A Categorical Account of Replicated Data Types

Fabio Gadducci
Dipartimento di Informatica, Università di Pisa, Italia
https://www.di.unipi.it/~gadducci
fabio.gadducci@unipi.it

Hernán Melgratti
Departamento de Computación, Universidad de Buenos Aires, Argentina
ICC-CONICET-UBA, Buenos Aires, Argentina
https://lafhis.dc.uba.ar/~melgratti
hmelgra@dc.uba.ar

Christian Roldán
Departamento de Computación, Universidad de Buenos Aires, Argentina
https://lafhis.dc.uba.ar/~croldan
croldan@dc.uba.ar

Matteo Sammartino
Department of Computer Science, University College London, UK
https://matteosammartino.com
m.sammartino@ucl.ac.uk

Abstract

Replicated Data Types (RDTs) have been introduced as a suitable abstraction 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 an algebraic 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 Theory of computation → Program semantics; Software and its engineering → General programming languages

Keywords and phrases Replicated data type, Specification, Functorial characterisation

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2019.42

Funding Research partially supported by the MIUR PRIN 2017FTXR7S "IT-MaTTerS", by the EU H2020 RISE programme under the Marie Skłodowska-Curie grant agreement 778233, by the UBACyT projects 20020170100544BA and 20020170100086BA, by the PIP project 11220130100148CO, by the Leverhulme Prize PLP-2016-129, and by the EPSRC grant EP/S028641/1.

Acknowledgements We thank the reviewers for their careful reading and insightful comments.

1 Introduction

The CAP theorem establishes that a distributed data store can simultaneously provide two of the following three properties: consistency, availability, and tolerance to 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 (RDTs) 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

$$\mathcal{S}_{lwwR}\left(\begin{tikzcd}[column sep=small] \langle \mathtt{wr}(1),\mathtt{ok}\rangle \arrow[dr] & & \langle \mathtt{wr}(2),\mathtt{ok}\rangle \arrow[dl] \\ & \langle \mathtt{rd},2\rangle & \end{tikzcd}\right) = \left\{ \begin{tikzcd}[row sep=small] \langle \mathtt{wr}(1),\mathtt{ok}\rangle \arrow[d, dash] & \langle \mathtt{wr}(1),\mathtt{ok}\rangle \arrow[d, dash] & \langle \mathtt{rd},2\rangle\} \arrow[d, dash] \\ \langle \mathtt{wr}(2),\mathtt{ok}\rangle \arrow[d, dash] , & \langle \mathtt{rd},2\rangle\} \arrow[d, dash] , & \langle \mathtt{wr}(1),\mathtt{ok}\rangle \arrow[d, dash] \\ \langle \mathtt{rd},2\rangle\} & \langle \mathtt{wr}(2),\mathtt{ok}\rangle & \langle \mathtt{wr}(2),\mathtt{ok}\rangle \end{tikzcd} \right\}$$

(a) Visibility relation with admissible arbitrations.

$$\mathcal{S}_{lwwR}\left(\begin{tikzcd} \langle \mathtt{wr}(1),\mathtt{ok}\rangle \arrow[d] \\ \langle \mathtt{rd},0\rangle \end{tikzcd}\right) = \emptyset$$

(b) Non admissible arbitrations.

Figure 1 A register specification.

operations are executed. A view is usually represented by a *visibility* relation, which is a binary, acyclic relation over the operations (a.k.a. *events*) executed by the system. The state of a store is described instead as a total order over the events, called *arbitration*, which describes the way in which conflicting concurrent operations are resolved. Different specification approaches for RDTs are presented in the literature, all of them building on the notions of visibility and arbitration [2, 3, 4, 5, 7, 6, 9, 11, 13, 14]. A purely functional approach for the specification of RDTs has been presented in [7, 6], where an RDT is associated with a function that maps each visibility relation into a set of arbitrations.

Consider an RDT `Register` that represents a memory cell, whose content can be updated and read. Following the approach in [7], the RDT `Register` is specified by a function that maps visibility relations into sets of arbitrations: we call here such function $\mathcal{S}_{lwwR}$. Figure 1a illustrates the definition of $\mathcal{S}_{lwwR}$ for the case in which the visibility relation involves two concurrent writes and a read. Events are depicted by pairs $\langle \mathtt{operation}, \mathtt{result}\rangle$ where $\mathtt{wr}(\mathtt{k})$ stands for an operation that writes the value $\mathtt{k}$ and $\mathtt{rd}$ stands for a read. The two writes are unrelated (i.e., they are not visible to each other), while the read operation sees both writes. The returned value of the read operation is 2, which coincides with one of the visible written values. According to Figure 1a, $\mathcal{S}_{lwwR}$ maps such visibility graph into a set containing those arbitrations (i.e., total orders over the three events in the visibility relation) in which $\mathtt{wr}(1)$ precedes $\mathtt{wr}(2)$. Arbitrations may not reflect the causal ordering of events; in fact, the last two arbitrations in the right-hand-side of the equation in Figure 1a place the read before the operation that writes the read value 2. We remark that arbitrations do not necessarily account for real-time orderings of events: they are instead possible ways in which events can be *logically* ordered to explain a given visibility. For instance, the excluded arbitrations in the image of $\mathcal{S}_{lwwR}$ are the total orders in which $\mathtt{wr}(2)$ precedes $\mathtt{wr}(1)$, i.e., the specification bans the behaviour in which a read operation returns a value that is different from the last written one. An extreme situation is the case in which the specification maps a visibility relation into an empty set of arbitrations, which means that events cannot be logically ordered to explain such visibility. For instance, the equation in Figure 1b assigns an empty set of arbitrations to a visibility relation in which the read operation returns a value that is different from the unique visible written value (i.e., it returns 0 instead of 1). In this way, the specification bans the behaviour in which a read operation returns a value that does not match a previous written value. As originally shown [7], this style of specification can be considered (and it is actually more general than) the model for the operational description of RDTs proposed in [4]. We refer the reader to [6] for a formal comparison of the two different approaches.

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 injective *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 sets 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 paths $\mathcal{X}_2$ 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 *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: as illustrated in [6], they correspond to what are called *return value consistent* RDTs in [4]. We establish a bijection between functors and specifications, showing that a 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 [6]. Section 5 introduces our semantical model, the category of set of paths, describing some of its basic properties with respect to limits and colimits. In Section 6 we present 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, a comparison of the proposed constructions with those presented in [7], and some hints towards future work.

2 Preliminaries on Relations

Relations. Given a finite set $\mathtt{E}$, a (binary) *relation* ρ over $\mathtt{E}$ is a subset $\rho \subseteq \mathtt{E} \times \mathtt{E}$ of the cartesian product of $\mathtt{E}$ with itself. We use the pair $\langle \mathtt{E}, \rho \rangle$ to denote a relation ρ over $\mathtt{E}$, in order to always have the set of events explicit, and simply $\emptyset$ to denote the empty relation.

A subset $\mathtt{E}' \subseteq \mathtt{E}$ is *downward closed* with respect to ρ if $\forall \mathtt{e} \in \mathtt{E}, \mathtt{e}' \in \mathtt{E}'. \mathtt{e}\ \rho\ \mathtt{e}'$ implies $\mathtt{e} \in \mathtt{E}'$ and, when ρ is clear, we write $\lfloor \mathtt{e} \rfloor$ for the smallest downward closed set including $\mathtt{e} \in \mathtt{E}$.

▶ **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*

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

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

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

Note that both classes of morphisms are closed under composition: we denote as **Bin** the category of relations and their morphisms and **PBin** 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*

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

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

Proof. For $\Rightarrow$), let us 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), $\mathtt{f}(\mathtt{e}')\ \gamma\ \mathtt{f}(\mathtt{e})$ implies $\mathtt{e}'\ \rho\ \mathtt{e}$.

For $\Leftarrow$), by the definition of pr-morphism $\mathtt{f}(\mathtt{e})\ \gamma\ \mathtt{f}(\mathtt{e}')$ implies $\exists \bar{\mathtt{e}} \in \mathtt{E}.\ \bar{\mathtt{e}}\ \rho\ \mathtt{e}'\ \wedge\ \mathtt{f}(\mathtt{e}) = \mathtt{f}(\bar{\mathtt{e}})$. Since $\mathtt{f}$ is injective, $\bar{\mathtt{e}} = \mathtt{e}$ and hence $\mathtt{e}\ \rho\ \mathtt{e}'$. So, let $\mathcal{T} = \bigcup_{\mathtt{e} \in \mathtt{E}} \mathtt{f}(\mathtt{e})$. We want to show that

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

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

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

▶ **Proposition 3** (Properties of **PBin**). *The inclusion functor* **PBin** $\rightarrow$ **Bin** *reflects finite colimits and binary pullbacks.*

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

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

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

We now introduce labelled relations. Consider the forgetful functors $\mathtt{U_r} : \mathbf{Bin} \rightarrow \mathbf{Set}$ and $\mathtt{U_p} : \mathbf{PBin} \rightarrow \mathbf{Set}$, the latter factoring through the inclusion functor **PBin** $\rightarrow$ **Bin**. Given a set $\mathcal{L}$ of labels, we consider the comma categories $\mathbf{Bin}(\mathcal{L}) = \mathtt{U_r} \downarrow \mathcal{L}$ and $\mathbf{PBin}(\mathcal{L}) = \mathtt{U_p} \downarrow \mathcal{L}$: finite colimits and binary pullbacks always exist and are essentially computed as in **Bin**.

Explicitly, an object in $\mathtt{U_r} \downarrow \mathcal{L}$ is a triple $(\mathtt{E}, \rho, \lambda)$ for a labeling function $\lambda : \mathtt{E} \rightarrow \mathcal{L}$. A label-preserving morphism $(\mathtt{E}, \rho, \lambda) \rightarrow (\mathtt{E}', \rho', \lambda')$ is a morphism $\mathtt{f} : (\mathtt{E}, \rho) \rightarrow (\mathtt{E}', \rho')$ such that $\forall \mathtt{s} \in \mathtt{E}.\ \lambda(\mathtt{s}) = \lambda'(\mathtt{f}(\mathtt{s}))$. Moreover, finite colimits and binary pullbacks exist and are computed as in **Bin**. Similar properties hold 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** (**PDag**). **PDag** *is the full sub-category of* **PBin** *whose objects are directed acyclic graphs.*

In other terms, objects are relations whose transitive closures are *strict* partial orders.

▶ Remark 6. The full sub-category of **Bin** whose objects are directed acyclic graphs is not suited for our purposes, since 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** $\rightarrow$ **PBin** *reflects finite colimits and binary pullbacks.*

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

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

Note that defining **Path** as only containing pr-morphisms would be too restrictive, 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** $\rightarrow$ **Bin** *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 $\mathtt{U_{rp}} : \mathbf{Path} \rightarrow \mathbf{Set}$ and $\mathtt{U_{pd}} : \mathbf{PDag} \rightarrow \mathbf{Set}$: 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, finite colimits and binary pullbacks always exist and are essentially computed as in **Bin**.

4 Replicated Data Type Specification

We briefly recall the set-theoretical model of replicated data types (RDT) introduced in [6]. Our main result is its categorical characterisation, which is given in the following sections.

First, some notation. We denote a graph as the triple $\langle \mathcal{E}, \prec, \lambda \rangle$ and a path as the triple $\langle \mathcal{E}, \leq, \lambda \rangle$, 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 $\mathcal{X} = \{\langle \mathcal{E}_i, \leq_i, \lambda_i \rangle\}_i$. First, we say that the paths of a set $\mathcal{X}$ are *compatible* if $\forall \mathtt{e}, i, j.\ \mathtt{e} \in \mathcal{E}_i \cap \mathcal{E}_j$ implies $\lambda_i(\mathtt{e}) = \lambda_j(\mathtt{e})$.

▶ **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 (finite) graphs and (finite) 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 [6], 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 $\langle \mathit{operation}, \mathit{value} \rangle$ that is *saturated* and *past-coherent*. The former property is a technical one: roughly, if $\mathtt{G}'$ is an extension of $\mathtt{G}$ with a 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 included in 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*

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

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 coherent specification for the `Register` RDT.

▶ **Example 13** (Register). Fix the set of labels $\mathcal{L} = \{\langle \mathtt{wr}(\mathtt{k}), \mathtt{ok}\rangle, \langle \mathtt{rd}, \mathtt{k}\rangle \mid \mathtt{k} \in \mathbb{N}\} \cup \{\langle \mathtt{rd}, \bot\rangle\}$. Then, the specification of the RDT `Register` is given by the function $\mathcal{S}_{lwwR}$ defined as

$$\mathtt{P} \in \mathcal{S}_{lwwR}(\mathtt{G}) \text{ iff } \forall \mathtt{e} \in \mathcal{E}_{\mathtt{G}}. \begin{cases} \lambda(\mathtt{e}) = \langle \mathtt{rd}, \bot\rangle \text{ implies } \forall \mathtt{e}' \prec_{\mathtt{G}} \mathtt{e}, \mathtt{k}.\ \lambda(\mathtt{e}') \neq \langle \mathtt{wr}(\mathtt{k}), \mathtt{ok}\rangle \\ \forall \mathtt{k}.\ \lambda(\mathtt{e}) = \langle \mathtt{rd}, \mathtt{k}\rangle \text{ implies } \exists \mathtt{e}' \prec_{\mathtt{G}} \mathtt{e}.\ \lambda(\mathtt{e}') = \langle \mathtt{wr}(\mathtt{k}), \mathtt{ok}\rangle \text{ and} \\ \qquad \forall \mathtt{e}'' \prec_{\mathtt{G}} \mathtt{e}, \mathtt{k}' \neq \mathtt{k}.\ \mathtt{e}' <_{\mathtt{P}} \mathtt{e}'' \text{ implies } \lambda(\mathtt{e}'') \neq \langle \mathtt{wr}(\mathtt{k}'), \mathtt{ok}\rangle \end{cases}$$

Intuitively, a visibility graph $\mathtt{G}$ is mapped to a non-empty set of arbitrations (i.e., $\mathcal{S}_{lwwR}(\mathtt{G}) \neq \emptyset$) only when each event $\mathtt{e}$ in $\mathtt{G}$ associated with a read operation has a return value $\mathtt{k}$ that matches the value written by the greatest event $\mathtt{e}'$ (according to $<_{\mathtt{P}}$). The result of a read is undefined (i.e., $\bot$) when it does not see any write (first condition).

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 first look at a free construction for paths, and then we turn our attention to morphisms.

5.1 Saturation

▶ **Definition 14** (Path 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*

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

Saturation is generalised 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 $\mathtt{f}$ from $\{\langle \mathtt{wr}(1), \mathtt{ok}\rangle, \langle \mathtt{wr}(2), \mathtt{ok}\rangle\}$ to $\{\langle \mathtt{wr}(1), \mathtt{ok}\rangle, \langle \mathtt{wr}(2), \mathtt{ok}\rangle, \langle \mathtt{rd}, 2\rangle\}$. Then, we have

$$\mathtt{sat}\left(\left\{ \begin{array}{c} \langle \mathtt{wr}(1), \mathtt{ok}\rangle \\ | \\ \langle \mathtt{wr}(2), \mathtt{ok}\rangle \end{array} \right\}, \mathtt{f}\right) = \left\{ \begin{array}{c} \langle \mathtt{wr}(1), \mathtt{ok}\rangle \\ | \\ \langle \mathtt{wr}(2), \mathtt{ok}\rangle \\ | \\ \langle \mathtt{rd}, 2\rangle\} \end{array}, \begin{array}{c} \langle \mathtt{wr}(1), \mathtt{ok}\rangle \\ | \\ \langle \mathtt{rd}, 2\rangle\} \\ | \\ \langle \mathtt{wr}(2), \mathtt{ok}\rangle \end{array}, \begin{array}{c} \langle \mathtt{rd}, 2\rangle\} \\ | \\ \langle \mathtt{wr}(1), \mathtt{ok}\rangle \\ | \\ \langle \mathtt{wr}(2), \mathtt{ok}\rangle \end{array} \right\}$$

Intuitively, saturation adds $\langle \mathtt{rd}, 2\rangle$ – 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.

▶ **Definition 16** (Path 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*

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

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 17.** *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 18.** *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 19.** Consider the set of paths $\mathcal{X}_1$ and $\mathcal{X}_2$ and the pr-morphism $\mathtt{f}$ below

$$\mathcal{X}_1 = \left\{ \begin{array}{c} \langle \mathtt{wr}(1), \mathtt{ok}\rangle \\ | \\ \langle \mathtt{wr}(2), \mathtt{ok}\rangle \end{array} \right\} \qquad \mathcal{X}_2 = \left\{ \begin{array}{c} \langle \mathtt{wr}(1), \mathtt{ok}\rangle \\ | \\ \langle \mathtt{wr}(2), \mathtt{ok}\rangle \\ | \\ \langle \mathtt{rd}, 2\rangle \end{array} \right\} \qquad \mathtt{f} : \begin{array}{c} \langle \mathtt{wr}(1), \mathtt{ok}\rangle \\ | \\ \langle \mathtt{wr}(2), \mathtt{ok}\rangle \end{array} \to \begin{array}{c} \langle \mathtt{wr}(1), \mathtt{ok}\rangle \\ | \\ \langle \mathtt{wr}(2), \mathtt{ok}\rangle \\ | \\ \langle \mathtt{rd}, 2\rangle \end{array}$$

the underlying function $\mathtt{f}$ (defined in Example 15) is *not* saturated with respect to $\mathcal{X}_2$ because

$$\left\{ \begin{array}{c} \langle \mathtt{wr}(1), \mathtt{ok}\rangle \\ | \\ \langle \mathtt{wr}(2), \mathtt{ok}\rangle \\ | \\ \langle \mathtt{rd}, 2\rangle \end{array} \right\} \neq \mathtt{sat}(\mathtt{ret}(\left\{ \begin{array}{c} \langle \mathtt{wr}(1), \mathtt{ok}\rangle \\ | \\ \langle \mathtt{wr}(2), \mathtt{ok}\rangle \\ | \\ \langle \mathtt{rd}, 2\rangle \end{array} \right\}, \mathtt{f}), \mathtt{f}) = \mathtt{sat}(\left\{ \begin{array}{c} \langle \mathtt{wr}(1), \mathtt{ok}\rangle \\ | \\ \langle \mathtt{wr}(2), \mathtt{ok}\rangle \end{array} \right\}, \mathtt{f})$$

In fact, the ps-morphism $\mathtt{f} : \mathcal{X}_1 \to \mathcal{X}_2$ only adds the new event $\langle \mathtt{rd}, 2\rangle$ on top of the path in $\mathcal{X}_1$, thus making it a *topological* ps-morphism (see Section 7.3 later on).

5.2 From saturation to categories

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

▶ **Definition 20** (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 paths $\mathcal{X}_2$ if any path in $\mathcal{X}_2$ can be obtained by adding events to some path in $\mathcal{X}_1$. This notion captures the idea that arbitrations of larger visibilities are obtained as extensions of smaller visibilities.

▶ **Example 21.** Consider the following three sets and the function $\mathtt{f}$ from Example 15

$$\mathcal{X}_1 = \left\{ \begin{array}{c} \langle \mathtt{wr}(1), \mathtt{ok}\rangle \\ | \\ \langle \mathtt{wr}(2), \mathtt{ok}\rangle \end{array} \right\} \quad \mathcal{X}_2 = \left\{ \begin{array}{c} \langle \mathtt{wr}(1), \mathtt{ok}\rangle \\ | \\ \langle \mathtt{wr}(2), \mathtt{ok}\rangle \\ | \\ \langle \mathtt{rd}, 2\rangle \end{array}, \begin{array}{c} \langle \mathtt{wr}(1), \mathtt{ok}\rangle \\ | \\ \langle \mathtt{rd}, 2\rangle \\ | \\ \langle \mathtt{wr}(2), \mathtt{ok}\rangle \end{array} \right\} \quad \mathcal{X}_3 = \left\{ \begin{array}{c} \langle \mathtt{wr}(1), \mathtt{ok}\rangle \\ | \\ \langle \mathtt{wr}(2), \mathtt{ok}\rangle \\ | \\ \langle \mathtt{rd}, 2\rangle \end{array}, \begin{array}{c} \langle \mathtt{wr}(2), \mathtt{ok}\rangle \\ | \\ \langle \mathtt{rd}, 2\rangle \\ | \\ \langle \mathtt{wr}(1), \mathtt{ok}\rangle \end{array} \right\}$$

Now, $\mathtt{f}$ induces a ps-morphism $\mathtt{f} : \mathcal{X}_1 \to \mathcal{X}_2$ because $\mathcal{X}_2 \subseteq \mathtt{sat}(\mathcal{X}_1, \mathtt{f})$ (the latter is shown in Example 15). On the contrary, there is no ps-morphism from $\mathcal{X}_1$ to $\mathcal{X}_3$: the rightmost path of $\mathcal{X}_3$ cannot be obtained by extending a path of $\mathcal{X}_1$ with an event labelled by $\langle \mathtt{rd}, 2\rangle$.

▶ **Definition 22** (Sets of Paths Category). *We define* $\mathbf{SPath}(\mathcal{L})$ *as the category whose objects are sets of paths labelled over $\mathcal{L}$ and arrows are ps-morphisms.*

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

Proof.

(Strict) initial object. The (unique) initial object 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: it 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 $\mathtt{f}'_i:\mathcal{E}_1\times_{\mathcal{E}}\mathcal{E}_2\to\mathcal{E}_i$ in the category of sets: it 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 above characterisation of pushouts is enabled by the fact that we considered injective functions. To help intuition, we now instantiate that characterisation 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 their 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 $\mathtt{sat}(\mathcal{X}_i,\mathtt{f}'_\mathtt{i})=\bigcup_{\mathtt{Q}\in\mathcal{X}_i}\{\mathtt{P}\mid\mathtt{P}\in\mathbb{P}(\bigcup_i\mathcal{E}_i,\bigcup_i\lambda_i)$ and $\mathtt{f}'_\mathtt{i}$ 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

$$\left\{\begin{array}{c}\langle\mathtt{wr}(1),\mathtt{ok}\rangle\\ |\\ \langle\mathtt{wr}(2),\mathtt{ok}\rangle\\ |\\ \langle\mathtt{rd},2\rangle\end{array}\right\}\leftarrow\left\{\begin{array}{c}\langle\mathtt{wr}(1),\mathtt{ok}\rangle\\ |\\ \langle\mathtt{wr}(2),\mathtt{ok}\rangle\end{array}\right\}\to\left\{\begin{array}{ccc}\langle\mathtt{wr}(1),\mathtt{ok}\rangle & & \langle\mathtt{wr}(2),\mathtt{ok}\rangle\\ | & & |\\ \langle\mathtt{wr}(2),\mathtt{ok}\rangle & , & \langle\mathtt{wr}(1),\mathtt{ok}\rangle\\ | & & |\\ \langle\mathtt{rd},1\rangle & & \langle\mathtt{rd},1\rangle\end{array}\right\}$$

then, the pushout is given by the following ps-morphisms

$$\left\{\begin{array}{c}\langle\mathtt{wr}(1),\mathtt{ok}\rangle\\ |\\ \langle\mathtt{wr}(2),\mathtt{ok}\rangle\\ |\\ \langle\mathtt{rd},2\rangle\end{array}\right\}\to\left\{\begin{array}{ccc}\langle\mathtt{wr}(1),\mathtt{ok}\rangle & & \langle\mathtt{wr}(1),\mathtt{ok}\rangle\\ | & & |\\ \langle\mathtt{wr}(2),\mathtt{ok}\rangle & & \langle\mathtt{wr}(2),\mathtt{ok}\rangle\\ | & , & |\\ \langle\mathtt{rd},1\rangle & & \langle\mathtt{rd},2\rangle\\ | & & |\\ \langle\mathtt{rd},2\rangle & & \langle\mathtt{rd},1\rangle\end{array}\right\}\leftarrow\left\{\begin{array}{ccc}\langle\mathtt{wr}(1),\mathtt{ok}\rangle & & \langle\mathtt{wr}(2),\mathtt{ok}\rangle\\ | & & |\\ \langle\mathtt{wr}(2),\mathtt{ok}\rangle & , & \langle\mathtt{wr}(1),\mathtt{ok}\rangle\\ | & & |\\ \langle\mathtt{rd},1\rangle & & \langle\mathtt{rd},1\rangle\end{array}\right\}$$

An analogous property holds for pullbacks. Let $\mathtt{f}_i:\mathcal{X}_i\to\mathcal{X}$ be ps-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.

$$\begin{tikzcd}
\bigcup_i\mathcal{X}_i|_{\mathcal{E}_1\cap\mathcal{E}_2} \arrow[r, hook] \arrow[d, hook] & \mathcal{X}_1 \arrow[d, hook] \\
\mathcal{X}_2 \arrow[r, hook] & \mathcal{X}_1\otimes\mathcal{X}_2
\end{tikzcd}$$

6 Structure and Operators for Visibility

We now study the category of visibility relations. We first introduce an operation that will be handy for our categorical characterisation. We say that a graph $\mathtt{G}$ is *rooted* if there exists a (necessarily unique) event $\mathtt{e}\in\mathcal{E}_\mathtt{G}$ such that $\mathtt{G}=\mathtt{G}|_{\lfloor\mathtt{e}\rfloor}$.

▶ **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* $\mathtt{G}^{\ell}_{\mathcal{E}'} = \langle \mathcal{E}_{\top}, \prec \cup (\mathcal{E}' \times \{\top\}), \lambda[\top \mapsto \ell] \rangle$.

Here, $\mathcal{E}_{\top}$ denotes the extension of the set $\mathcal{E}$ with a new event $\top$, labelled into ℓ. Intuitively, $\mathtt{G}^{\ell}_{\mathcal{E}'}$ is obtained by adding to the visibility relation $\mathtt{G}$ a new event "seeing" some events in $\mathcal{E}'$. We call the inclusion $\mathtt{G} \to \mathtt{G}^{\ell}_{\mathcal{E}'}$ an *extension morphism*. Should $\mathtt{G}^{\ell}_{\mathcal{E}'}$ be rooted, we call it a *root* extension of $\mathtt{G}$, and the associated inclusion a root extension morphism.

▶ **Proposition 27.** *Rooted graphs form a family of separators of* $\mathbf{PDag}(\mathcal{L})$.

Proof. We need to show that for any pair of pr-morphisms $\mathtt{f}_1, \mathtt{f}_2 : \mathtt{G}_1 \to \mathtt{G}_2$ such that $\mathtt{f}_1 \neq \mathtt{f}_2$ there is a rooted graph $\mathtt{G}$ and a morphism $\mathtt{f} : \mathtt{G} \to \mathtt{G}_1$ such that $\mathtt{f};\mathtt{f}_1 \neq \mathtt{f};\mathtt{f}_2$. Given $\mathtt{e} \in \mathcal{E}_{\mathtt{G}_1}$ such that $\mathtt{f}_1(\mathtt{e}) \neq \mathtt{f}_2(\mathtt{e})$, it suffices to consider the pr-morphism $\mathtt{f} : \mathtt{G}_1|_{\lfloor \mathtt{e} \rfloor} \to \mathtt{G}_1$. ◀

We now further curb the arrows in $\mathbf{PDag}(\mathcal{L})$ to *monic* ones. Intuitively, we are only interested in what happens if we add further events to visibility relations. We thus consider the sub-category $\mathbf{PIDag}(\mathcal{L})$ of direct acyclic graphs and monic pr-morphisms. Note that the chosen morphism $\mathtt{f}$ in the proof of Proposition 27 is mono, since morphisms in $\mathbf{PDag}(\mathcal{L})$ are monic if and only if the underlying function is injective. We can then show that rooted graphs are also a family of generators for the sub-category $\mathbf{PIDag}(\mathcal{L})$.

We first need a technical lemma.

▶ **Lemma 28** (Monos under pushouts, 2). *Pushouts in* $\mathbf{PDag}(\mathcal{L})$ *preserve monos.*

We can then state an important characterisation of $\mathbf{PIDag}(\mathcal{L})$.

▶ **Proposition 29.** $\mathbf{PIDag}(\mathcal{L})$ *is the smallest sub-category of* $\mathbf{PDag}(\mathcal{L})$ *containing all root extension morphisms and closed under finite colimits.*

Proof. First, note that, since pushouts in $\mathbf{PDag}(\mathcal{L})$ preserve monos, the smallest sub-category of $\mathbf{PDag}(\mathcal{L})$ containing all root extensions and closed under finite colimits is surely a sub-category also of $\mathbf{PIDag}(\mathcal{L})$. So, given a monic pr-morphism $\mathtt{f} : \mathtt{G}_1 \to \mathtt{G}_2$, we need to prove that it can be generated from root extension morphisms via colimits. We proceed by induction on the cardinality of $\mathcal{E}_{\mathtt{G}_2}$.

If the cardinality is 0, then $\mathtt{f}$ must be the identity of the empty graph. Otherwise, consider $\mathtt{G}_2$ and assume that it is rooted with root $\mathtt{e}$. Now, if $\mathtt{e} \in \mathsf{img}(\mathtt{f})$, since the image of a pr-morphism is downward closed, it turns out that $\mathtt{f}$ is the identity of $\mathtt{G}_2$. If it is not in the image, then $\mathtt{f}$ can be decomposed as $\mathtt{G}_1 \to (\mathtt{G}_2 \setminus \mathtt{e}) \to \mathtt{G}_2$: the left-most is given by induction, while the right-most is a root extension morphism. Without loss of generality, let us assume that $\mathtt{G}_2$ has two distinct roots, namely $\mathtt{e}_1$ and $\mathtt{e}_2$, and that the image of $\mathtt{f}$ is contained in $\mathtt{G}_2|_{\lfloor \mathtt{e}_1 \rfloor}$. Now, $\mathtt{f}$ can be decomposed as $\mathtt{G}_1 \to \mathtt{G}_2|_{\lfloor \mathtt{e}_1 \rfloor} \to \mathtt{G}_2$: the left-most is given by induction, while the right-most is obtained via the pushout of the span $\mathtt{G}_2|_{\lfloor \mathtt{e}_1 \rfloor} \cap \mathtt{G}_2|_{\lfloor \mathtt{e}_2 \rfloor} \to \mathtt{G}_2|_{\lfloor \mathtt{e}_i \rfloor}$. ◀

7 A categorical correspondence

It is now the time for moving towards our categorical characterisation of specifications. In this section we will show that coherent specifications induce functors preserving the relevant categorical structure (soundness) and, conversely, that a certain class of functors (basically, those preserving finite colimits and binary pullbacks) induce coherent specifications (completeness). Finally, we will prove that these functions between functors and specifications are mutually inverse, establishing a one-to-one correspondence (up-to isomorphism).

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 $\mathcal{E}$ is downward closed, then $\mathcal{S}(\mathtt{G})|_{\mathcal{E}} \subseteq \mathcal{S}(\mathtt{G}|_{\mathcal{E}})$.*

Proof. Let $\mathcal{E}$ be downward closed, and note that this amounts to requiring $\mathcal{E} = \bigcup_{e \in \mathcal{E}} \lfloor \mathtt{e} \rfloor$, hence for all $\mathtt{e} \in \mathcal{E}$ we have that $(\mathtt{G}|_{\mathcal{E}})|_{\lfloor \mathtt{e} \rfloor} = \mathtt{G}|_{\lfloor \mathtt{e} \rfloor}$. By the latter and by coherence we have $\mathcal{S}(\mathtt{G})|_{\mathcal{E}} = (\bigotimes_{\mathtt{e} \in \mathcal{E}_{\mathtt{G}}} \mathcal{S}(\mathtt{G}|_{\lfloor \mathtt{e} \rfloor}))\Big|_{\mathcal{E}}$ 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 in the former can always be restricted to a suitable path with fewer events on the latter (the converse in general does not hold). ◀

7.1 Soundness

The notion of specification introduced in Definition 11 is oblivious to the existence of morphisms between graphs. In the following we impose a minimal consistency requirement, i.e., that a specification maps isomorphic graphs to isomorphic sets of paths, along the same isomorphism on events. That is, if there exists an isomorphism in **PDag** from $\mathtt{G}_1$ to $\mathtt{G}_2$ with underlying bijection $\mathtt{f} : \mathcal{E}_{\mathtt{G}_1} \to \mathcal{E}_{\mathtt{G}_2}$, then for all specifications $\mathcal{S}$ there is an isomorphism in $\mathbf{SPath}(\mathcal{L})$ from $\mathcal{S}(\mathtt{G}_1)$ to $\mathcal{S}(\mathtt{G}_2)$ with the same underlying function.

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

Proof. For $\mathtt{G}$ we define $\mathbb{M}(\mathcal{S})(\mathtt{G})$ as $\mathcal{S}(\mathtt{G})$ and for $\mathtt{f} : \mathtt{G} \to \mathtt{G}'$ we define $\mathbb{M}(\mathcal{S})(\mathtt{f})$ 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 showing 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 isomorphisms, 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 given by 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 $\mathcal{S}(\mathtt{G}') \subseteq \mathtt{sat}(\mathcal{S}(\mathtt{G}), \mathtt{f})$. ◀

It is a well-known fact that the category of sets and injective functions lacks pushouts. The same also holds for $\mathbf{PIDag}(\mathcal{L})$. However, recall now that pushouts in $\mathbf{PDag}(\mathcal{L})$ preserve monos (Lemma 28). Thus in the following we say that a functor $\mathbb{F} : \mathbf{PIDag}(\mathcal{L}) \to \mathbf{SPath}(\mathcal{L})$ weakly preserves finite pushouts (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 32.** *Let $\mathcal{S}$ be a 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. ◀

7.2 Completeness

It is now time for moving to the completeness results of our work, showing (a few alternatives on) how to obtain a specification from a functor.

▶ **Theorem 33.** *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 a coherent specification $\mathbb{S}(\mathbb{F})$.*

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

$$\begin{tikzcd} \mathtt{G}|_{\lfloor \mathtt{e}_1 \rfloor \cap \lfloor \mathtt{e}_2 \rfloor} \arrow[r, hook] \arrow[d, hook] & \mathtt{G}|_{\lfloor \mathtt{e}_2 \rfloor} \arrow[d, hook] \\ \mathtt{G}|_{\lfloor \mathtt{e}_1 \rfloor} \arrow[r, hook] & \mathtt{G}|_{\lfloor \mathtt{e}_1 \rfloor \cup \lfloor \mathtt{e}_2 \rfloor} \end{tikzcd} \tag{7.1}$$

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

$$\begin{tikzcd} \mathbb{F}(\mathtt{G}|_{\lfloor \mathtt{e}_1 \rfloor \cap \lfloor \mathtt{e}_2 \rfloor}) \arrow[r, hook] \arrow[d, hook] & \mathbb{F}(\mathtt{G}|_{\lfloor \mathtt{e}_2 \rfloor}) \arrow[d, hook] \\ \mathbb{F}(\mathtt{G}|_{\lfloor \mathtt{e}_1 \rfloor}) \arrow[r, hook] & \mathbb{F}(\mathtt{G}|_{\lfloor \mathtt{e}_1 \rfloor \cup \lfloor \mathtt{e}_2 \rfloor}) \end{tikzcd} \tag{7.2}$$

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

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

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

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

Isomorphism preservation follows from $\mathbb{F}$ being a functor. ◀

Combined with Theorem 32, the result above intuitively tells us that the coherence of a specification roughly corresponds to the weak preservation of colimits. However, the set-theoretical requirement $\mathbb{F}(\mathtt{G}) \subseteq \mathbb{P}(\mathcal{E}_\mathtt{G}, \lambda_\mathtt{G})$ is still unsatisfactory, yet apparently unavoidable, because a generic $\mathbb{F}$ could associate *any* set of paths to a graph. We can sharpen the result by requiring functors to preserve specific properties for suitable arrows of $\mathbf{PIDag}(\mathcal{L})$. The candidates are root extension morphisms, given the properties shown in Section 6. In order to define the functors, we also need to consider a suitable subset of the arrows of $\mathbf{SPath}(\mathcal{L})$.

▶ **Definition 34** (Saturated specifications). *Let $\mathcal{S}$ be a specification. It is* saturated *if for all graphs $\mathtt{G}$ and extensions $\mathtt{G}^\ell_\mathcal{E}$ 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 18), that is*

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

A *saturation ps-morphism* (along ℓ) is a saturated ps-morphism $\mathtt{f} : \mathcal{X}_1 \to \mathcal{X}_2$ with underlying function $(\mathcal{E}, \lambda) \to (\mathcal{E}_\top, \lambda[\top \mapsto \ell])$. We can now prove an instance of Theorem 33 concerning saturated specifications.

▶ **Proposition 35.** *Let $\mathbb{F} : \mathbf{PIDag}(\mathcal{L}) \to \mathbf{SPath}(\mathcal{L})$ be a functor mapping root extension morphisms into saturation ps-morphisms (along the same labels). If $\mathbb{F}$ weakly preserves finite colimits, it induces a saturated, coherent specification $\mathbb{S}(\mathbb{F})$.*

Proof. We first show that $\mathbb{F}$ preserves monos, which renders the assumption of Theorem 33 about preservation of pullbacks redundant. We will essentially follow the proof of Proposition 29. Given $\mathtt{f}\colon \mathtt{G}_1 \to \mathtt{G}_2$ in $\mathbf{PIDag}(\mathcal{L})$, we proceed by induction on the cardinality of $\mathcal{E}_{\mathtt{G}_2}$. If $\mathcal{E}_{\mathtt{G}_2}$ is $\emptyset$, i.e., it is the initial object, then $\mathtt{f}$ is the identity on $\emptyset$, and the claim follows by functors preserving identities. Suppose now that $\mathtt{G}_2$ is rooted with root $\mathtt{e}$. If $\mathtt{e} \in \mathsf{img}(\mathtt{f})$, then $\mathtt{f}$ again is the identity. Otherwise, $\mathtt{f}$ can be decomposed as $\mathtt{G}_1 \to (\mathtt{G}_2 \setminus \mathtt{e}) \to \mathtt{G}_2$: the left-most one satisfies the induction hypothesis, and the right-most one is a root extension morphism, which by hypothesis is mapped to a (monic) saturation ps-morphism. Therefore, by functoriality of $\mathbb{F}$, the claim holds for the composition of these morphisms. If $\mathtt{G}_2$ is not rooted, then $\mathtt{f}$ can be similarly decomposed as $\mathtt{G}_1 \to \mathtt{G}_2|_{\lfloor \mathtt{e}_1 \rfloor} \to \mathtt{G}_2$. By induction the claim holds for the left-most morphism. The right-most one is obtained via a pushout of the form (7.1), which is mapped by $\mathbb{F}$ to a pushout of the form (7.2), because $\mathbb{F}$ (weakly) preserves finite colimits. By induction hypothesis, the span of this pushout consists of monic ps-morphisms, therefore we use Lemma 24 to conclude that the pushout morphisms are monic as well, hence the right-most morphism satisfies our claim. Again, the claim for the whole of $\mathtt{f}$ follows from functoriality of $\mathbb{F}$. A similar inductive argument can be used to show that $\mathbb{F}(\mathtt{G})$ is a set of paths over $(\mathcal{E}_\mathtt{G}, \lambda_\mathtt{G})$ (up to a label-preserving isomorphism of events). Therefore we can now re-use the proof of Theorem 33 and obtain that $\mathbb{S}(\mathbb{F})$ is a coherent specification.

It remains to be shown that $\mathbb{S}(\mathbb{F})$ is saturated, that is $\mathbb{F}(\mathtt{G}^\ell_\mathcal{E}) = \mathtt{sat}(\mathtt{ret}(\mathbb{F}(\mathtt{G}^\ell_\mathcal{E}), \mathtt{f}), \mathtt{f})$. If $\mathtt{G}^\ell_\mathcal{E}$ is rooted, this follows from $\mathbb{F}$ mapping root extensions to saturation ps-morphisms. Otherwise, by coherence, $\mathbb{F}(\mathtt{G}^\ell_\mathcal{E})$ can be decomposed into the product $\bigotimes_{\mathtt{e} \in (\mathcal{E}_\mathtt{G})_\top} \mathbb{F}(\mathtt{G}^\ell_\mathcal{E}|_{\lfloor \mathtt{e} \rfloor})$. For each component of the product we have a root extension $\mathtt{G}^\ell_\mathcal{E}|_{\lfloor \mathtt{e} \rfloor} \setminus \mathtt{e} \to \mathtt{G}^\ell_\mathcal{E}|_{\lfloor \mathtt{e} \rfloor}$, which is mapped by $\mathbb{F}$ to a saturation ps-morphism, therefore we have $\mathbb{F}(\mathtt{G}^\ell_\mathcal{E}|_{\lfloor \mathtt{e} \rfloor}) = \mathtt{sat}(\mathtt{ret}(\mathbb{F}(\mathtt{G}^\ell_\mathcal{E}|_{\lfloor \mathtt{e} \rfloor}), \mathtt{f}_\mathtt{e}), \mathtt{f}_\mathtt{e})$, where $\mathtt{f}_\mathtt{e}$ is the underlying function between events of the root extension. Saturation of $\mathbb{F}(\mathtt{G}^\ell_\mathcal{E})$ follows by computing the product of these sets of paths. ◀

7.3 More Completeness

The need of finding a suitable image for root extension morphisms allows for alternative choices. To this end, we introduce a different subset of the arrows of $\mathbf{SPath}(\mathcal{L})$.

▶ **Definition 36** (Path extension/prefixing). *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 extension of* $\mathtt{P}$ *along* $\mathtt{f}$ *is defined as*

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

Similarly, let $\mathtt{Q}$ *be a path and* $\mathtt{f} : \mathcal{E} \to \mathcal{E}_\mathtt{Q}$ *a function preserving labels. The prefixing of* $\mathtt{Q}$ *along* $\mathtt{f}$ *is defined as*

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

Both definitions immediately extend to sets of paths. Should $\mathtt{f}$ be injective, $\mathtt{pre}(\mathtt{Q}, \mathtt{f})$ would be a singleton, and if $\mathtt{f}$ is an inclusion, then $\mathtt{pre}(\mathtt{Q}, \mathtt{f}) = \mathtt{Q}|_\mathcal{E}$, for the latter a prefix of $\mathtt{Q}$. Also, note that similarly $\mathtt{P}$ has to be a prefix for all the paths in $\mathtt{ext}(\mathtt{P}, \mathtt{f})$.

▶ **Example 37.** A topological specification $\mathcal{S}_{topR}$ for a `Register` can be defined as $\mathcal{S}_{lwwR}$ in Example 13 with the additional requirement that paths are topological orderings of visibilities

$$\mathtt{P} \in \mathcal{S}_{topR}(\mathtt{G}) \text{ iff } \mathtt{P} \in \mathcal{S}_{lwwR}(\mathtt{G}) \text{ and } \prec_\mathtt{G} \subseteq \leq_\mathtt{P}$$

In this way, $\mathcal{S}_{topR}(\mathtt{G})$ excludes e.g. the two right-most arbitrations of the equation in Figure 1a.

▶ **Definition 38** (Topological specifications). *Let* $\mathcal{S}$ *be a specification. It is* topological *if*

$$\forall \mathtt{G}, \mathcal{E}, \ell.\ \mathcal{S}(\mathtt{G}^\ell_\mathcal{E}) = \mathtt{ext}(\mathtt{pre}(\mathcal{S}(\mathtt{G}^\ell_\mathcal{E}), \mathtt{f}), \mathtt{f})$$

A *topological ps-morphism* (along ℓ) is a ps-morphism $\mathtt{f} : \mathcal{X}_1 \to \mathcal{X}_2$ with underlying function $(\mathcal{E}, \lambda) \to (\mathcal{E}_\top, \lambda[\top \mapsto \ell])$ such that $\mathcal{X}_2 = \mathtt{ext}(\mathtt{pre}(\mathcal{S}(\mathcal{X}_2), \mathtt{f}), \mathtt{f})$. The name is directly reminiscent of what are called topological RDTs in [10, 5], and in fact it similarly guarantees that arbitrations preserve the visibility order. We can thus prove another instance of Theorem 33, now concerning topological specifications.

▶ **Proposition 39.** *Let $\mathbb{F} : \mathbf{PIDag}(\mathcal{L}) \to \mathbf{SPath}(\mathcal{L})$ be a functor mapping root extension morphisms into topological ps-morphisms (along the same labels). If $\mathbb{F}$ weakly preserves finite colimits, it induces a topological, coherent specification $\mathbb{S}(\mathbb{F})$.*

7.4 Interchangeability of Functors and Coherent Specifications

The connection between the construction of Theorem 32 and Theorem 33 is quite tight, and in fact induces a one-to-one correspondence between functors and coherent specifications.

▶ **Theorem 40.** *Let $\mathcal{S}$ be a coherent specification. Then $\mathbb{S}(\mathbb{M}(\mathcal{S})) = \mathcal{S}$. Conversely, let $\mathbb{F} : \mathbf{PIDag}(\mathcal{L}) \to \mathbf{SPath}(\mathcal{L})$ be a functor verifying the hypothesis of Theorem 33. Then $\mathbb{M}(\mathbb{S}(\mathbb{F})) \simeq \mathbb{F}$.*

Proof. We first show that $\mathbb{M}(\mathbb{S}(\mathbb{F})) \simeq \mathbb{F}$. For notational convenience, we denote $\mathbb{M}(\mathbb{S}(\mathbb{F}))$ by $\mathbb{M}'$. We will show the existence of a natural isomorphism $\varphi \colon \mathbb{M}' \Rightarrow \mathbb{F}$. By definition, we have $\mathbb{M}'(\mathtt{G}) = \mathbb{S}(\mathbb{F})(\mathtt{G}) = \mathbb{F}(\mathtt{G})$, 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 showing $\mathbb{M}'(\mathtt{f}) = \mathbb{F}(\mathtt{f})$, 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 $(\mathcal{E}_{\mathtt{G}}, \lambda_{\mathtt{G}}) \to (\mathcal{E}_{\mathtt{G}'}, \lambda_{\mathtt{G}'})$.

Now we show that $\mathbb{S}(\mathbb{M}(\mathcal{S})) = \mathcal{S}$ for any coherent specification $\mathcal{S}$. This follows directly from the definition of $\mathbb{M}$ and $\mathbb{S}$. In fact, $\mathbb{S}(\mathbb{M}(\mathcal{S}))(\mathtt{G}) = \mathbb{M}(\mathcal{S})(\mathtt{G}) = \mathcal{S}(\mathtt{G})$. ◀

The one-to-one correspondence can be lifted to the specific classes of saturated/topological coherent specifications and to the functors of Proposition 35/Proposition 39, respectively. However, what is most relevant is the fact the interchangeability allows one to leverage the categorical machinery of the functor category for providing operators on specifications.

▶ Remark 41. Besides coherence, one of the keys of the previous correspondence is the (quite reasonable) choice of specifications that preserve isomorphisms. In general terms, whenever one needs to consider the relationship between different specifications, it is necessary to take into account how the underlying sets of events are related. This is quite easy to accomplish if we move to the functorial presentation. For example, we can say that a specification $\mathcal{S}_1$ refines a specification $\mathcal{S}_2$ if $\mathcal{S}_1(\mathtt{G}) \subseteq \mathcal{S}_2(\mathtt{G})$ for all graphs $\mathtt{G}$. However, this is a very concrete characterisation: it would be more general to check for the existence of a ps-morphism $\mathcal{S}_2(\mathtt{G}_2) \to \mathcal{S}_1(\mathtt{G}_1)$ whose underlying function $\mathtt{f} : \mathcal{E}_{\mathtt{G}_2} \to \mathcal{E}_{\mathtt{G}_1}$ is a bijection, in order to abstract from the identities of the events. In this case, a further constraint would be that $\mathtt{f}$ is preserved along the image of the morphisms in $\mathbf{PIDag}(\mathcal{L})$. These requirements boil down to the existence of a natural transformation $\mathbb{M}(\mathcal{S}_2) \to \mathbb{M}(\mathcal{S}_1)$.

8 Conclusions and Further Works

In this paper we have provided a functorial characterisation of RDT specifications. Our starting point is the denotational approach proposed in [7, 6], in which RDT specifications are associated with functions mapping visibility graphs into sets of admissible arbitrations that are also saturated and coherent, and where a preliminary functorial correspondence was proposed. In this paper we streamlined and expanded the latter result. We considered the category $\mathbf{PDag}(\mathcal{L})$ of labelled, acyclic graphs and pr-morphisms for representing visibility graphs. We equip $\mathbf{PDag}(\mathcal{L})$ with operators that model the evolution of visibility graphs and we show that the sub-category $\mathbf{PIDag}(\mathcal{L})$ of monic morphisms can be generated by the subset of root extensions via pushouts. 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 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 coherent specification $\mathbb{S}(\mathbb{F})$. Moreover, $\mathbb{M}(\mathcal{S})$ and $\mathbb{S}(\mathbb{F})$ are shown to be inverses of each other.

With respect to the categorical results expressed in [7], besides the additional characterisation of topological specifications, the key improvement has been the proof that the coherence of specifications has a precise counterpart in terms of the weak preservation of colimits on their functorial presentations, as stated by Theorem 32 and Theorem 33. We thus removed the set-theoretical requirements occurring e.g. in [7, Section 5.3], as witnessed by the definition of *coherent* functor there. We believe that this purely functorial characterisation of RDTs, as further witnessed by Proposition 35 and Proposition 39, provides an ideal setting for the development of techniques for handling RDT composition, as briefly pointed out by the functorial characterisation of refinement between specifications. Our long term goal is to equip RDT specifications with a set of operators that enables us to specify and reason about complex RDTs compositionally, i.e., in terms of their constituent parts. We aim at providing a uniform formal treatment to the compositional approaches proposed in [1, 10, 12].

References

1 Carlos Baquero, Paulo Sérgio Almeida, Alcino Cunha, and Carla Ferreira. Composition in state-based replicated data types. *Bulletin of the EATCS*, 123, 2017.

2 Ahmed Bouajjani, Constantin Enea, and Jad Hamza. Verifying eventual consistency of optimistic replication systems. In Suresh Jagannathan and Peter Sewell, editors, *POPL 2014*, pages 285–296. ACM, 2014.

3 Sebastian Burckhardt, Alexey Gotsman, and Hongseok Yang. Understanding eventual consistency. Technical Report MSR-TR-2013-39, Microsoft Research, 2013.

4 Sebastian Burckhardt, Alexey Gotsman, Hongseok Yang, and Marek Zawirski. Replicated data types: specification, verification, optimality. In Suresh Jagannathan and Peter Sewell, editors, *POPL 2014*, pages 271–284. ACM, 2014.

5 Andrea Cerone, Giovanni Bernardi, and Alexey Gotsman. A framework for transactional consistency models with atomic visibility. In Luca Aceto and David de Frutos-Escrig, editors, *CONCUR 2015*, volume 42 of *LIPIcs*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2015.

6 Fabio Gadducci, Hernán Melgratti, and Christian Roldán. On the semantics and implementation of replicated data types. *Science of Computer Programming*, 167:91–113, 2018.

7 Fabio Gadducci, Hernán C. Melgratti, and Christian Roldán. A denotational view of replicated data types. In Jean-Marie Jacquet and Mieke Massink, editors, *COORDINATION 2017*, volume 10319 of *LNCS*, pages 138–156. Springer, 2017.

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

9 Alexey Gotsman and Sebastian Burckhardt. Consistency models with global operation sequencing and their composition. In Andréa W. Richa, editor, *DISC 2017*, volume 91 of *LIPIcs*, pages 23:1–23:16. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2017.

10 Alexey Gotsman and Hongseok Yang. Composite replicated data types. In Jan Vitek, editor, *ESOP 2015*, volume 9032 of *LNCS*, pages 585–609. Springer, 2015.

11 Gowtham Kaki, Kapil Earanky, K. C. Sivaramakrishnan, and Suresh Jagannathan. Safe replication through bounded concurrency verification. In *OOPSLA 2018*, volume 2 of *PACMPL*, pages 164:1–164:27. ACM, 2018.

12 Adriaan Leijnse, Paulo Sérgio Almeida, and Carlos Baquero. Higher-order patterns in replicated data types. In *PaPoC 2019*. ACM, 2019.

13 Marc Shapiro, N. Preguiça, Carlos Baquero, and Marek Zawirski. Conflict-free replicated data types. In Xavier Défago, Franck Petit, and Vincent Villain, editors, *SSS 2011*, volume 6976 of *LNCS*, pages 386–400. Springer, 2011.

14 K. C. Sivaramakrishnan, Gowtham Kaki, and Suresh Jagannathan. Declarative programming over eventually consistent data stores. In David Grove and Steve Blackburn, editors, *PLDI 2015*, pages 413–424. ACM, 2015.