(\mathcal{L})$ be the sub-category of $\mathbf{SPath}(\mathcal{L})$ of saturated monos.

▶ **Corollary 32** (functors induced by saturated specifications)**.** *An iso-coherent, saturated specification $\mathcal{S}$ induces a functor $\mathbb{S}(\mathcal{S}) : \mathbf{PIDag}(\mathcal{L}) \to \mathbf{SSPath}(\mathcal{L})$.*

As is the case for the category of sets and injective functions, $\mathbf{PIDag}(\mathcal{L})$ lacks pushouts. However, we have an easy way out via the inclusion functor into $\mathbf{PDag}(\mathcal{L})$.

▶ **Lemma 33** (Mono of $\mathbf{PDag}$)**.** *Pushouts in* $\mathbf{PDag}$ *preserve monos.*

Thus in the following we say that a functor $\mathbb{F} : \mathbf{PIDag}(\mathcal{L}) \to \mathbf{SPath}(\mathcal{L})$ weakly preserves binary pushout (and in fact, finite colimits) if any commuting square in $\mathbf{PIDag}(\mathcal{L})$ that is a pushout (via the inclusion functor) in $\mathbf{PDag}(\mathcal{L})$ is mapped by $\mathbb{F}$ to a pushout in $\mathbf{SPath}(\mathcal{L})$.

▶ **Theorem 34.** *Let* $\mathcal{S}$ *be an iso-coherent specification. The induced functor* $\mathbb{M}(\mathcal{S}) : \mathbf{PIDag}(\mathcal{L}) \to \mathbf{SPath}(\mathcal{L})$ *weakly preserves finite colimits and preserves binary pullbacks.*

Proof. The initial object is easy, since it holds by construction. As for pushouts and pullbacks: since $\mathcal{S}$ is coherent, it boils down to Lemma 24. ◀

We can now move to the completeness part.

▶ **Theorem 35.** *Let* $\mathbb{F} : \mathbf{PIDag}(\mathcal{L}) \to \mathbf{SPath}(\mathcal{L})$ *be a functor such that* $\mathbb{F}(\mathtt{G}) \subseteq \mathbb{P}(\mathcal{E}_{\mathtt{G}}, \lambda_{\mathtt{G}})$*. If* $\mathbb{F}$ *weakly preserves finite colimits and preserves binary pullbacks, it induces an iso-coherent specification* $\mathcal{S}(\mathbb{F})$*.*

Proof. Let $\mathcal{S}(\mathbb{F})(\mathtt{G}) = \mathbb{F}(\mathtt{G})$. We shall show that $\mathbb{F}(\mathtt{G})$ is coherent. Consider the following pushout in $\mathbf{PDag}(\mathcal{L})$:

$$\begin{array}{ccc} \mathtt{G}|_{\lfloor \mathtt{e}_1 \rfloor \cap \lfloor \mathtt{e}_2 \rfloor} & \hookrightarrow & \mathtt{G}|_{\lfloor \mathtt{e}_2 \rfloor} \\ \downarrow & & \downarrow \\ \mathtt{G}|_{\lfloor \mathtt{e}_1 \rfloor} & \hookrightarrow & \mathtt{G}|_{\lfloor \mathtt{e}_1 \rfloor \cup \lfloor \mathtt{e}_2 \rfloor} \end{array}$$

Since $\mathbb{F}$ preserves pullbacks, thus monos, and weakly preserves pushouts, this diagram is mapped by $\mathbb{F}$ to the following pushout in $\mathbf{SPath}(\mathcal{L})$:

$$\begin{array}{ccc} \mathbb{F}(\mathtt{G}|_{\lfloor \mathtt{e}_1 \rfloor \cap \lfloor \mathtt{e}_2 \rfloor}) & \hookrightarrow & \mathbb{F}(\mathtt{G}|_{\lfloor \mathtt{e}_2 \rfloor}) \\ \downarrow & & \downarrow \\ \mathbb{F}(\mathtt{G}|_{\lfloor \mathtt{e}_1 \rfloor}) & \hookrightarrow & \mathbb{F}(\mathtt{G}|_{\lfloor \mathtt{e}_1 \rfloor \cup \lfloor \mathtt{e}_2 \rfloor}) \end{array}$$

where all underlying functions between events are inclusions. By Lemma 24 we have:

$$\mathbb{F}(\mathtt{G}|_{\lfloor \mathtt{e}_1 \rfloor \cup \lfloor \mathtt{e}_2 \rfloor}) \simeq \mathbb{F}(\mathtt{G}|_{\lfloor \mathtt{e}_1 \rfloor}) \otimes \mathbb{F}(\mathtt{G}|_{\lfloor \mathtt{e}_2 \rfloor})$$

Since clearly $\mathtt{G} = \mathtt{G}|_{\bigcup_{\mathtt{e} \in \mathcal{E}_{\mathtt{G}}} \lfloor \mathtt{e} \rfloor}$, by associativity of pushouts we obtain coherence:

$$\mathbb{F}(\mathtt{G}) \simeq \bigotimes_{\mathtt{e} \in \mathcal{E}_{\mathtt{G}}} \mathbb{F}(\mathtt{G}|_{\lfloor \mathtt{e} \rfloor})$$

Iso-coherence follows from $\mathbb{F}$ being a functor, hence preserving isos. ◀

Furthermore, the two constructions are inverse to each other.

▶ **Proposition 36.** *We have* $\mathbb{M}(\mathcal{S}(\mathbb{F})) \simeq \mathbb{F}$*.*

Proof. For notational convenience, we denote $\mathbb{M}(\mathcal{S}(\mathbb{F}))$ by $\mathbb{M}'$. We will show the existence of a natural isomorphism $\varphi\colon \mathbb{M}' \Rightarrow \mathbb{F}$. By definition, we have $\mathbb{M}'(\mathtt{G}) = \mathcal{S}(\mathbb{F})(\mathtt{G}) = \mathbb{F}(\mathtt{G})$, therefore we can define $\varphi_{\mathtt{G}} = \textsc{id}_{\mathbb{F}(\mathtt{G})}$. We need to prove that it is natural, which in this case amounts to show $\mathbb{M}'(\mathtt{f}) = \mathbb{F}(\mathtt{f})$, for $\mathtt{f}\colon \mathtt{G} \to \mathtt{G}'$ in $\mathbf{PIDag}(\mathcal{L})$. This follows from $\mathbb{M}'(\mathtt{f})$ and $\mathbb{F}(\mathtt{f})$ having the same underlying function between events, namely the inclusion $(\mathcal{E}_{\mathbb{F}(\mathtt{G})}, \lambda_{\mathbb{F}(\mathtt{G})}) \to (\mathcal{E}_{\mathbb{F}(\mathtt{G}')}, \lambda_{\mathbb{F}(\mathtt{G}')})$. ◀

We can sharpen the result above by removing the on-the-nose requirement $\mathbb{F}(\mathtt{G}) \subseteq \mathbb{P}(\mathcal{E}_{\mathtt{G}}, \lambda_{\mathtt{G}})$. To this end, we need to further constraint the class of functors. First, we consider the effect of top extension on sets of paths.

▶ **Definition 37.** *Let $\mathcal{X} \subseteq \mathbb{P}(\mathcal{E}, \lambda)$ be a set of paths and $\ell \in \mathcal{L}$ a label. Its* top extension *is defined as $\{\mathtt{P}^{\ell} \mid \mathtt{P} \in \mathcal{X}\}$. Its* saturated top extension *is defined as $\mathtt{sat}(\mathtt{P}, \mathtt{f})$ for $\mathtt{f} : (\mathcal{E}, \lambda) \to (\mathcal{E}_{\top}, \lambda[\top \mapsto \ell])$ the obvious inclusion.*

We say that $\mathbb{F}$ *preserves top extensions* if it maps top extensions (of dags) to top extensions (of paths). We can now state two additional instances of Theorem 35. We call *topological* those specifications such that $\mathcal{S}(\mathtt{G}) \subseteq \{\mathtt{P} \mid \prec_{\mathtt{G}} \subseteq \leq_{\mathtt{P}}\}$. In other words, a topological specification maps a dag $\mathtt{G}$ to paths that are topological sorts of $\mathtt{G}$.

▶ **Proposition 38.** *Let $\mathbb{F} : \mathbf{PIDag}(\mathcal{L}) \to \mathbf{SPath}(\mathcal{L})$ that preserves top extensions. If $\mathbb{F}$ weakly preserves finite colimits and preserves binary pullbacks, it induces an iso-coherent topological specification $\mathcal{S}(\mathbb{F})$.*

Finally, we consider the sub-category of $\mathbf{SSPath}(\mathcal{L})$ of saturated ps-morphisms,

▶ **Proposition 39.** *Let $\mathbb{F} : \mathbf{PIDag}(\mathcal{L}) \to \mathbf{SSPath}(\mathcal{L})$ be a functor that preserves top extensions. If $\mathbb{F}$ weakly preserves finite colimits and preserves binary pullbacks, it induces an iso-coherent saturated specification $\mathcal{S}(\mathbb{F})$.*

8 Conclusions

In this paper we have provided a functorial characterisation of RDTs specifications. Our starting point is the denotational approach proposed in [6], in which RDTs specifications are associated with those functions mapping visibility graphs into sets of admissible arbitrations that are also saturated and coherent. In this work, we consider the category $\mathbf{PDag}(\mathcal{L})$ that has labelled, acyclic graphs as objects and pr-morphisms as arrows for representing visibility graphs. We equip $\mathbf{PDag}(\mathcal{L})$ with operators that model the evolution of visibility graphs and we show that monic arrows in $\mathbf{PDag}(\mathcal{L})$ can be obtained as pushouts. We call $\mathbf{PIDag}(\mathcal{L})$ the full-subcategory of acyclic graphs and monic pr-morphisms. For arbitrations, we take $\mathbf{SPath}(\mathcal{L})$, which is the category of sets of labelled, total orders and ps-morphisms. Then, we show that each coherent specification mapping isomorphic graphs into isomorphic set of paths (i.e., iso-coherent) induces a functor $\mathbb{M}(\mathcal{S}) : \mathbf{PIDag}(\mathcal{L}) \to \mathbf{SPath}(\mathcal{L})$. Conversely, we prove that a functor $\mathbb{F} : \mathbf{PIDag}(\mathcal{L}) \to \mathbf{SPath}(\mathcal{L})$ that preserves finite colimits and binary pullbacks induces an iso-coherent specification $\mathcal{S}(\mathbb{F})$. Moreover, $\mathbb{M}(\mathcal{S})$ and $\mathcal{S}(\mathbb{F})$ are shown to be inverse of each other.

We believe that our characterisation of RDTs provides an ideal setting for the development of techniques for handling RDT composition. Our long term goal is to equip RDT specifications with a set of operators that enable us to specify and reason about complex RDTs compositionally, i.e., in terms of constituent parts. We aim to provide a uniform formal treatment of compositional approaches such as those proposed in [1, 10, 12].

References

1 Baquero, C., Almeida, P.S., Cunha, A., Ferreira, C., et al.: Composition in state-based replicated data types. Bulletin of EATCS **3**(123) (2017)

2 Bouajjani, A., Enea, C., Hamza, J.: Verifying eventual consistency of optimistic replication systems. In: ACM SIGPLAN Notices. vol. 49(1), pp. 285–296. ACM (2014)

3 Burckhardt, S., Gotsman, A., Yang, H.: Understanding eventual consistency. Tech. Rep. MSR-TR-2013-39, Microsoft Research (2013)

4 Burckhardt, S., Gotsman, A., Yang, H., Zawirski, M.: Replicated data types: specification, verification, optimality. In: Jagannathan, S., Sewell, P. (eds.) POPL 2014. pp. 271–284. ACM (2014)

5 Cerone, A., Bernardi, G., Gotsman, A.: A framework for transactional consistency models with atomic visibility. In: Aceto, L., de Frutos-Escrig, D. (eds.) CONCUR 2015. LIPIcs, vol. 42. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik (2015)

6 Gadducci, F., Melgratti, H., Roldán, C.: A denotational view of replicated data types. In: International Conference on Coordination Languages and Models. pp. 138–156. Springer (2017)

7 Gadducci, F., Melgratti, H., Roldán, C.: On the semantics and implementation of replicated data types. Science of Computer Programming **167**, 91–113 (2018)

8 Gilbert, S., Lynch, N.: Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services. SIGACT News **33**(2), 51–59 (Jun 2002)

9 Gotsman, A., Burckhardt, S.: Consistency models with global operation sequencing and their composition. In: LIPIcs-Leibniz International Proceedings in Informatics. vol. 91. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik (2017)

10 Gotsman, A., Yang, H.: Composite replicated data types. In: Vitek, J. (ed.) ESOP 2015. LNCS, vol. 9032, pp. 585–609. Springer (2015)

11 Kaki, G., Earanky, K., Sivaramakrishnan, K., Jagannathan, S.: Safe replication through bounded concurrency verification. Proceedings of the ACM on Programming Languages **2**(OOPSLA), 164 (2018)

12 Leijnse, A., Almeida, P.S., Baquero, C.: Higher-order patterns in replicated data types. In: Proceedings of the 6th Workshop on Principles and Practice of Consistency for Distributed Data. p. 5. ACM (2019)

13 Shapiro, M., Preguiça, N., Baquero, C., Zawirski, M.: Conflict-free replicated data types. In: Défago, X., Petit, F., Villain, V. (eds.) SSS 2011. LNCS, vol. 6976, pp. 386–400. Springer (2011)

14 Sivaramakrishnan, K.C., Kaki, G., Jagannathan, S.: Declarative programming over eventually consistent data stores. In: Grove, D., Blackburn, S. (eds.) PLDI 2015. pp. 413–424. ACM (2015)

A Proof of Lem. 2.

Proof.

For $\Leftarrow$), assume that $\mathtt{f}$ is a pr-morphism, then

1. By definition of pr-morphism

$$\mathtt{f}(\mathtt{s})\ \gamma\ \mathtt{f}(\mathtt{s}') \text{ implies } \exists \bar{\mathtt{s}} \in \mathtt{E}.\bar{\mathtt{s}}\ \rho\ \mathtt{s}'\ \wedge\ \mathtt{f}(\mathtt{s}) = \mathtt{f}(\bar{\mathtt{s}})$$

   Since $\mathtt{f}$ is injective, $\bar{\mathtt{e}} = \mathtt{e}$ holds, and hence $\mathtt{e}\ \rho\ \mathtt{e}'$.

2. Let $\mathcal{T} = \bigcup_{\mathtt{e}\in\mathtt{E}} \mathtt{f}(\mathtt{e})$. We want to show that

$$\forall \mathtt{t} \in \mathtt{T}.\forall \mathtt{t}' \in \mathcal{T}.\mathtt{t}\gamma\ \mathtt{t}'\ \textit{implies}\ \mathtt{t} \in \mathcal{T}$$

   The proof follows by contradiction. Assume that $\exists \mathtt{t} \in \mathtt{T}.\exists \mathtt{t}' \in \mathcal{T}.\mathtt{t}\gamma\ \mathtt{t}'\ \wedge\ \mathtt{t} \notin \mathcal{T}$. By definition of $\mathcal{T}, \exists \mathtt{e} \in \mathtt{E}$ such that $\mathtt{f}(\mathtt{e}) = \mathtt{t}'$. Since $\mathtt{f}$ is pr-morphism, then

$$\mathtt{t}\ \gamma\ \mathtt{f}(\mathtt{e}) \text{ implies } \exists \mathtt{e}' \in \mathtt{E}.\mathtt{e}'\ \rho\ \mathtt{e}\ \wedge\ \mathtt{t} = \mathtt{f}(\mathtt{e}')$$

   Therefore $\mathtt{t} = \mathtt{f}(\mathtt{e}') \in \mathcal{T}$, which contradicts the assumption $\mathtt{t} \notin \mathcal{T}$.

For $\Rightarrow$), assume that 1) and 2) hold. Take $\mathtt{e} \in \mathtt{E}$ and $\mathtt{t} \in \mathtt{T}$. If $\mathtt{t}\ \gamma\ \mathtt{f}(\mathtt{e})$, then there exists $\mathtt{e}' \in \mathtt{E}$ such that $\mathtt{t} = \mathtt{f}(\mathtt{e}')$ because of (2). By (1) holds, $\mathtt{f}(\mathtt{e}')\ \gamma\ \mathtt{f}(\mathtt{e})$ implies $\mathtt{e}'\ \rho\ \mathtt{e}$. ◀