Science of Computer Programming 167 (2018) 91–113

Contents lists available at ScienceDirect

Science of Computer Programming

www.elsevier.com/locate/scico

On the semantics and implementation of replicated data types

Fabio Gadducci [a], Hernán Melgratti [b,c], Christian Roldán [b,*]

[a] Università di Pisa, Dipartimento di Informatica, Pisa, Italy
[b] Universidad de Buenos Aires, Facultad de Ciencias Exactas y Naturales, Departamento de Computación, Buenos Aires, Argentina
[c] CONICET-Universidad de Buenos Aires, Instituto de Investigación en Ciencias de la Computación (ICC), Buenos Aires, Argentina

ARTICLE INFO

Article history:
Received 2 November 2017
Received in revised form 7 May 2018
Accepted 20 June 2018
Available online xxxx

Keywords:
Replicated data types
Specification
Implementation correctness

ABSTRACT

Replicated data types (RDTs) concern the specification and implementation of data structures handled by replicated data stores, i.e., distributed data stores that maintain copies of the same data item on multiple devices. A distinctive feature of RDTs is that the behaviour of an operation depends on the state of the replica over which it performs, and hence, its result may differ from replica to replica. Abstractly, RDTs are specified in terms of two relations, *visibility* and *arbitration*. The former establishes whether an operation observes the effects of the execution of another operation, the latter is a total order on operations used to resolve conflicts between operations executed concurrently over different replicas. Traditionally, an operation of an RDT is specified as a function mapping a visibility and an arbitration into the expected result of the operation. This paper recasts such standard approaches into a denotational framework in which a data type is a function mapping visibility into admissible arbitrations. This characterisation provides a more abstract view of RDTs that (i) highlights some implicit assumptions shared in operational approaches to specification; (ii) accommodates underspecification and refinement; (iii) enables a direct characterisation of the correct implementations of an RDT in terms of a simulation relation between the states of a concrete implementation and of the abstract one determined by the specification.

© 2018 Published by Elsevier B.V.

1. Introduction

Distributed systems replicate their state over different nodes in order to satisfy several non-functional requirements, such as performance, availability, and reliability. It then becomes crucial to keep a consistent view of the replicated data. However, this is a challenging task because consistency is in conflict with two common requirements of distributed applications: *availability* (every request is eventually executed) and tolerance to network *partitions* (the system operates even in the presence of failures that prevent communication among components). In fact, it is impossible for a system to simultaneously achieve strong Consistency, Availability and Partition tolerance [1]. Since many domains cannot renounce availability or avoid network partitions, developers need to cope with weaker notions of consistency by allowing, e.g., replicas to (temporarily) exhibit some discrepancies, as long as they eventually converge to the same state.

This setting challenges the way in which data is specified: states, state transitions and return values should account for the different views that a data item may simultaneously have. Consider a data type `Register`: a memory cell that is read and updated by, respectively, operations `rd` and `wr`. In a replicated scenario, the value obtained when reading a register

* Corresponding author.
E-mail addresses: fabio.gadducci@unipi.it (F. Gadducci), hmelgra@dc.uba.ar (H. Melgratti), croldan@dc.uba.ar (C. Roldán).

https://doi.org/10.1016/j.scico.2018.06.003
0167-6423/© 2018 Published by Elsevier B.V.

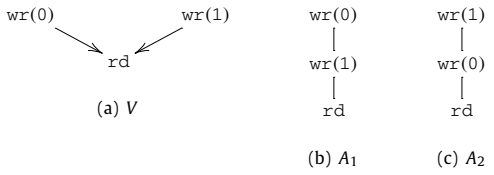


Fig. 1. A scenario for the replicated data type `Register`.

after two concurrent updates `wr(0)` and `wr(1)` (i.e., updates taking place over different replicas) is affected by the way in which updates propagate among replicas: the result might be (i) undefined (when the read is performed over a third replica that has not received any of the updates), (ii) 0, or (iii) 1. Basically, the return value depends on the updates that are seen by that read operation. Choosing the return value is straightforward if a read sees just one update, less so if a read is performed over a replica that knows of both updates, since all replicas should consistently pick the same value among the available ones. A common strategy for registers is that the *last-write wins*: the last update is chosen when several concurrent updates are observed. This strategy implicitly assumes that all the events in a system can be arranged in a total order. Several recent approaches focus on the operational specification of replicated data types [2–9]. Usually, the specification describes the meaning of an operation in terms of two relations among events: *visibility*, which explains the causes for each result, and *arbitration*, which totally orders events. Consider the visibility relation V in Fig. 1a and the arbitrations A_1 and A_2 in Fig. 1b and Fig. 1c, respectively. The meaning of `rd` is such that $\mathtt{rd}(V, A_1) = 1$ and $\mathtt{rd}(V, A_2) = 0$. We remark that operational approaches require specifications to be *functional*: for every operation, visibility and arbitration, there is exactly one return value. In this way operational specifications commit to concrete policies for resolving conflicts.

This work aims at putting on firm grounds the operational approaches for RDTs by giving them a purely functional description. In our view, RDTs are functions that map visibility graphs (i.e., configurations) into sets of admissible arbitrations, i.e., all the executions that generate a particular configuration. In this setting, a configuration mapped to an empty set of admissible arbitrations stands for an unreachable configuration, i.e., a configuration that cannot be explained in terms of any arbitration. We rely on such an abstract view of RDTs to highlight some of the implicit assumptions shared by most of the operational approaches. In particular, we characterise operational approaches, such as [2,3], as those specifications that satisfy three properties: besides the evident requirement of being (locally) *functional* (i.e., deterministic and total), they must be *coherent* (i.e., larger states are explained as the composition of smaller ones) and *saturated* (e.g., an operation that has not been seen by any other operation can be arbitrated in any position, even before the events that it sees). We show this inclusion to be strict and discuss some interesting cases that do not fall in this class. Moreover, we show that our formulation elegantly accounts for underspecification and refinement, which are standard notions in data type specification.

The notion of implementation correctness, which is central to the theory of abstract data types [10–12], relates the expected behaviour of a family of operations as defined by a specification with the one that is provided by a more concrete realisation. In a replicated scenario, such concrete realisations consist of several replicas that keep their own local state and propagate changes asynchronously. On the one hand, we assume that the behaviour of an implementation is given in terms of two *labelled transition systems* (LTSs): one that describes a single replica and another, which is obtained by composition, that accounts for the joint behaviour of several replicas. Technically, this is achieved by providing a composition operator over LTSs that reflects the adopted communication model. On the other hand, we note that our specifications have an implicit operational interpretation, which describes the expected behaviours of a single replica and of the composition of several replicas. Technically, each specification induces two LTSs: one, called *one-replica*, prescribes the behaviour of a single replica, and another, called *multi-replica*, defines the behaviour of multiple replicas. Then, implementation correctness is defined in terms of simulation relations between the LTSs associated with an implementation, i.e., a replica or a set of replicas, with the LTS corresponding to the specification, i.e., one-replica or multi-replica. We show that implementation correctness is preserved under standard parallel composition (synchronous or asynchronous buffered communication). Consequently, in order to show that an implementation is correct, we only need to show that a single replica is correct. We illustrate the approach with the implementation of a few well-known RDTs.

The paper has the following structure. Section 2 introduces the basic definitions concerning labelled directed acyclic graphs. Section 3 discusses our functional mechanism for the presentation of Replicated Data Types. Section 4 compares our proposal with the classical operational one [4]. Section 5 studies the correctness of the replicated data types implementations with respect to our specifications. Finally, in the closing section we draw some conclusions, discuss related works, and highlight further developments.

This paper is a revised and extended version of [13]. We enrich our previous work by providing an approach to assess whether an implementation of an RDT on top of several concurrent replicas is correct (the material in § 5 is completely new to this paper). In addition, we provide full proofs of already published results.

2. Labelled directed acyclic graphs

In this section we recall the basics of labelled directed acyclic graphs, which are used for our description of replicated data types. We rely on countable sets $\mathcal{E}$ of events $\mathsf{e}, \mathsf{e}', \ldots, \mathsf{e}_1, \ldots$ and $\mathcal{L}$ of labels $\ell, \ell', \ldots, \ell_1, \ldots$

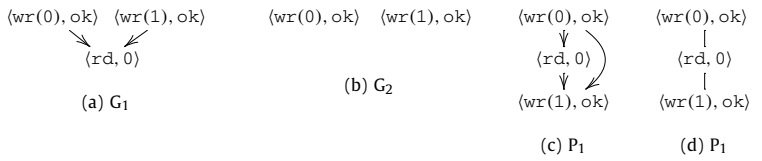


Fig. 2. Two simple LDAGs and two paths.

Definition 1 *(Labelled directed acyclic graph).* A *Labelled Directed Acyclic Graph* (LDAG) *over* a set of labels $\mathcal{L}$ is a triple $\mathtt{G} = \langle \mathcal{E}_\mathtt{G}, \prec_\mathtt{G}, \lambda_\mathtt{G} \rangle$ such that $\mathcal{E}_\mathtt{G}$ is a set of events, $\prec_\mathtt{G} \subseteq \mathcal{E}_\mathtt{G} \times \mathcal{E}_\mathtt{G}$ is a binary relation whose transitive closure is a strict partial order, and $\lambda_\mathtt{G} : \mathcal{E}_\mathtt{G} \to \mathcal{L}$ is a labelling function. An LDAG $\mathtt{G}$ is a *path* if $\prec_\mathtt{G}$ is a strict total order.

We write $\mathbb{G}(\mathcal{L})$ and $\mathbb{P}(\mathcal{L})$ to respectively denote the sets of LDAGs and paths over $\mathcal{L}$. We use $\mathtt{G}$ to range over $\mathbb{G}(\mathcal{L})$ and $\mathtt{P}$ to range over $\mathbb{P}(\mathcal{L})$. Moreover, we write $<_\mathtt{P}$ instead of $\prec_\mathtt{P}$ to make evident that paths are total orders. We say that $\mathtt{P} = \langle \mathcal{E}_\mathtt{P}, <_\mathtt{P}, \lambda_\mathtt{P} \rangle$ is a path over $\mathcal{E}$ if $\mathcal{E}_\mathtt{P} = \mathcal{E}$ and write $\mathbb{P}(\mathcal{E}, \lambda)$ for $\{\mathtt{P} \mid \mathtt{P}$ is a path over $\mathcal{E}$ and $\lambda_\mathtt{P} = \lambda\}$. We usually omit the subscript $\mathtt{G}$ (or $\mathtt{P}$) when referring to the elements of $\mathtt{G}$ (of $\mathtt{P}$, respectively) when no confusion arises. We write ε for the empty LDAG, i.e., such that $\mathcal{E}_\varepsilon = \emptyset$.

Example 1. Consider the set $\mathcal{L} = \{\langle \mathtt{rd}, 0\rangle, \langle \mathtt{rd}, 1\rangle, \langle \mathtt{wr}(0), \mathtt{ok}\rangle, \langle \mathtt{wr}(1), \mathtt{ok}\rangle\}$ of labels that describe the operations of a 1-bit register. Each label is a tuple $\langle op, rv\rangle$ where *op* denotes an operation and *rv* its return value. For homogeneity, we associate the return value $\mathtt{ok}$ to every write operation. Now, consider the LDAG over $\mathcal{L}$ that is defined as $\mathtt{G}_1 = \langle \{\mathsf{e}_1, \mathsf{e}_2, \mathsf{e}_3\}, \prec, \lambda\rangle$, where $\prec = \{(\mathsf{e}_1, \mathsf{e}_3), (\mathsf{e}_2, \mathsf{e}_3)\}$ and λ is such that $\lambda(\mathsf{e}_1) = \langle \mathtt{wr}(0), \mathtt{ok}\rangle$, $\lambda(\mathsf{e}_2) = \langle \mathtt{wr}(1), \mathtt{ok}\rangle$, and $\lambda(\mathsf{e}_3) = \langle \mathtt{rd}, 0\rangle$. A graphical representation of $\mathtt{G}_1$ is provided in Fig. 2a. Note that we do not depict the events and just write instead the corresponding labels when no confusion arises. A representation of the LDAG $\mathtt{G}_2$, where $\prec_{\mathtt{G}_2}$ is empty, is in Fig. 2b. Neither $\mathtt{G}_1$ nor $\mathtt{G}_2$ is a path because they are not total orders. $\mathtt{P}_1$ in Fig. 2c is an LDAG that is also a path. Hereinafter we use undirected arrows when depicting paths and avoid drawing transitions that are obtained by transitivity, as shown in Fig. 2d. All LDAGs in Fig. 2 belong to $\mathbb{G}(\mathcal{L})$, but only $\mathtt{P}_1$ is in $\mathbb{P}(\mathcal{L})$.

2.1. LDAG operations

We now present a few operations on LDAGs that will be used in the following sections. We start by introducing some notation for binary relations. We write Id for the identity relation over events and $\preccurlyeq$ for the reflexive closure $\prec \cup \mathrm{Id}$. Moreover, we will use $\prec^+$ and $\prec^*$ for respectively the transitive closure of $\prec$ and $\preccurlyeq$. We write $- \prec \mathsf{e}$ (and similarly $- \preccurlyeq \mathsf{e}$, $- \prec^+ \mathsf{e}$, and $- \prec^* \mathsf{e}$) for the preimage of e, i.e., $- \prec \mathsf{e} = \{\mathsf{e}' \mid \mathsf{e}' \prec \mathsf{e}\}$. We use $\prec|_\mathcal{E}$ for the restriction of $\prec$ to the elements in $\mathcal{E}$, i.e. $\prec|_\mathcal{E} = \prec \cap (\mathcal{E} \times \mathcal{E})$. Analogously, $\lambda|_\mathcal{E}$ is the domain restriction of λ to the elements in $\mathcal{E}$. We write $\mathcal{E}_\top$ for the extension of the set $\mathcal{E}$ with a fresh element, i.e., $\mathcal{E}_\top = \mathcal{E} \cup \{\top\}$ such that $\top \notin \mathcal{E}$.

Definition 2 *(Restriction and extension).* Let $\mathtt{G} = \langle \mathcal{E}, \prec, \lambda\rangle$ and $\mathcal{E}' \subseteq \mathcal{E}$. We define

- $\mathtt{G}|_{\mathcal{E}'} = \langle \mathcal{E}', \prec|_{\mathcal{E}'}, \lambda|_{\mathcal{E}'}\rangle$ as the *restriction of* $\mathtt{G}$ *to* $\mathcal{E}'$;
- $\mathtt{G}^\ell_{\mathcal{E}'} = \langle \mathcal{E}_\top, \prec \cup (\mathcal{E}' \times \{\top\}), \lambda[\top \mapsto \ell]\rangle$ as the *extension of* $\mathtt{G}$ *over* $\mathcal{E}'$ *with* ℓ.

Restriction obviously lifts to sets $\mathcal{X}$ of LDAGs, i.e., $\mathcal{X}|_\mathcal{E} = \{\mathtt{G}|_\mathcal{E} \mid \mathtt{G} \in \mathcal{X}\}$. We omit the subscript $\mathcal{E}'$ in $\mathtt{G}^\ell_{\mathcal{E}'}$ when $\mathcal{E}' = \mathcal{E}$.

Example 2. Consider the LDAGs $\mathtt{G}_1$ and $\mathtt{G}_2$ depicted in Fig. 2a and Fig. 2b, respectively. Then, $\mathtt{G}_2 = \mathtt{G}_1|_{-\prec \mathsf{e}_3}$ and $\mathtt{G}_1$ is isomorphic (as a graph) to $\mathtt{G}_2{}^{\langle \mathtt{rd}, 0\rangle}$. Indeed, $\mathtt{G}_1$ can be obtained from $\mathtt{G}_2$ by adding a new node labelled by $\langle \mathtt{rd}, 0\rangle$, in such a way that the new node is related with every node in $\mathtt{G}_2$ via $\prec$.

The following operator allows for the combination of several paths and plays a central role in our characterisation of replicated data types.

Definition 3 *(Product).* Let $\mathcal{X} = \{\langle \mathcal{E}_i, <_i, \lambda_i\rangle\}_i$ be a set of paths such that $\forall \mathsf{e}, i, j . \mathsf{e} \in \mathcal{E}_i \cap \mathcal{E}_j$ implies $\lambda_i(\mathsf{e}) = \lambda_j(\mathsf{e})$. The product of $\mathcal{X}$ is

$$\bigotimes \mathcal{X} = \{\mathcal{Q} \mid \mathcal{Q} \text{ is a path over } \bigcup_i \mathcal{E}_i \text{ and } \mathcal{Q}|_{\mathcal{E}_i} \in \mathcal{X}\}$$

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.

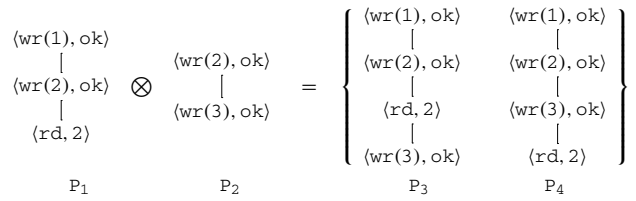


Fig. 3. Product between two paths.

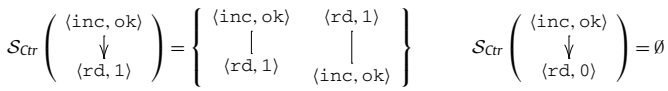


Fig. 4. `Counter` specification.

Example 3. Consider the paths $\mathtt{P}_1$ and $\mathtt{P}_2$ in Fig. 3, which share the event labelled $\langle \mathtt{wr}(2), \mathtt{ok} \rangle$. Their product has two paths $\mathtt{P}_3$ and $\mathtt{P}_4$, each of them contains the elements of $\mathtt{P}_1$ and $\mathtt{P}_2$ and preserves the relative order of the elements in the original paths. We remark that the product is empty when the paths have incompatible orders. For instance, $\mathtt{P}_3$ and $\mathtt{P}_4$ have the same set of elements yet incompatible orders, thus $\mathtt{P}_3 \otimes \mathtt{P}_4 = \emptyset$.

It is straightforward to show that $\otimes$ is associative and commutative. Hence, we freely use $\otimes$ over sets of sets of paths.

3. Specifications

We introduce our notion of specification and apply it to well-known data types.

Definition 4 *(Specification).* A *specification* $\mathcal{S}$ is a function $\mathcal{S} : \mathbb{G}(\mathcal{L}) \to 2^{\mathbb{P}(\mathcal{L})}$ such that $\mathcal{S}(\varepsilon) = \{\varepsilon\}$ and $\forall \mathtt{G}.\ \mathcal{S}(\mathtt{G}) \in 2^{\mathbb{P}(\mathcal{E}_\mathtt{G}, \lambda_\mathtt{G})}$.

A specification $\mathcal{S}$ maps an LDAG (i.e., a visibility relation) to a set of paths (i.e., its admissible arbitrations). Note that $\mathtt{P} \in \mathcal{S}(\mathtt{G})$ is a path over $\mathcal{E}_\mathtt{G}$, and hence a total order of the events in $\mathtt{G}$. However, we do not require $\mathtt{P}$ to be a topological ordering of $\mathtt{G}$, i.e., $\prec_\mathtt{G} \subseteq <_\mathtt{P}$ may not hold. Although some specification approaches consider only arbitrations that include visibility [6,7], our definition accommodates also presentations, such as [3,4], in which arbitrations may not preserve visibility. Since our approach is independent of this choice, we adopted the most liberal presentation. We also remark that it could be the case that $\mathcal{S}(\mathtt{G}) = \emptyset$, which means that $\mathcal{S}$ forbids the configuration $\mathtt{G}$ (more details in Example 4 below). For technical convenience, we impose $\mathcal{S}(\varepsilon) = \{\varepsilon\}$ and disallow $\mathcal{S}(\varepsilon) = \emptyset$: a specification cannot forbid the empty configuration, which denotes the initial state of a data type.

We now illustrate the specification of some well-known replicated data types.

Example 4 *(Counter).* The data type `Counter` provides operations for incrementing and reading an integer register with initial value `0`. A read operation returns the number of increments seen by that read. An increment is always successful and returns the value `ok`. Formally, we consider the set of labels $\mathcal{L} = \{\langle \mathtt{inc}, \mathtt{ok} \rangle\} \cup (\{\mathtt{rd}\} \times \mathbb{N})$. Then, the specification of `Counter` is given by $\mathcal{S}_{Ctr}$ defined such that

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

A visibility graph $\mathtt{G}$ has admissible arbitrations (i.e., $\mathcal{S}_{Ctr}(\mathtt{G}) \neq \emptyset$) only when each event e in $\mathtt{G}$ labelled by `rd` has a return value $\mathtt{k}$ that matches the number of increments anteceding e in $\mathtt{G}$. We illustrate two cases for the definition of $\mathcal{S}_{Ctr}$ in Fig. 4. While the configuration in Fig. 4a has admissible arbitrations, the one in Fig. 4b has not, because the unique event labelled by `rd` returns `0` when it is actually preceded by an observed increment. In other words, an execution is not allowed to generate such a visibility graph. We remark that $\mathcal{S}_{Ctr}$ imposes no constraint on the ordering $<_\mathtt{P}$. In fact, a path $\mathtt{P} \in \mathcal{S}_{Ctr}(\mathtt{G})$ does not need to be a topological ordering of $\mathtt{G}$ as, for instance, the rightmost path in the set of Fig. 4a.

Example 5 *(Last-write-wins register).* A `Register` stores a value that can be read and updated. We assume that the initial value of a register is undefined. Take $\mathcal{L} = \{\langle \mathtt{wr}(\mathtt{k}), \mathtt{ok} \rangle \mid \mathtt{k} \in \mathbb{N}\} \cup (\{\mathtt{rd}\} \times \mathbb{N} \cup \{\bot\})$ as the set of labels. Then $\mathcal{S}_{lwwR}$ gives the semantics of a register that adopts the last-write-wins strategy.

$$\mathtt{P} \in \mathcal{S}_{lwwR}(\mathtt{G}) \quad \text{iff}$$

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

An LDAG $\mathtt{G}$ has admissible arbitrations only when each event associated with a read returns a previously written value. As per the first condition above, a read operation returns the undefined value $\bot$ when it does not see any write. By the second condition, a read e returns a natural number $\mathtt{k}$ when it sees an operation e' that writes the value $\mathtt{k}$. In such case, any admissible arbitration $\mathtt{P}$ must order e' as the greatest (accordingly to $<_{\mathtt{P}}$) among all the write operations seen by e.

Example 6 *(Generic Register).* We now define a `Generic Register` that does not commit to a particular strategy for resolving conflicts.

$$\mathtt{P} \in \mathcal{S}_{gR}(\mathtt{G}) \quad \text{iff}$$

$$\forall \mathsf{e} \in \mathcal{E}_{\mathtt{G}}.\ \begin{cases} \lambda(\mathsf{e}) = \langle \mathtt{rd}, \bot \rangle \text{ implies } \forall \mathsf{e}' \prec_{\mathtt{G}} \mathsf{e}. \forall \mathtt{k}. \lambda(\mathsf{e}') \neq \langle \mathtt{wr}(\mathtt{k}), \mathtt{ok} \rangle \\ \forall \mathtt{k}. \lambda(\mathsf{e}) = \langle \mathtt{rd}, \mathtt{k} \rangle \text{ implies } \exists \mathsf{e}' \prec_{\mathtt{G}} \mathsf{e}. \lambda(\mathsf{e}') = \langle \mathtt{wr}(\mathtt{k}), \mathtt{ok} \rangle \text{ and} \\ \quad \forall \mathsf{e}''. \forall \mathtt{k}''. \lambda(\mathsf{e}'') = \langle \mathtt{rd}, \mathtt{k}'' \rangle \text{ and } - \prec_{\mathtt{G}} \mathsf{e} = - \prec_{\mathtt{G}} \mathsf{e}'' \text{ implies } \mathtt{k} = \mathtt{k}'' \end{cases}$$

As in Example 5, the return value of a read corresponds to a written value seen by that read, but the specification does not determine which value should be chosen. We require instead that all read operations with the same causes (i.e., $- \prec_{\mathtt{G}} \mathsf{e} = - \prec_{\mathtt{G}} \mathsf{e}'$) have the same result. Since this condition has to be satisfied by any admissible configuration $\mathtt{G}$, it ensures convergence. Requiring convergence seems meaningful since we are identifying some minimal conditions ensuring the correctness of a specification without making any assumption on the chosen paths. Indeed, convergence can be proved for what we call *deterministic* specifications (i.e., specifications in which the return value of each operation is uniquely determined by the visibility and arbitration relations, as formally characterised in Section 4.1), whose instances are in e.g. Example 4 and Example 5. Thus, our proposal only apparently disagree with approaches like [3,4], where convergence is ensured automatically since the specifications they consider are implicitly deterministic (as shown formally in Section 4.2).

3.1. Refinement

Refinement is a standard approach in data type specification, which allows for a hierarchical organisation that goes from abstract descriptions to concrete implementations. The main benefit of refinement relies on the fact that applications can be developed and reasoned about in terms of abstract data types, which hide implementation details and leave some freedom for the implementation. Consider the specification $\mathcal{S}_{gR}$ of the `Generic Register` introduced in Example 6, which only requires a policy for conflict resolution that ensures convergence. On the contrary, the specification $\mathcal{S}_{lwwR}$ in Example 5 explicitly states that concurrent updates must be resolved by adopting the last-write-wins policy. Since the latter policy ensures convergence, we would like to think about $\mathcal{S}_{lwwR}$ as a refinement of $\mathcal{S}_{gR}$. We characterise refinement in our setting as follows.

Definition 5 *(Refinement).* Let $\mathcal{S}_1, \mathcal{S}_2$ be specifications. We say that $\mathcal{S}_1$ refines $\mathcal{S}_2$ and write $\mathcal{S}_1 \sqsubseteq \mathcal{S}_2$ if $\forall \mathtt{G}.\ \mathcal{S}_1(\mathtt{G}) \subseteq \mathcal{S}_2(\mathtt{G})$.

Example 7. It can be easily checked that $\mathtt{P} \in \mathcal{S}_{lwwR}(\mathtt{G})$ implies $\mathtt{P} \in \mathcal{S}_{gR}(\mathtt{G})$ for any $\mathtt{G}$. Consequently, $\mathcal{S}_{lwwR}$ is a refinement of $\mathcal{S}_{gR}$.

Example 8. Consider the data type `Set`, which provides (among others) the operations `add`, `rem` and `lookup` for respectively adding, removing, and examining the elements within a set. Different alternatives have been proposed in the literature for resolving conflicts in the presence of concurrent additions and removals of elements (see [14] for a detailed discussion). We illustrate two possible alternatives by considering the execution scenario depicted in Fig. 5, for $\mathtt{P}$ a topological ordering of $\mathtt{G}$. A reasonable semantics for `lookup` over $\mathtt{G}$ and $\mathtt{P}$ would fix the return value $\mathtt{S}$, which should contain all the elements in the set, as one of the following two values: $\emptyset$ or $\{1\}$. In fact, under the last-write-wins policy, the specification prescribes that `lookup` returns $\{1\}$ in this scenario. Differently, the strategy of `2P-Set`[1] establishes that the result is $\emptyset$.

The following definition provides a specification for an abstract data type `Set` that allows (among others) any of the above policies.

[1] In `2P-Set`, the addition of elements that have been previously removed has no effect.

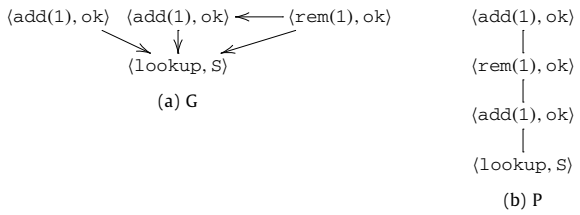


Fig. 5. A scenario for the replicated data type `Set`.

$$\mathtt{P} \in \mathcal{S}_{Set}(\mathtt{G}) \text{ iff } \forall \mathsf{e} \in \mathcal{E}_{\mathtt{G}}.\forall \mathtt{S} \in 2^{\mathbb{N}}.\lambda(\mathsf{e}) = \langle \mathtt{lookup}, \mathtt{S}\rangle \text{ implies } B_{\mathsf{e}} \subseteq \mathtt{S} \subseteq A_{\mathsf{e}} \text{ and } \mathtt{Conv}_{\mathsf{e},\mathtt{S}}$$

where

$$A_{\mathsf{e}} = \{\mathtt{k} \mid \exists \mathsf{e}' \in \mathcal{E}_{\mathtt{G}}.\mathsf{e}' \prec_{\mathtt{G}} \mathsf{e} \text{ and } \lambda(\mathsf{e}') = \langle \mathtt{add}(\mathtt{k}), \mathtt{ok}\rangle\}$$
$$B_{\mathsf{e}} = A_{\mathsf{e}} \setminus \{\mathtt{k} \mid \exists \mathsf{e}' \in \mathcal{E}_{\mathtt{G}}.\mathsf{e}' \prec_{\mathtt{G}} \mathsf{e} \text{ and } \lambda(\mathsf{e}') = \langle \mathtt{rem}(\mathtt{k}), \mathtt{ok}\rangle\}$$
$$\mathtt{Conv}_{\mathsf{e},\mathtt{S}} = \forall \mathsf{e}' \in \mathcal{E}_{\mathtt{G}}.\forall \mathtt{S}' \in 2^{\mathbb{N}}.\lambda(\mathsf{e}') = \langle \mathtt{lookup}, \mathtt{S}'\rangle \text{ and } - \prec_{\mathtt{G}} \mathsf{e} = - \prec_{\mathtt{G}} \mathsf{e}' \text{ implies } \mathtt{S} = \mathtt{S}'$$

The set A_{e} contains the elements added to (and possibly removed from) the set seen by e while B_{e} contains those elements for which e sees no removal. Thus, the condition $B_{\mathsf{e}} \subseteq \mathtt{S} \subseteq A_{\mathsf{e}}$ states that `lookup` returns a set that contains at least all the elements added but not removed (i.e., in B_{e}). However, the return value `S` may contain elements that have been added and later removed (the choice is left unspecified). Analogously to the specification of $\mathcal{S}_{gR}$ in Example 6, $\mathtt{Conv}_{\mathsf{e},\mathtt{S}}$ ensures convergence.

Then, a concrete resolution policy such as `2P-Set` can be specified as follows

$$\mathtt{P} \in \mathcal{S}_{2P\text{-}Set}(\mathtt{G}) \text{ iff } \forall \mathsf{e} \in \mathcal{E}_{\mathtt{G}}.\forall \mathtt{S} \in 2^{\mathbb{N}}.\lambda(\mathsf{e}) = \langle \mathtt{lookup}, \mathtt{S}\rangle \text{ implies } \mathtt{S} = B_{\mathsf{e}}$$

A different policy, called `Or-Set`, states that additions win against remove operations. This policy can be defined as follows

$$\mathtt{P} \in \mathcal{S}_{Or\text{-}Set}(\mathtt{G}) \text{ iff } \forall \mathsf{e} \in \mathcal{E}_{\mathtt{G}}.\forall \mathtt{S} \in 2^{\mathbb{N}}.\lambda(\mathsf{e}) = \langle \mathtt{lookup}, \mathtt{S}\rangle \text{ implies } \mathtt{S} = C_{\mathsf{e}}$$

where

$$C_{\mathsf{e}} = \{\mathtt{k} \mid \exists \mathsf{e}' \in \mathcal{E}_{\mathtt{G}}.\mathsf{e}' \prec_{\mathtt{G}} \mathsf{e} \text{ and } \lambda(\mathsf{e}') = \langle \mathtt{add}(\mathtt{k}), \mathtt{ok}\rangle \text{ and } \forall \mathsf{e}''.\mathsf{e}' \prec_{\mathtt{G}} \mathsf{e}'' \prec_{\mathtt{G}} \mathsf{e} \text{ implies } \lambda(\mathsf{e}'') \neq \langle \mathtt{rem}(\mathtt{k}), \mathtt{ok}\rangle\}.$$

It is immediate to note that $\mathcal{S}_{2P\text{-}Set}$ is a refinement of $\mathcal{S}_{Set}$. It can be also noticed that $B_{\mathsf{e}} \subseteq C_{\mathsf{e}} \subseteq A_{\mathsf{e}}$. Consequently, $\mathcal{S}_{Or\text{-}Set}$ is also a refinement of $\mathcal{S}_{Set}$.

3.2. Classes of specifications

We discuss two properties of specifications. Firstly, we look at specifications for which the behaviour of larger computations matches that of their prefixes.

Definition 6 *(Past-coherent specification).* Let $\mathcal{S}$ be a specification. We say that $\mathcal{S}$ is past-coherent (briefly, coherent) if

$$\forall \mathtt{G}.\ \mathcal{S}(\mathtt{G}) = \bigotimes_{\mathsf{e} \in \mathcal{E}_{\mathtt{G}}} \mathcal{S}(\mathtt{G}|_{-\prec^*\mathsf{e}})$$

A past-coherent specification $\mathcal{S}$ is such that the arbitrations for any configuration `G` (i.e., the set of paths $\mathcal{S}(\mathtt{G})$) can be obtained by composing the arbitrations associated with all its sub-configurations $\mathtt{G}|_{-\prec^*\mathsf{e}}$.

Example 9. The specifications in Example 4, Example 5 and Example 6 are all coherent, because their definitions are in terms of restrictions of the LDAGs. This can be checked by application of Definition 6. Consider e.g. the specification of the data type `Counter` in Example 4. Hence, if $\mathtt{P} \in \mathcal{S}_{Ctr}(\mathtt{G})$ then

$$\forall \mathsf{e} \in \mathcal{E}_{\mathtt{G}}.\forall \mathtt{k}.\lambda(\mathsf{e}) = \langle \mathtt{rd}, \mathtt{k}\rangle \text{ implies } \mathtt{k} = \#\{\mathsf{e}' \mid \mathsf{e}' \prec_{\mathtt{G}} \mathsf{e} \text{ and } \lambda(\mathsf{e}') = \langle \mathtt{inc}, \mathtt{ok}\rangle\}$$

Let $\bar{\mathsf{e}} \in \mathcal{E}_{\mathtt{G}}$ and $\mathtt{G}_{\bar{\mathsf{e}}} = \mathtt{G}|_{-\prec^*\bar{\mathsf{e}}}$. Clearly the property above holds $\forall \mathsf{e} \in \mathcal{E}_{\mathtt{G}_{\bar{\mathsf{e}}}}$. Thus $\mathtt{P}|_{-\prec^*\bar{\mathsf{e}}} \in \mathcal{S}_{Ctr}(\mathtt{G}_{\bar{\mathsf{e}}}) = \mathcal{S}_{Ctr}(\mathtt{G}|_{-\prec^*\bar{\mathsf{e}}})$ and consequently $\mathtt{P} \in \bigotimes_{\bar{\mathsf{e}} \in \mathcal{E}_{\mathtt{G}}} \mathcal{S}_{Ctr}(\mathtt{G}|_{-\prec^*\bar{\mathsf{e}}})$.

Conversely, let $\mathtt{P} \in \bigotimes_{\mathsf{e} \in \mathcal{E}_{\mathtt{G}}} \mathcal{S}_{Ctr}(\mathtt{G}|_{-\prec^*\mathsf{e}})$ and $\bar{\mathsf{e}} \in \mathcal{E}_{\mathtt{G}}$. Then $\mathtt{P}|_{-\prec^*\bar{\mathsf{e}}} \in \mathcal{S}_{Ctr}(\mathtt{G}_{\bar{\mathsf{e}}})$, hence if $\lambda(\bar{\mathsf{e}}) = \langle \mathtt{rd}, \mathtt{k}\rangle$ then $\mathtt{k} = \#\{\mathsf{e}' \mid \mathsf{e}' \prec_{\mathtt{G}_{\bar{\mathsf{e}}}} \mathsf{e} \text{ and } \lambda(\mathsf{e}') = \langle \mathtt{inc}, \mathtt{ok}\rangle\}$. By definition of $\mathtt{G}_{\bar{\mathsf{e}}}$, $\mathtt{k} = \#\{\mathsf{e}' \mid \mathsf{e}' \prec_{\mathtt{G}} \mathsf{e} \text{ and } \lambda(\mathsf{e}') = \langle \mathtt{inc}, \mathtt{ok}\rangle\}$ and consequently $\mathtt{P} \in \mathcal{S}_{Ctr}(\mathtt{G})$.

Now consider the specification $\mathcal{S}$ defined such that the equalities in Fig. 6 hold. $\mathcal{S}$ is not coherent because the arbitrations for the LDAG in Fig. 6b should contain all the interleavings for the paths associated with its sub-configurations, as depicted in Fig. 6a.

$$\mathcal{S}(\ \langle \mathtt{rd}, 0\rangle\ ) = \{\ \langle \mathtt{rd}, 0\rangle\ \}$$
$$\mathcal{S}(\ \langle \mathtt{inc}, \mathtt{ok}\rangle\ ) = \{\ \langle \mathtt{inc}, \mathtt{ok}\rangle\ \}$$

(a)

$$\mathcal{S}\left(\ \begin{array}{c}\langle \mathtt{rd}, 0\rangle \\ \\ \langle \mathtt{inc}, \mathtt{ok}\rangle\end{array}\ \right) = \left\{\ \begin{array}{c}\langle \mathtt{rd}, 0\rangle \\ | \\ \langle \mathtt{inc}, \mathtt{ok}\rangle\end{array}\ \right\}$$

(b)

Fig. 6. A non-coherent specification.

$$\mathcal{S}(\ \langle \mathtt{inc}, \mathtt{ok}\rangle\ ) = \{\ \langle \mathtt{inc}, \mathtt{ok}\rangle\ \}$$
$$\mathcal{S}(\ \langle \mathtt{rd}, 0\rangle\ ) = \{\ \langle \mathtt{rd}, 0\rangle\ \}$$

$$\mathcal{S}\left(\ \begin{array}{c}\langle \mathtt{rd}, 0\rangle \\ \downarrow \\ \langle \mathtt{inc}, \mathtt{ok}\rangle\end{array}\ \right) = \left\{\ \begin{array}{c}\langle \mathtt{rd}, 0\rangle \\ | \\ \langle \mathtt{inc}, \mathtt{ok}\rangle\end{array}\ \right\}$$

Fig. 7. A non-saturated specification.

A second class of specifications is concerned with saturation. Intuitively, a saturated specification allows every top element on the visibility to be arbitrated in any position. We first introduce the notion of saturation for a path.

Definition 7 *(Path saturation).* Let $\mathtt{P}$ be a path and ℓ a label. We write $\mathtt{sat}(\mathtt{P}, \ell)$ for the set of paths obtained by saturating $\mathtt{P}$ with respect to ℓ, defined as follows

$$\mathtt{sat}(\mathtt{P}, \ell) = \{\mathtt{Q} \mid \mathtt{Q} \in \mathbb{P}(\mathcal{E}_{\mathtt{P}^\ell}, \lambda_{\mathtt{P}^\ell}) \text{ and } \mathtt{Q}|_{\mathcal{E}_\mathtt{P}} = \mathtt{P}\}$$

A path $\mathtt{P}$ that is saturated with a label ℓ generates the set of all paths obtained by placing a new event labelled by ℓ in any position within $\mathtt{P}$. By analogy, a saturated specification thus extends a computation by adding a new operation that can be arbitrated in any position.

Definition 8 *(Saturated specification).* Let $\mathcal{S}$ be a specification. We say that $\mathcal{S}$ is saturated if

$$\forall \langle \mathtt{G}, \mathtt{P}\rangle, \mathcal{E}, \ell.\ \mathtt{P} \in \mathcal{S}(\mathtt{G}^\ell_\mathcal{E})\Big|_{\mathcal{E}_\mathtt{G}} \text{ implies } \mathtt{sat}(\mathtt{P}, \ell) \subseteq \mathcal{S}(\mathtt{G}^\ell_\mathcal{E})$$

Example 10. The specifications in Example 4, Example 5 and Example 6 are all saturated because a new event $\mathtt{e}$ can be arbitrated in any position. In fact, the specifications in Example 4 and Example 6 do not use any information about arbitration, while the specification in Example 5 constrains arbitrations only for events that are not maximal. Fig. 7 shows a specification that is not saturated because it does not allow the arbitration of the top event (the one labelled $\langle \mathtt{inc}, \mathtt{ok}\rangle$) as the first operation in a path. We remark that the specification is coherent although it is not saturated.

4. Replicated data type

In this section we show that our notion of specification can be considered as (and it is actually more general than) a model for the operational description of RDTs proposed in [3,4]. We start by recasting the original definition of RDT (as given in [4, Def. 4.5]) in terms of LDAGs. As hinted in the introduction, the meaning of each operation of an RDT is specified in terms of a context, written $\mathtt{C}$, which is a pair $\langle \mathtt{G}, \mathtt{P}\rangle$ such that $\mathtt{P} \in \mathbb{P}(\mathcal{E}_\mathtt{G}, \lambda_\mathtt{G})$. We write $\mathbb{C}(\mathcal{L})$ for the set of contexts over $\mathcal{L}$, and fix a set $\mathcal{O}$ of operations and a set $\mathcal{V}$ of values. Then, the operational description of RDTs in [3,4] can be formulated as follows.

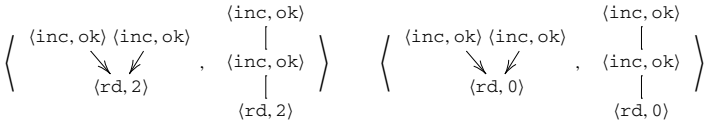
Definition 9 *(Replicated data type).* A *Replicated Data Type* (RDT) is a function $\mathcal{F} : \mathcal{O} \times \mathbb{C}(\mathcal{O}) \to \mathcal{V}$.

In words, for any visibility graph $\mathtt{G}$ and arbitration $\mathtt{P}$, the specification $\mathcal{F}$ indicates the result of executing the operation $\mathtt{op}$ over $\mathtt{G}$ and $\mathtt{P}$, which is $\mathcal{F}(\mathtt{op}, \langle \mathtt{G}, \mathtt{P}\rangle)$.

Example 11. The data type `Counter` introduced in Example 4 is formally specified in [3,4] as follows

$$\mathcal{F}_{ctr}(\mathtt{inc}, \langle \mathtt{G}, \mathtt{P}\rangle) = \mathtt{ok}$$
$$\mathcal{F}_{ctr}(\mathtt{rd}, \langle \mathtt{G}, \mathtt{P}\rangle) = \#\{\mathtt{e} \mid \mathtt{e} \in \mathtt{G} \text{ and } \lambda(\mathtt{e}) = \mathtt{inc}\}$$

Given a context $\langle \mathtt{G}, \mathtt{P}\rangle$ in $\mathbb{C}(\mathcal{O} \times \mathcal{V})$, we may check whether the value associated with each operation matches the definition of a particular RDT. This notion is known as *return value consistency* [4, Def. 4.8]. In order to relate contexts with and without return values, we use the following notation: given $\mathtt{G} \in \mathbb{G}(\mathcal{O} \times \mathcal{V})$, by $\overline{\mathtt{G}} \in \mathbb{G}(\mathcal{O})$ we denote the LDAG obtained by projecting the labels of $\mathtt{G}$ in the obvious way, i.e., by removing the second component of every label.



(a) Consistent. (b) Non consistent.

Fig. 8. RVAL consistency for $\mathcal{F}_{ctr}$.

Definition 10 *(Return value consistent).* Let $\mathcal{F}$ be an RDT and $\langle \mathtt{G}, \mathtt{P} \rangle \in \mathbb{C}(\mathcal{O} \times \mathcal{V})$ a context. We say that $\mathcal{F}$ is *Return Value Consistent* (RVAL) over $\mathtt{G}$ and $\mathtt{P}$ and write $\textsc{rval}(\mathcal{F}, \mathtt{G}, \mathtt{P})$ if $\forall \mathsf{e} \in \mathcal{E}_{\mathtt{G}}.\lambda(\mathsf{e}) = \langle \mathtt{op}, \mathtt{v} \rangle$ implies $\mathcal{F}(\mathtt{op}, \overline{\mathtt{G}}\big|_{-\prec \mathsf{e}}, \overline{\mathtt{P}}\big|_{-\prec \mathsf{e}}) = \mathtt{v}$.

Moreover, we define

$$\textsc{prval}(\mathcal{F}, \mathtt{G}) = \{\mathtt{P} \mid \textsc{rval}(\mathcal{F}, \mathtt{G}, \mathtt{P})\}$$

Example 12. Consider the RDT $\mathcal{F}_{ctr}$ introduced in Example 11. The context in Fig. 8a is RVAL consistent while the one in Fig. 8b is not because $\mathcal{F}_{ctr}$ requires `rd` to return the number of `inc` operations seen by that read, which in this case should be `2`.

The following result states that return value consistent paths are all coherent, in the sense that they match the behaviour allowed for any shorter configuration.

Lemma 1. *Let $\mathcal{F}$ be an* RDT *and* $\mathtt{G}$ *an* LDAG*. Then*

$$\textsc{prval}(\mathcal{F}, \mathtt{G}) = \bigotimes_{\mathsf{e} \in \mathcal{E}_{\mathtt{G}}} \textsc{prval}(\mathcal{F}, \mathtt{G}|_{-\prec^* \mathsf{e}}).$$

4.1. Functional specifications

We now focus on the relation between our notion of specification, as introduced in Definition 4, and the operational description of RDTs, as introduced in [3,4] and formalised in Definition 9 in terms of LDAGs. Specifically, we characterise a proper subclass of specifications that precisely correspond to RDTs.

For this section we restrict our attention to specifications over the set of labels $\mathcal{O} \times \mathcal{V}$, i.e., $\mathcal{S} : \mathbb{G}(\mathcal{O} \times \mathcal{V}) \to 2^{\mathbb{P}(\mathcal{O} \times \mathcal{V})}$.

Definition 11 *(Total specification).* Let $\mathcal{S}$ be a specification. We say that $\mathcal{S}$ is total if

$$\forall \langle \mathtt{G}, \mathtt{P} \rangle, \mathcal{E}, \mathtt{op}.\ \exists \mathtt{G}_1,\ \mathtt{v}.\ \overline{\mathtt{G}} = \overline{\mathtt{G}_1} \ \wedge\ \overline{\mathtt{P}} \in \overline{\mathcal{S}((\mathtt{G}_1)^{\langle \mathtt{op}, \mathtt{v} \rangle}_{\mathcal{E}})}\Big|_{\mathcal{E}_{\mathtt{G}_1}}$$

Intuitively, a specification is total when every operation of the data type can be performed in any state of the computation. Formally, this is stated by considering a context $\langle \overline{\mathtt{G}}, \overline{\mathtt{P}} \rangle$ as the representation of the state of a computation. We remark that differently from $\mathtt{G}$ and $\mathtt{P}$, whose labels are in $\mathcal{O} \times \mathcal{V}$, $\overline{\mathtt{G}}$ and $\overline{\mathtt{P}}$ have no information about the return value of the operations (i.e., their labels are in $\mathcal{O}$). Hence, totality says that it is always possible to take an equivalent representation of the state (i.e., $\langle \mathtt{G}_1, \mathtt{P}_1 \rangle$ instead of $\langle \mathtt{G}, \mathtt{P} \rangle$) and extend it with an operation `op`. This is achieved by requiring $\overline{\mathtt{P}} \in \overline{\mathcal{S}((\mathtt{G}_1)^{\langle \mathtt{op}, \mathtt{v} \rangle}_{\mathcal{E}})}\Big|_{\mathcal{E}_{\mathtt{G}_1}}$ for some return value $\mathtt{v}$. Thus $\mathcal{S}((\mathtt{G}_1)^{\langle \mathtt{op}, \mathtt{v} \rangle}_{\mathcal{E}})$ is not empty, hence, the specification allows the execution of `op` over $\mathtt{G}_1$.

We remark that a total specification does not prevent the definition of an operation that admits more than one return value in certain configurations, i.e., $\mathtt{v}$ in Definition 11 does not need to be unique. For instance, consider the `Generic Register` in Example 6, where operation `rd` may return any of the causally-independent, previously written values. Albeit being total, the specification for `rd` is not deterministic. On the contrary, a specification is deterministic if an operation executed over a configuration admits at most one return value, as formally stated below.

Definition 12 *(Deterministic specification).* Let $\mathcal{S}$ be a specification. We say that $\mathcal{S}$ is deterministic if

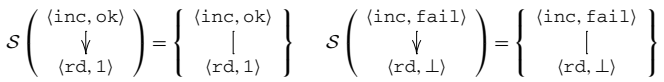
$$\forall \mathtt{G}, \mathcal{E}, \mathtt{op}, \mathtt{v}, \mathtt{v}'.\ \mathtt{v} \neq \mathtt{v}' \text{ implies } \overline{\mathcal{S}(\mathtt{G}^{\langle \mathtt{op}, \mathtt{v} \rangle}_{\mathcal{E}})}\Big|_{\mathcal{E}_{\mathtt{G}}} \cap \overline{\mathcal{S}(\mathtt{G}^{\langle \mathtt{op}, \mathtt{v}' \rangle}_{\mathcal{E}})}\Big|_{\mathcal{E}_{\mathtt{G}}} = \emptyset$$

We say that $\mathcal{S}$ is weakly deterministic if the property holds for $\mathcal{E} = \emptyset$.

A less restrictive notion for determinism could allow the result for an added operation to depend also on the given admissible path. We say that a specification $\mathcal{S}$ is *value-deterministic* if

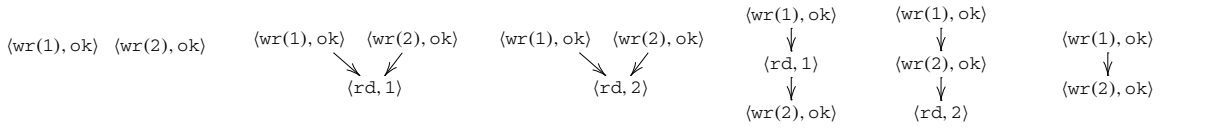
$$\forall \mathtt{G}, \mathcal{E}, \mathtt{op}, \mathtt{v}, \mathtt{v}'.\ \mathtt{v} \neq \mathtt{v}' \wedge\ \mathtt{G} \neq \varepsilon \text{ implies } \mathcal{S}(\mathtt{G}^{\langle \mathtt{op}, \mathtt{v} \rangle}_{\mathcal{E}})\Big|_{\mathcal{E}_{\mathtt{G}}} \cap \mathcal{S}(\mathtt{G}^{\langle \mathtt{op}, \mathtt{v}' \rangle}_{\mathcal{E}})\Big|_{\mathcal{E}_{\mathtt{G}}} = \emptyset$$

We say that a specification is *functional* if it is both deterministic and total.



(a) (b)

Fig. 9. A value-deterministic and coherent specification.



(a) G. (b) $G^{\langle rd,1\rangle}$. (c) $G^{\langle rd,2\rangle}$. (d) P_1. (e) P_2. (f) $P = P_1|_{\mathcal{T}_G} = P_2|_{\mathcal{T}_G}$.

Fig. 10. Generic register.

Example 13. Fig. 9 shows a value-deterministic specification. Although a read operation that follows an increment may return two different values, such a difference is explained by the previous computation: in one case the increment succeeds while in the other fails. However, the specification is not deterministic because it admits a sequence of operations to be decorated with different return values.

Example 14. It is straightforward to check that the specifications in Example 4 and Example 5 are deterministic. For Example 4 we reason as follows. The case for $\mathtt{op} = \mathtt{inc}$ follows immediately because the only possible return value is $\mathtt{ok}$. When $\mathtt{op} = \mathtt{rd}$, from the definition of $\mathcal{S}_{Ctr}$ (Example 4) we conclude that $\mathcal{S}_{Ctr}(G_{\mathcal{E}}^{\langle \mathtt{rd},\mathtt{v}\rangle}) \neq \emptyset$ only when $\mathtt{v} = \#\{\mathsf{e} \mid \mathsf{e} \in \mathcal{E} \text{ and } \lambda(\mathsf{e}) = \langle \mathtt{inc}, \mathtt{ok}\rangle\}$. Consequently, for any $\mathtt{v}' \neq \mathtt{v}$, $\mathcal{S}_{Ctr}(G_{\mathcal{E}}^{\langle \mathtt{rd},\mathtt{v}'\rangle}) = \emptyset$ holds, hence, $\mathcal{S}_{Ctr}$ is deterministic. For Example 5, we can reason analogously.

On the contrary, the specification of the `Generic Register` in Example 6 is not even value-deterministic. It suffices to consider a configuration G with two different written values, as shown in Fig. 10a. Consider now the two extensions $G^{\langle \mathtt{rd},1\rangle}$ and $G^{\langle \mathtt{rd},2\rangle}$ depicted in Fig. 10b and Fig. 10c and the two paths P_1 and P_2 in Fig. 10d and Fig. 10e. By the definition of $\mathcal{S}_{gR}$, we can conclude that $P_1 \in \mathcal{S}_{gR}(G^{\langle \mathtt{rd},1\rangle})$ and $P_2 \in \mathcal{S}_{gR}(G^{\langle \mathtt{rd},2\rangle})$. The path P in Fig. 10f corresponds to both $P_1|_{\mathcal{T}_G}$ and $P_2|_{\mathcal{T}_G}$. Consequently, $\mathcal{S}_{gR}(G^{\langle \mathtt{rd},1\rangle})\big|_{\mathcal{T}_G} \cap \mathcal{S}_{gR}(G^{\langle \mathtt{rd},2\rangle})\big|_{\mathcal{T}_G} \neq \emptyset$.

Similarly, `Set` in Example 8 is not deterministic.

The lemma below states a simple criterion for determinism.

Lemma 2. *Let $\mathcal{S}$ be a coherent and deterministic specification. Then*

$$\forall G_1, G_2.\ \overline{G_1} = \overline{G_2} \textit{ implies } G_1 = G_2 \vee \overline{\mathcal{S}(G_1)} \cap \overline{\mathcal{S}(G_2)} = \emptyset$$

So, if two configurations are annotated with the same operations yet with different values, then their admissible paths are already all different even if we disregard return values.

We then consider a last property that guarantees some sort of additional locality to the notion of coherence.

Definition 13 *(Local specification).* Let $\mathcal{S}$ be a specification. We say that $\mathcal{S}$ is local if

$$\forall G, \mathcal{E}.\ \exists G_1.\ \forall \mathtt{op}, \mathtt{v}.\ \ \overline{G}\big|_{\mathcal{E}} = \overline{G_1} \ \wedge\ \overline{\mathcal{S}(G_{\mathcal{E}}^{\langle \mathtt{op},\mathtt{v}\rangle})}\Big|_{-\preccurlyeq \top} \subseteq \overline{\mathcal{S}(G_1{}^{\langle \mathtt{op},\mathtt{v}\rangle})}$$

We say that a specification is *locally functional* if it is both functional and local.

Intuitively, locality states that the admissible paths of an extended configuration are constrained: given a configuration G, its extension with an operation $\mathtt{op}$ with respect to the events $\mathcal{E}$ can be explained by the sub-configuration of G that only contains those events, i.e., $\overline{G}\big|_{\mathcal{E}} = \overline{G_1}$. Such behaviour is given by $\mathcal{S}(G_1{}^{\langle \mathtt{op},\mathtt{v}\rangle})$.

As a side remark, it is noteworthy that locality has an impact on determinism.

Lemma 3. *Let $\mathcal{S}$ be a local and weakly deterministic specification. Then it is deterministic.*

4.2. Correspondence between RDTs *and specifications*

This section establishes the connection between RDTs and specifications. We first introduce a mapping from RDTs to specifications.

Definition 14 *(RDTs as specifications).* Let $\mathcal{F}$ be an RDT. We write $\mathbb{S}(\mathcal{F})$ for the specification associated with $\mathcal{F}$, defined as follows

$$\mathbb{S}(\mathcal{F})(\mathtt{G}) = \textsc{prval}(\mathcal{F}, \mathtt{G})$$

Next result shows that RDTs correspond to specifications that are coherent, functional and saturated.

Lemma 4. *For every RDT $\mathcal{F}$, $\mathbb{S}(\mathcal{F})$ is coherent, locally functional, and saturated.*

The inverse mapping from specifications to RDTs is defined below.

Definition 15 *(Specifications as RDTs).* Let $\mathcal{S}$ be a specification. We write $\mathbb{F}(\mathcal{S})$ for the RDT associated with $\mathcal{S}$, defined as follows

$$\mathbb{F}(\mathcal{S})(\mathtt{op}, \overline{\mathtt{G}}, \overline{\mathtt{P}}) = \mathtt{v} \text{ if } \exists \mathtt{G}_1 . \ \overline{\mathtt{G}} = \overline{\mathtt{G}_1} \ \wedge \ \overline{\mathtt{P}} \in \overline{\mathcal{S}(\mathtt{G}_1{}^{\langle \mathtt{op}, \mathtt{v} \rangle})}\Big|_{\mathcal{E}_{\mathtt{G}_1}}$$

$\mathbb{F}(\mathcal{S})$ may not be well-defined for some $\mathcal{S}$, e.g. when $\mathcal{S}$ is not deterministic. The following lemma states the conditions under which $\mathbb{F}(\mathcal{S})$ is well-defined.

Lemma 5. *For every coherent, functional, and saturated specification $\mathcal{S}$, $\mathbb{F}(\mathcal{S})$ is well-defined.*

The following two results show that RDTs are a particular class of specifications, and hence, provide a fully abstract characterisation of operational RDTs.

Theorem 1. *For every coherent, locally functional, and saturated specification $\mathcal{S}$, $\mathcal{S} = \mathbb{S}(\mathbb{F}(\mathcal{S}))$.*

Theorem 2. *For every RDT $\mathcal{F}$, $\mathcal{F} = \mathbb{F}(\mathbb{S}(\mathcal{F}))$.*

The above characterisation implies that there are data types that cannot be specified as operational RDTs. Consider e.g. `Generic Register` and `Set`, as introduced respectively in Example 6 and Example 8. As noted in Example 14, they are not deterministic. Hence, they cannot be translated as RDTs. We remark that a non-deterministic specification does not imply a non-deterministic conflict resolution, but it allows for under-specification.

5. On the correct implementation of replicated data types

The previous section was devoted to the proof of correspondence between the novel notion of specification we introduced and a more standard proposal for modelling replicated data types. In this section we argue that our definition is flexible enough for reasoning about the correctness of possible implementations, in terms of the classical notion of *simulation*. More precisely, first we show how a specification naturally gives rise to a labelled transition system (LTS). Then, we consider state-based implementations of replicated data types [3].

5.1. From specifications to labelled transitions systems

We note that specifications have an implicit operational interpretation.

Definition 16 *(One-replica LTS).* Let $\mathcal{S}$ be a specification. Then, the abstract *one-replica* LTS $\mathcal{T}_{\mathcal{S}}$ has pairs $\langle \mathtt{G}, \mathtt{P} \rangle$ as states, pairs $\langle \mathtt{op}, \mathtt{v} \rangle$ as labels, and triples $\langle \mathtt{G}, \mathtt{P} \rangle \xrightarrow{\ell} \langle \mathtt{G}', \mathtt{P}' \rangle$ as transitions such that

- $\langle \mathtt{G}, \mathtt{P} \rangle$ is a state whenever $\mathtt{P} \in \mathcal{S}(\mathtt{G})$;
- $\langle \mathtt{G}, \mathtt{P} \rangle \xrightarrow{\ell} \langle \mathtt{G}', \mathtt{P}' \rangle$ is a transition whenever $\mathtt{G}' = \mathtt{G}^{\ell}$ and $\mathtt{P}'\big|_{\mathcal{E}_{\mathtt{G}}} = \mathtt{P}$.

A pair $\langle \mathtt{G}, \mathtt{P} \rangle$ such that $\mathtt{P} \in \mathcal{S}(\mathtt{G})$ abstractly represents an admissible computation according to $\mathcal{S}$. Moreover, $\mathcal{S}$ allows the extension of such computation with an event labelled by ℓ whenever the specification allows us to extend $\mathtt{G}$ and $\mathtt{P}$ with a fresh event labelled by ℓ.

Our next step is to show that a specification can be equipped with a notion of parallel composition. But first, we need to consider a suitable kind of morphism.

Definition 17 *(Downward closure).* Let $\mathtt{G} = \langle \mathcal{E}, \prec, \lambda \rangle$ be an LDAG and $\mathcal{E}' \subseteq \mathcal{E}$. We say that $\mathcal{E}'$ is downward closed if

$$\forall \mathsf{e} \in \mathcal{E}' . - \prec \mathsf{e} \subseteq \mathcal{E}'.$$

Definition 18 *(Past-reflecting morphisms).* An LDAG morphism $\mathtt{f} : \mathtt{G}_1 \to \mathtt{G}_2$ from $\mathtt{G}_1$ to $\mathtt{G}_2$ is a function $\mathtt{f} : \mathcal{E}_{\mathtt{G}_1} \to \mathcal{E}_{\mathtt{G}_2}$ such that $\lambda_{\mathtt{G}_1} = \mathtt{f}; \lambda_{\mathtt{G}_2}$ and $\mathsf{e} \prec_{\mathtt{G}_1} \mathsf{e}'$ implies $\mathtt{f}(\mathsf{e}) \prec_{\mathtt{G}'} \mathtt{f}(\mathsf{e}')$. An injective LDAG morphism $\mathtt{f} : \mathtt{G}_1 \to \mathtt{G}_2$ is past-reflecting if

- $\mathtt{f}(\mathsf{e}) \prec_{\mathtt{G}_2} \mathtt{f}(\mathsf{e}')$ implies $\mathsf{e} \prec_{\mathtt{G}_1} \mathsf{e}'$;
- $\bigcup_{\mathsf{e} \in \mathcal{E}_{\mathtt{G}_1}} \mathtt{f}(\mathsf{e})$ is downward closed.

Hence, past-reflecting injective morphisms $\mathtt{f} : \mathtt{G}_1 \to \mathtt{G}_2$ are uniquely characterised as such by the properties of the image of $\mathcal{E}_{\mathtt{G}_1}$ with respect to $\mathtt{G}_2$.

Definition 19 *(Compatibility).* Let $\mathtt{G}_1$, $\mathtt{G}_2$ be two LDAGs. They are compatible if

- $\mathsf{e} \in \mathcal{E}_{\mathtt{G}_1} \cap \mathcal{E}_{\mathtt{G}_2}$ implies $\lambda_1(\mathsf{e}) = \lambda_2(\mathsf{e})$;
- the injective morphisms $\mathtt{G}_i \to \mathtt{G}_1 \cup \mathtt{G}_2$ for $i \in \{1, 2\}$ are past-reflecting.

We write $\mathtt{G}_1 \sqcup \mathtt{G}_2$ to denote the union of compatible LDAGs.

Note that the operation $\sqcup$ over LDAGs is idempotent, associative, and commutative. Also note that if $\mathtt{G}_1$ and $\mathtt{G}_2$ are compatible, then the product $\mathtt{P}_1 \otimes \mathtt{P}_2$ is correctly labelled for any pair $\mathtt{P}_1 \in \mathcal{S}(\mathtt{G}_1)$ and $\mathtt{P}_2 \in \mathcal{S}(\mathtt{G}_2)$ (i.e., events that belongs to both $\mathcal{E}_{\mathtt{G}_1}$ and $\mathcal{E}_{\mathtt{G}_2}$ have the same label) because each path is built out of the set of events (and the corresponding labels) of the associated LDAG. Consequently, we write $\langle \mathtt{G}_1 \sqcup \ldots \sqcup \mathtt{G}_n, \mathtt{P} \rangle$ for a replicated state consisting of n compatible replicas, with $\langle \mathtt{G}_i, \mathtt{P}|_{\mathcal{E}_{\mathtt{G}_i}} \rangle$ denoting the state of the replica i.

Definition 20 *(Multi-replica LTS).* Let $\mathcal{S}$ be a coherent specification. Then, the abstract multi-replica LTS $\mathcal{T}_{\mathcal{S}}$ is generated by the corresponding one-replica LTS and the additional rule below.

(COMP)

$$\frac{\langle \mathtt{G}_1, \mathtt{P}|_{\mathcal{E}_{\mathtt{G}_1}} \rangle \xrightarrow{\ell} \langle \mathtt{G}'_1, \mathtt{P}'_1 \rangle \qquad \mathtt{P}' \in \mathtt{P} \otimes \mathtt{P}'_1}{\langle \mathtt{G}_1 \sqcup \mathtt{G}_2, \mathtt{P} \rangle \xrightarrow{\ell} \langle \mathtt{G}'_1 \sqcup \mathtt{G}_2, \mathtt{P}' \rangle}$$

Rule (COMP) describes the computations associated to a replicated state. Whenever a part of a replicated state (denoted by $\langle \mathtt{G}_1, \mathtt{P}|_{\mathcal{E}_{\mathtt{G}_1}} \rangle$) evolves to $\langle \mathtt{G}'_1, \mathtt{P}'_1 \rangle$ by performing the action ℓ, then the whole system evolves to a state obtained by composing the part that has not changed (i.e., $\mathtt{G}_2$) and the new state $\langle \mathtt{G}'_1, \mathtt{P}'_1 \rangle$.

Indeed, the states corresponding to the union of compatible LDAGs recalls parallel composition, and the soundness of the rule with respect to the target is witnessed by the proposition below, ensuring that the transitions of a composite state are in correspondence with the transitions of its components.

Lemma 6. *Let $\mathcal{S}$ be a coherent specification, $\mathtt{G}_1$, $\mathtt{G}_2$ compatible* LDAGs, *and $\mathtt{P} \in \mathcal{S}(\mathtt{G}_1 \sqcup \mathtt{G}_2)$ a path. If $\langle \mathtt{G}_1, \mathtt{P}|_{\mathcal{E}_{\mathtt{G}_1}} \rangle \xrightarrow{\ell} \langle \mathtt{G}'_1, \mathtt{P}' \rangle$ then $\mathtt{P} \otimes \mathtt{P}' \subseteq \mathcal{S}(\mathtt{G}'_1 \sqcup \mathtt{G}_2)$.*

5.2. Implementing a specification

In this section we address the problem of identifying what a concrete implementation of a replicated data type is. We consider a setting in which a replicated data type is realised on top of a set of replicas, where each replica keeps a local version of the data and clients perform read and write operations over replicas. Roughly speaking, each request made by a client is handled by one of the replicas. In particular, a read request is answered according to the local state of the replica that handles that request. Analogously, any update operation is executed over the local state of a replica. The changes are then propagated asynchronously to the remaining replicas, which will update their own states later on. Since changes are propagated asynchronously and different replicas may be serving concurrent updates, each replica is responsible for resolving conflicts locally. We will focus on state-based implementation of replicated data types, as opposed to operation-based implementation [2]. In a state-based implementation, replicas propagate changes by communicating their own states.

Following the approach in [3], we describe an implementation of a data type in terms of the behaviour of a replica and we assume that all the replicas of an implementation behave the same. We will also consider LTSs as the operational model of replicas. We then let Σ (ranged over by $\sigma, \sigma_0, \ldots$) denote the set of possible states of a replica and for a specification $\mathcal{S}$ we use the following set of operations as labels

$$\mathcal{A}_{\mathcal{S}} = \mathcal{L} \cup (\{\mathtt{send}, \mathtt{rcv}\} \times \Sigma) \cup \{\tau\}$$

$$\text{(READ)}\quad \frac{k = \Sigma_{s \in \mathcal{R}} v(s)}{\langle r, v\rangle \xrightarrow{rd,k} \langle r, v\rangle} \qquad \text{(INC)}\quad \langle r, v\rangle \xrightarrow{inc,ok} \langle r, v[r \mapsto v(r)+1]\rangle$$

$$\text{(SEND)}\quad \langle r, v\rangle \xrightarrow{send\langle r,v\rangle} \langle r, v\rangle \qquad \text{(RCV)}\quad \langle r, v\rangle \xrightarrow{rcv\langle r',v'\rangle} \langle r, \max\{v, v'\}\rangle$$

Fig. 11. Implementation of data type Counter.

The set $\mathcal{A}_{\mathcal{S}}$ of labels is then built-up from the operations of the data type (i.e., the elements in $\mathcal{L} = \mathcal{O} \times \mathcal{V}$), with additionally two special kinds of operations $\langle \mathtt{send}, \sigma\rangle$ and $\langle \mathtt{rcv}, \sigma\rangle$ that are used by the replicas to synchronise their states and finally the label τ for internal transitions.

The behaviour of a replica of a specification $\mathcal{S}$ is given by an LTS $\mathcal{C}_{\mathcal{S}}$ with labels in $\mathcal{A}_{\mathcal{S}}$ that is *total*, i.e., such that $\forall \sigma, \mathtt{op}.\ \exists v.\ \sigma \xrightarrow{op,v}$. Totality ensures that all operations in the implementation of a replicated data type are non-blocking, i.e., a replica is able to perform any operation of the data type at any state.

A complete implementation of a replicated data type is then obtained by putting several replicas in parallel. As a formal counterpart, we rely on a composition operator that prescribes the way in which different replicas synchronise. Its definition depends on the chosen communication model (synchronous, asynchronous, broadcast, ...) and we defer its formal treatment to § 5.4.

Example 15. We consider the data type `Counter` in Example 4 and the state-based implementation presented in [3], which is defined below

- $\Sigma = \mathcal{R} \times (\mathcal{R} \mapsto \mathbb{N})$, where $\mathcal{R}$ is the set of replicas' identifiers;
- $\mathcal{L} = \{\langle \mathtt{inc}, \mathtt{ok}\rangle\} \cup (\{\mathtt{rd}\} \times \mathbb{N})$ is the set of data type operations;
- $\rightarrow$ is given by the inference rules in Fig. 11.

The states of a replica are pairs of the form $\langle r, v\rangle$, where r is the replica's id and v is a mapping that keeps track of the known increments performed over all replicas, i.e., $v(r')$ is the number of increments performed over the replica r'.

We now comment on the inference rules in Fig. 11. Rule (READ) describes a replica that is handling a client's request for reading the counter. In such case, the replica returns the value k, which corresponds to the total number of increments known from all replicas. This transition does not change the state of the replica. Differently, the state changes when performing an increment, as described by rule (INC). The new state records the fact that r has performed another increment. The change has only local effect and can be propagated later on, by using rule (SEND). A replica also updates its local state when it receives a change propagated by another replica, as described by the rule (RCV). When receiving a message $\langle r', v'\rangle$, the replica updates its local mapping to $\max\{v, v'\}$, which is defined as follows

$$\forall s. \max\{v, v'\}(s) = \max\{v(s), v'(s)\}$$

The natural question is whether the behaviour of the replica in Example 15 correctly implements the specification in Example 4. We address this problem in two steps: first of all, we show that a single replica is a correct implementation of the data type (§ 5.3); then, we analyse the combined behaviour of several replicas (§ 5.4).

5.3. Linking a specification with the behaviour of a replica

In this section we provide a criterion to formally prove that the implementation of a replica is correct. More precisely, the correspondence between the behaviour of a replica and a specification is given as a (weak) simulation relation.

Definition 21 *(Implementation correctness).* Let $\mathcal{S}$ be a specification, $\mathcal{T}_{\mathcal{S}}$ the one-replica (multi-replica) abstract LTS, and $\mathcal{C}_{\mathcal{S}}$ an implementation. Then, an *implementation relation* $\mathcal{I}_{\mathcal{S}}$ is a relation between the states of $\mathcal{C}_{\mathcal{S}}$ and $\mathcal{T}_{\mathcal{S}}$ such that if $(\sigma, \langle \mathtt{G}, \mathtt{P}\rangle) \in \mathcal{I}_{\mathcal{S}}$ then for any σ', σ'', op and v

1. if $\sigma \xrightarrow{op,v} \sigma'$ then $\exists \mathtt{G}', \mathtt{P}'$ such that $\langle \mathtt{G}, \mathtt{P}\rangle \xrightarrow{op,v} \langle \mathtt{G}', \mathtt{P}'\rangle$ and $(\sigma', \langle \mathtt{G}', \mathtt{P}'\rangle) \in \mathcal{I}_{\mathcal{S}}$;
2. if $\sigma \xrightarrow{rcv,\sigma'} \sigma''$ then $\exists \mathtt{G}', \mathtt{P}', \mathtt{P}''$ such that $(\sigma', \langle \mathtt{G}', \mathtt{P}'\rangle) \in \mathcal{I}_{\mathcal{S}}$ and $\mathtt{P}'' \in \mathtt{P} \otimes \mathtt{P}'$ and $(\sigma'', \langle \mathtt{G} \sqcup \mathtt{G}', \mathtt{P}''\rangle) \in \mathcal{I}_{\mathcal{S}}$;
3. if $\sigma \xrightarrow{send,\sigma'} \sigma$ then $\exists \mathtt{G}', \mathtt{P}'$ such that $(\sigma', \langle \mathtt{G}', \mathtt{P}'\rangle) \in \mathcal{I}_{\mathcal{S}}$ and $\mathtt{G} \sqcup \mathtt{G}' = \mathtt{G}$ and $\mathtt{P} \otimes \mathtt{P}' = \{\mathtt{P}\}$;
4. if $\sigma \xrightarrow{\tau} \sigma'$ then $(\sigma', \langle \mathtt{G}, \mathtt{P}\rangle) \in \mathcal{I}_{\mathcal{S}}$.

We write $\sim^{o}_{\mathcal{S}}$ for the largest implementation relation with respect to the abstract one-replica LTS of the specification $\mathcal{S}$, and $\sim^{m}_{\mathcal{S}}$ for the multi-replica one.

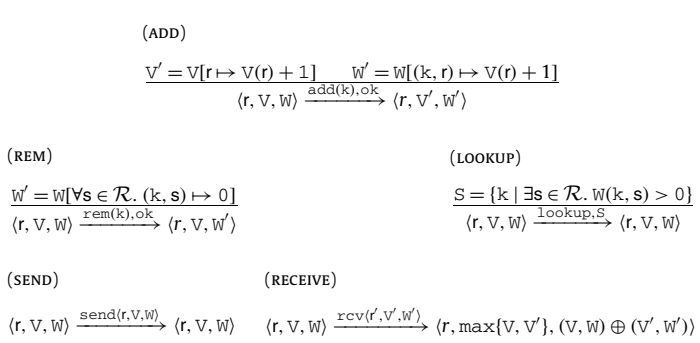


Fig. 12. Implementation of data type OR-SET

Any pair $(\sigma, \langle G, P\rangle)$ in the relation $\mathcal{I}_{\mathcal{S}}$ establishes that the state σ is explained in terms of the visibility G and the arbitration P, which is admitted by the specification (i.e., $P \in \mathcal{S}(G)$). Moreover, there is a close correspondence between the evolution of σ and $\langle G, P\rangle$, which is stated by the items 1–4. Item 1 predicates on the transitions corresponding to the operations of the data type. Basically, if the replica performs the operation op that produces the result v, then the specification allows G and P to be extended with the corresponding event, which is captured by the transition $\langle G, P\rangle \xrightarrow{op,v} \langle G', P'\rangle$ formalised in Definition 16 and Definition 20.

Item 2 regards those transitions in which the replica receives updates propagated by other replicas. In a state-based implementation, the content of a message rcv is a state computed by another replica. For this reason, we require the received state σ' to be related to a configuration admitted by the specification, i.e., $(\sigma', \langle G', P'\rangle) \in \mathcal{I}_{\mathcal{S}}$. Moreover, $\langle G', P'\rangle$ must be consistent with the history already seen by the replica, i.e., the common history in $\langle G, P\rangle$ and $\langle G', P'\rangle$ must coincide. This is established by requiring that the union of the two visibilities, i.e., $G \sqcup G'$, and the merge of the two paths, i.e., $P'' \in P \otimes P'$, are defined. Under the above conditions, the state σ'' obtained as the combination of σ and σ' must correspond to the combination of $\langle G, P\rangle$ and $\langle G', P'\rangle$, i.e., $(\sigma'', \langle G \sqcup G', P''\rangle) \in \mathcal{I}_{\mathcal{S}}$.

Item 3 states that a replica only propagates messages carrying information about its current state. Note that σ' may contain information about some events on the current state. This is formally stated by conditions $G \sqcup G' = G$ and $P \otimes P' = \{P\}$ (and it is going to be useful in § 5.4). Item 4 is self-explanatory.

In the following, we will refer to one-replica or multi-replica correctness of an implementation when the abstract LTS $\mathcal{T}_{\mathcal{S}}$ in the definition above is one-replica or multi-replica, respectively.

Example 16. We can show that the replica defined in Example 15 is a correct one-replica implementation of the data type $Counter$ in Example 4 by showing that the following relation satisfies the conditions stated in Definition 21 (proof details are in Appendix B) with respect to the one-replica abstract LTS

$$\mathcal{I} = \{(\langle r, v\rangle, \langle G, P\rangle) \mid \text{there exists } f : \mathcal{E}_G \to \mathcal{R} \text{ such that } \forall r \in \mathcal{R}. v(r) = \#\{e \mid f(e) = r \text{ and } \lambda(e) = \langle inc, ok\rangle\}\}$$

Example 17. In this example we show that the optimised $OR\text{-}set$ implementation in [3] is correct with respect to the specification in Example 8. The implementation of a replica is given by

- $\Sigma = \mathcal{R} \times (\mathcal{R} \mapsto \mathbb{N}) \times ((\mathcal{V} \times \mathcal{R}) \mapsto \mathbb{N})$;
- $\mathcal{L} = \{\langle add(k), ok\rangle, \langle rem(k), ok\rangle \mid k \in \mathcal{V}\} \cup (\{lookup\} \times 2^{\mathcal{V}})$;
- $\to$ is given by the inference rules in Fig. 12.

States are triples $\langle r, V, W\rangle$, where $r \in \mathcal{R}$ is the replica's id. The mapping V associates each replica with a version number and means that r is up-to-date with the version $V(r')$ of the replica r'. The mapping W indicates for each replica r' and element k the newest version of r' in which the element k has been added, if any (i.e., $W(k, r') = 0$ means that k either has not been added to r' or it has been added and then deleted). It is assumed that $W(k, r) \leq V(r)$ for all r and k.

We now discuss the rules in Fig. 12. Rule (ADD) describes the behaviour of a replica r that handles a request for adding the element k to the set. In this case, r changes its local state by (i) creating a new version, i.e., the entry $V(r)$ is updated to $V(r) + 1$ to reflect that there is a new version for r, and (ii) recording that the element k has been added in the newly created version of r, i.e., the entry $W(k, r)$ is updated to $V(r) + 1$. The removal of the element k is described by rule (REM). In this case, for all $s \in \mathcal{R}$, $W(k, s)$ is set to 0 to indicate the elimination of k. When performing a $lookup$, the replica r returns the set S of those elements that are present in at least one of the known versions of the replicas (Rule (LOOKUP)).

Rule (SEND) is straightforward. During synchronisation, which is described in rule (RECEIVE), the mappings V and W are updated with the information in the received message. It is assumed that the received mappings V' and W' are consistent, i.e., $W'(k, r_i) \leq V'(r_i)$ for all k. For V, the new state keeps the higher version for each replica, i.e., $\max\{V, V'\}$. The combination

$$\text{(parL)}\quad \frac{\sigma_1 \xrightarrow{\ell} \sigma_1'}{\sigma_1 \| \sigma_2 \xrightarrow{\ell} \sigma_1' \| \sigma_2} \qquad \text{(comm)}\quad \frac{\sigma_1 \xrightarrow{\mathtt{send},\sigma} \sigma_1' \quad \sigma_2 \xrightarrow{\mathtt{rcv},\sigma} \sigma_2'}{\sigma_1 \| \sigma_2 \xrightarrow{\tau} \sigma_1' \| \sigma_2'}$$

Fig. 13. Synchronous communication of replicas (symmetric rules are omitted).

of $\mathtt{W}$ and $\mathtt{W}'$ is more involved because it handles the conflicts due to concurrent operations over the same element. There is a conflict between $\mathtt{W}$ and $\mathtt{W}'$ for $(\mathtt{k}, \mathtt{s})$ when one mapping indicates that $\mathtt{k}$ is present in the replica $\mathtt{s}$ and the other does not, i.e., $\mathtt{W}(\mathtt{k}, \mathtt{s}) > 0$ and $\mathtt{W}'(\mathtt{k}, \mathtt{s}) = 0$ or vice versa. In case of a conflict, the mappings $\mathtt{V}$ and $\mathtt{V}'$ are used to resolve it. The operator $\oplus$ is defined as follows

$$(\mathtt{V}, \mathtt{W}) \oplus (\mathtt{V}', \mathtt{W}')(\mathtt{k}, \mathtt{s}) = \begin{cases} 0 & \textsc{IsREM}(\mathtt{W}, \mathtt{W}', \mathtt{V}, \mathtt{k}, \mathtt{s}) \ \vee \textsc{IsREM}(\mathtt{W}', \mathtt{W}, \mathtt{V}', \mathtt{k}, \mathtt{s}) \\ \max\{\mathtt{W}(\mathtt{k}, \mathtt{s}), \mathtt{W}'(\mathtt{k}, \mathtt{s})\} & \text{otherwise} \end{cases}$$

where $\textsc{IsREM}(\mathtt{W}, \mathtt{W}', \mathtt{V}, \mathtt{k}, \mathtt{s}) = (\mathtt{W}(\mathtt{k}, \mathtt{s}) = 0 \wedge \mathtt{W}'(\mathtt{k}, \mathtt{s}) \leq \mathtt{V}(\mathtt{s}))$ is the predicate characterising the fact that $\mathtt{k}$ has been removed from the replica $\mathtt{s}$ accordingly to the mappings $\mathtt{W}$, $\mathtt{W}'$, and $\mathtt{V}$.

We show that a replica is correct with respect to the specification in Example 8 by showing that the following relation satisfies the conditions in Definition 21

$$\begin{aligned} \mathcal{I} = \{(\langle \mathtt{r}, \mathtt{V}, \mathtt{W}\rangle, \langle \mathtt{G}, \mathtt{P}\rangle) \mid & \text{ there exists } f : \mathcal{E}_{\mathtt{G}} \to \mathcal{R} \text{ such that} \\ & \forall \mathtt{k}.(\exists \mathtt{r} \in \mathcal{R}.\mathtt{W}(\mathtt{k}, \mathtt{r}) > 0 \iff \exists\, \mathtt{S}.\ \mathcal{S}_{\textit{Or-Set}}(\mathtt{G}^{\langle \mathtt{lookup}, \mathtt{S}\rangle}) \neq \emptyset \ \wedge \ \mathtt{k} \in \mathtt{S})\} \end{aligned}$$

Correctness and refinement. It is straightforward to notice that correctness is preserved by refinement, i.e., if $\mathcal{S}$ is a refinement of $\mathcal{S}'$ and an implementation I is correct with respect to $\mathcal{S}$, then I is correct with respect to $\mathcal{S}'$. This follows from the fact that $\langle \mathtt{G}, \mathtt{P}\rangle \xrightarrow{\mathtt{op},\mathtt{v}} \langle \mathtt{G}', \mathtt{P}'\rangle$ in $\mathcal{S}$ implies $\langle \mathtt{G}, \mathtt{P}\rangle \xrightarrow{\mathtt{op},\mathtt{v}} \langle \mathtt{G}', \mathtt{P}'\rangle$ in $\mathcal{S}'$. Consequently, we can conclude that the implementation in Example 17 is also a correct implementation of the non-deterministic specification of $\mathcal{S}_{\textit{Set}}$ introduced in Example 8.

5.4. On the behaviour of multiple replicas

We now address the problem of showing that the parallel composition of several correct replicas is actually a correct implementation of a data type. In this section we focus on coherent specifications and start by considering the standard synchronous communication model, which is defined as follows.

Definition 22 *(Synchronous implementation).* Let $\mathcal{S}$ be a specification and $\mathcal{C}_{\mathcal{S}}$ an implementation. Then the *synchronous* extension $\mathcal{C}_{\mathcal{S}}^{\sigma}$ of $\mathcal{C}_{\mathcal{S}}$ is obtained by closing the set of states with a binary operation $\|$ and extending the transition relation with the additional rules in Fig. 13 (where symmetric versions are omitted).

We now want to reach the conclusion that whenever we have an implementation $\mathcal{C}_{\mathcal{S}}$ that is one-replica correct, then the synchronous extension $\mathcal{C}_{\mathcal{S}}^{\sigma}$ is multi-replica correct. The intuition is that the state of the parallel composition of replicas is described in terms of the configurations associated with each of the components. The following result suffices for our purposes, since it states that the simulation for a coherent specification is closed under synchronous parallel composition, and it relies on Lemma 6.

Lemma 7. *Let $\mathcal{S}$ be a coherent specification, $\mathcal{C}_{\mathcal{S}}$ an implementation that is one-replica correct, and σ_0, σ_1 two states of $\mathcal{C}_{\mathcal{S}}$. If $\sigma_i \sim_{\mathcal{S}}^{o} \langle \mathtt{G}_i, \mathtt{P}_i\rangle$ for $i \in \{1, 2\}$, then $\sigma_1 \| \sigma_2 \sim_{\mathcal{S}}^{m} \langle \mathtt{G}_1 \sqcup \mathtt{G}_2, \mathtt{P}\rangle$ for any $\mathtt{P} \in \mathtt{P}_1 \otimes \mathtt{P}_2$.*

Then, we obtain immediately the desired result.

Proposition 1. *Let $\mathcal{S}$ be a coherent specification and $\mathcal{C}_{\mathcal{S}}$ an implementation that is one-replica correct. Then, $\mathcal{C}_{\mathcal{S}}^{\sigma}$ is multi-replica correct.*

Example 18. The result above allows us to conclude that the implementation consisting of the parallel composition of replicas behaving as described in Example 15 is correct with respect to the specification of $\mathcal{S}_{\mathtt{Ctr}}$ in Example 4, which is coherent (see Example 9). Similarly, we can conclude that the parallel composition of several replicas for `OR-Set` in Example 17 is a correct implementation.

We remark that different communication models can be accommodated analogously. We may consider the (admittedly simplistic) asynchronous implementation $\mathcal{C}_{\mathcal{S}}^{\alpha}$ obtained by adding a family of operators $|_{\mathtt{B}}$ defined in Fig. 14, where $\mathtt{B}$ denotes a FIFO buffer. Formally, we represent $\mathtt{B}$ as a sequence $\sigma_1 \cdot \ldots \cdot \sigma_n$ of states and write ε for the empty sequence. Rule (sendL)

$$\text{(PARL)}\quad \frac{\sigma_1 \xrightarrow{\ell} \sigma_1'}{\sigma_1 \mid_{\mathrm{B}} \sigma_2 \xrightarrow{\ell} \sigma_1' \mid_{\mathrm{B}} \sigma_2} \qquad \text{(SENDL)}\quad \frac{\sigma_1 \xrightarrow{\mathtt{send},\sigma} \sigma_1'}{\sigma_1 \mid_{\mathrm{B}} \sigma_2 \xrightarrow{\tau} \sigma_1' \mid_{\mathrm{B}\cdot\sigma} \sigma_2} \qquad \text{(RECEIVEL)}\quad \frac{\sigma_1 \xrightarrow{\mathtt{rcv},\sigma} \sigma_1'}{\sigma_1 \mid_{\sigma\cdot\mathrm{B}} \sigma_2 \xrightarrow{\tau} \sigma_1' \mid_{\mathrm{B}} \sigma_2}$$

Fig. 14. Buffered communication of replicas (symmetric rules are omitted)

says that the state σ sent by a replica is added at the end of the buffer. Symmetrically, rule (RECEIVEL) states that a replica consumes updates from the beginning of the buffer. Also in this case, we can prove that implementation correctness is preserved.

Lemma 8. *Let $\mathcal{S}$ be a coherent specification, $\mathcal{C}_{\mathcal{S}}$ an implementation that is one-replica correct, and σ_0, σ_1 two states of $\mathcal{C}_{\mathcal{S}}$. If $\sigma_i \sim^{o}_{\mathcal{S}} \langle \mathtt{G_i}, \mathtt{P_i} \rangle$ for $i \in \{1, 2\}$, then $\sigma_1 \mid_{\varepsilon} \sigma_2 \sim^{m}_{\mathcal{S}} \langle \mathtt{G_1} \sqcup \mathtt{G_2}, \mathtt{P} \rangle$ for any $\mathtt{P} \in \mathtt{P_1} \otimes \mathtt{P_2}$.*

Proposition 2. *Let $\mathcal{S}$ be a coherent specification and $\mathcal{C}_{\mathcal{S}}$ be an implementation that is one-replica correct. Then, $\mathcal{C}^{\alpha}_{\mathcal{S}}$ is multi-replica correct.*

6. Related works

Weak consistency require to deal with conflict resolution [2,14,15] and convergence of replicas [8,16–18]. The design and implementation of data types that ensure convergence has been a very active area of research, notably, conflict-free and commutative replicated data type [2,14], which avoid implementing conflict resolution policies.

Different lines of work have addressed the problem of specifying and implementing replicated data types, considering also those that require policies for conflict resolution [3,17]. The approach in [3] has been addressed in detail in § 4. We remark that all of their specifications implicitly define a precise strategy to resolve conflicts. On the contrary, our specifications provide a more abstract view of RDTs, which do not commit to a particular strategy for conflict resolution. Implementation correctness is characterised in [3] in terms of *replication-aware simulations*. A replication-aware simulation is defined on top of abstract executions decorated with auxiliary information, such as time-stamps, which is defined *ad hoc* for each data type. Global correctness, i.e., the behaviour of several replicas, is derived from some agreement properties that impose a set of proof obligations that need to be checked. On the contrary, our characterisation of correctness relies on the abstract states induced by the specification. In our case, checking implementation correctness reduces to standard LTS simulation. Their approach can be instantiated to handle both state-based and operation-based implementations.

A different approach has been proposed in [17] to deal with optimistic replication systems. In this case, a specification associates an operation and a return value to the set of all possible executions (represented in terms of partial orders) that explain that particular return value for that operation. Moreover, they allow different operations to be associated with different partial orders. In this way, they deal with speculative systems. Furthermore, they reduce the problem of verifying eventual consistency to a model-checking and reachability problem.

A framework combining RDTs and transactions has been presented in [6]. As shown in [13], our specification style enables a categorical presentation of RDTs and the development of composition operators. Our results for parallel operators and implementation correctness suggests a way of dealing with RDT composition.

As far as the verification of (commutative) replicated data types is concerned, a framework in Isabelle/HOL has been proposed in [19], and other lines of work [8,9] have focused on the related problem of verifying properties of applications that use replicated data. In the long run, our goal is to exploit tools and techniques from the theory of simulation for precisely such purposes.

7. Conclusions and future works

We propose a denotational view of replicated data types. While most of the traditional approaches are operational in flavour [3,6,9], we strived for a specification formalism that could exploit the tools of algebraic specification theory. More precisely, we associate to each configuration (i.e., visibility) a set of admissible arbitrations. Differently from previous approaches, our presentation naturally accommodates non-deterministic specifications and enables abstract definitions allowing for different strategies in conflict resolution. Our formulation brings to light some properties held by mainstream specification formalisms: beside the obvious property of (local) functionality, they satisfy coherence and saturation. A coherent specification can neither prescribe an arbitration order between events that are unrelated by visibility nor allow for additional arbitrations over past events when a configuration is extended (i.e., a new top element is added to visibility). Instead, a saturated specification cannot impose any constraint to the arbitration of top events. Note that saturation does not hold when requiring that admissible arbitrations should be also topological orderings of visibility. Hence, the approaches in [3,4] generate specifications that are not saturated. We remark that this relation between visibility and arbitration translates in a quite different property in our setting, and this suggests that consistency models defined as relations between visibility

and arbitration (e.g., *monotonic* and *causal* consistency) could have alternative characterisations. We plan to explore these connections in future works.

Another question concerns coherence, which prevents a specification from choosing an arbitration order on events that are unrelated by visibility and forbids, e.g., the definition of strategies that arbitrate first the events coming from a particular replica. Consequently, it becomes natural to look for those RDTs and consistency models that are the counterpart of non-coherent specifications, still preserving some suitable notion of causality between events. We consider that the weaker property $\mathcal{S}(\mathtt{G})|_{-\prec^*\mathsf{e}} \subseteq \mathcal{S}(\mathtt{G}|_{-\prec^*\mathsf{e}})$ (that is, no additional arbitration over past events when a configuration is extended) is a worthwhile alternative, accommodating for many examples that impose less restrictions on the set of admissible paths (hence, that may allow more freedom to the arbitration).

We proposed an approach for checking that an implementation is correct with respect to a specification. Our characterisation relies on a notion of simulation and enjoys the property of being compositional with respect to standard communication models. This relieves us from checking that the behaviour of several replicas is correct. Our characterisation focuses only on state-based implementations. We are planning to extend the characterisation to operation-based implementations.

Acknowledgements

We thank the anonymous reviewers for their careful reading of our paper and their insightful comments. The first author has been partially supported by CONICET International Cooperation Grant 995/15. The second and third author have been partially supported by UBACyT project 2014–2017 20020130200092BA and CONICET project PIP 11220130100148CO.

Appendix A. Proof of the results in Section 4

Proof of Lemma 1. Let $\mathtt{P} \in \bigotimes_{\mathsf{e}\in\mathcal{E}_\mathtt{G}} \textsc{prval}(\mathcal{F}, \mathtt{G}|_{-\prec^*\mathsf{e}})$, that is, $\forall \mathsf{e} \in \mathcal{E}_\mathtt{G}.\ \ \mathtt{P}|_{-\prec^*\mathsf{e}} \in \textsc{prval}(\mathcal{F}, \mathtt{G}|_{-\prec^*\mathsf{e}})$. By return value consistency (Definition 10), this is equivalent to

$$\forall \mathsf{e} \in \mathcal{E}_\mathtt{G}.\ \forall \mathsf{e}' \in \mathcal{E}_{\mathtt{G}|_{-\prec^*\mathsf{e}}}.\ \lambda(\mathsf{e}') = \langle \mathtt{op}', \mathtt{v}'\rangle \text{ implies } \mathcal{F}(\mathtt{op}', (\bar{\mathtt{G}}\big|_{-\prec^*\mathsf{e}})\Big|_{-\prec\mathsf{e}'}, (\bar{\mathtt{P}}\big|_{-\prec^*\mathsf{e}})\Big|_{-\prec\mathsf{e}'}) = \mathtt{v}'$$

Note that $(\bar{\mathtt{G}}\big|_{-\prec^*\mathsf{e}})\Big|_{-\prec\mathsf{e}'}$ coincides with $\bar{\mathtt{G}}\big|_{-\prec\mathsf{e}'}$, and the same for $\bar{\mathtt{P}}$. Then, the result follows because the formula above coincides with

$$\forall \mathsf{e} \in \mathcal{E}_\mathtt{G}.\ \lambda(\mathsf{e}) = \langle \mathtt{op}, \mathtt{v}\rangle \text{ implies } \mathcal{F}(\mathtt{op}, \bar{\mathtt{G}}\big|_{-\prec\mathsf{e}}, \bar{\mathtt{P}}\big|_{-\prec\mathsf{e}}) = \mathtt{v} \qquad \square$$

Proof of Lemma 2. Consider $\mathtt{G}_1, \mathtt{G}_2$ such that $\overline{\mathtt{G}_1} = \overline{\mathtt{G}_2}$ and $\mathtt{G}_1 \neq \mathtt{G}_2$. We prove that $\overline{\mathcal{S}(\mathtt{G}_1)} \cap \overline{\mathcal{S}(\mathtt{G}_2)} = \emptyset$ follows. Since $\mathtt{G}_1 \neq \mathtt{G}_2$ there exists an event $\bar{\mathsf{e}}$ such that

$$\mathtt{G}_1|_{-\prec^+\bar{\mathsf{e}}} = \mathtt{G}_2|_{-\prec^+\bar{\mathsf{e}}} \text{ and } \lambda_1(\bar{\mathsf{e}}) = \langle \mathtt{op}, \mathtt{v}_1\rangle, \lambda_2(\bar{\mathsf{e}}) = \langle \mathtt{op}, \mathtt{v}_2\rangle \text{ for } \mathtt{v}_1 \neq \mathtt{v}_2$$

Let $\mathtt{G} = \mathtt{G}_\mathtt{i}|_{-\prec^+\bar{\mathsf{e}}}$. By determinism, we have that $\overline{\mathcal{S}(\mathtt{G}^{\lambda_1(\bar{\mathsf{e}})}_{-\prec\bar{\mathsf{e}}})}\Big|_{\mathcal{E}_\mathtt{G}} \cap \overline{\mathcal{S}(\mathtt{G}^{\lambda_2(\bar{\mathsf{e}})}_{-\prec\bar{\mathsf{e}}})}\Big|_{\mathcal{E}_\mathtt{G}} = \emptyset$, and equivalently that $\overline{\mathcal{S}(\mathtt{G}_1|_{-\prec^*\bar{\mathsf{e}}})}\Big|_{-\prec^+\bar{\mathsf{e}}} \cap \overline{\mathcal{S}(\mathtt{G}_2|_{-\prec^*\bar{\mathsf{e}}})}\Big|_{-\prec^+\bar{\mathsf{e}}} = \emptyset$.

Now, assume that there exist $\mathtt{P}_\mathtt{i} \in \mathcal{S}(\mathtt{G}_\mathtt{i})$ such that $\overline{\mathtt{P}_1} = \overline{\mathtt{P}_2}$, then by coherence $\forall \mathsf{e}.\ \ \mathtt{P}_\mathtt{i}|_{-\prec^*\mathsf{e}} \in \mathcal{S}(\mathtt{G}_\mathtt{i}|_{-\prec^*\mathsf{e}})$. Consequently, $\forall \mathsf{e}.\ \ \mathtt{P}_\mathtt{i}|_{-\prec^+\mathsf{e}} \in \mathcal{S}(\mathtt{G}_\mathtt{i}|_{-\prec^*\mathsf{e}})\big|_{-\prec^+\mathsf{e}}$, which contradicts the statement above for $\mathsf{e} = \bar{\mathsf{e}}$. $\square$

Proof of Lemma 13. By locality, for any $\mathtt{G}$ and $\mathcal{E}$ there exists $\mathtt{G}_1$ such that for all $\mathtt{op}, \mathtt{v}$, we have $\bar{\mathtt{G}}\big|_{\mathcal{E}} = \overline{\mathtt{G}_1} \ \wedge\ \overline{\mathcal{S}(\mathtt{G}^{\langle \mathtt{op},\mathtt{v}\rangle}_{\mathcal{E}})}\Big|_{-\preccurlyeq\top} \subseteq \overline{\mathcal{S}(\mathtt{G}_1{}^{\langle \mathtt{op},\mathtt{v}\rangle})}$. Consequently, for any $\mathtt{op}$ and $\mathtt{v} \neq \mathtt{v}'$,

$$\overline{\mathcal{S}(\mathtt{G}^{\langle \mathtt{op},\mathtt{v}\rangle}_{\mathcal{E}})}\Big|_{-\preccurlyeq\top} \subseteq \overline{\mathcal{S}(\mathtt{G}_1{}^{\langle \mathtt{op},\mathtt{v}\rangle})}$$

$$\overline{\mathcal{S}(\mathtt{G}^{\langle \mathtt{op},\mathtt{v}'\rangle}_{\mathcal{E}})}\Big|_{-\preccurlyeq\top} \subseteq \overline{\mathcal{S}(\mathtt{G}_1{}^{\langle \mathtt{op},\mathtt{v}'\rangle})}$$

If $\mathcal{S}$ is not deterministic, then for some $\mathtt{op}$ and $\mathtt{v} \neq \mathtt{v}'$ we have $\overline{\mathcal{S}(\mathtt{G}^{\langle \mathtt{op},\mathtt{v}\rangle}_{\mathcal{E}})}\Big|_{\mathcal{E}_\mathtt{G}} \cap \overline{\mathcal{S}(\mathtt{G}^{\langle \mathtt{op},\mathtt{v}'\rangle}_{\mathcal{E}})}\Big|_{\mathcal{E}_\mathtt{G}} \neq \emptyset$. Therefore $\overline{\mathcal{S}(\mathtt{G}^{\langle \mathtt{op},\mathtt{v}\rangle}_{\mathcal{E}})}\Big|_{-\preccurlyeq\top} \cap \overline{\mathcal{S}(\mathtt{G}^{\langle \mathtt{op},\mathtt{v}'\rangle}_{\mathcal{E}})}\Big|_{-\preccurlyeq\top} \neq \emptyset$ and thus we have $\overline{\mathcal{S}(\mathtt{G}_1{}^{\langle \mathtt{op},\mathtt{v}\rangle})} \cap \overline{\mathcal{S}(\mathtt{G}_1{}^{\langle \mathtt{op},\mathtt{v}'\rangle})} \neq \emptyset$, against weakly determinism for $\mathtt{G}_1$. $\square$

Proof of Lemma 4. We prove the four properties of $\mathbb{S}(\mathcal{F})$.

- Coherent. Immediate by Lemma 1.
- Saturated. We have to prove that

$$\forall \langle \mathtt{G}, \mathtt{P}\rangle, \mathcal{E}, \langle \mathtt{op}, \mathtt{v}\rangle.\ \mathtt{P} \in \mathbb{S}(\mathcal{F})(\mathtt{G}_{\mathcal{E}}^{\langle \mathtt{op},\mathtt{v}\rangle})\Big|_{\mathcal{E}_{\mathtt{G}}} \text{ implies } \mathtt{sat}(\mathtt{P}, \langle \mathtt{op}, \mathtt{v}\rangle) \subseteq \mathbb{S}(\mathcal{F})(\mathtt{G}_{\mathcal{E}}^{\langle \mathtt{op},\mathtt{v}\rangle})$$

  Since $\mathtt{P} \in \mathbb{S}(\mathcal{F})(\mathtt{G}_{\mathcal{E}}^{\langle \mathtt{op},\mathtt{v}\rangle})\Big|_{\mathcal{E}_{\mathtt{G}}}$, then there is $\mathtt{P}_1 \in \mathtt{sat}(\mathtt{P}, \langle \mathtt{op}, \mathtt{v}\rangle) \cap \mathbb{S}(\mathcal{F})(\mathtt{G}_{\mathcal{E}}^{\langle \mathtt{op},\mathtt{v}\rangle})$.

$$\begin{array}{rcl}
\mathtt{P}_1 \in \mathbb{S}(\mathcal{F})(\mathtt{G}_{\mathcal{E}}^{\langle \mathtt{op},\mathtt{v}\rangle}) & \overset{\text{Definition 14}}{\equiv} & \mathtt{P}_1 \in \textsc{prval}(\mathcal{F}, \mathtt{G}_{\mathcal{E}}^{\langle \mathtt{op},\mathtt{v}\rangle}) \\
& \overset{\text{Definition 10}}{\equiv} & \textsc{rval}(\mathcal{F}, \mathtt{G}_{\mathcal{E}}^{\langle \mathtt{op},\mathtt{v}\rangle}, \mathtt{P}_1) \\
& \overset{\text{Definition 10}}{\equiv} & \textsc{rval}(\mathcal{F}, \mathtt{G}_{\mathcal{E}}^{\langle \mathtt{op},\mathtt{v}\rangle}\Big|_{\mathcal{E}_{\mathtt{G}}}, \mathtt{P}_1|_{\mathcal{E}_{\mathtt{G}}}) \wedge \mathcal{F}(\mathtt{op}, \overline{\mathtt{G}_{\mathcal{E}}^{\langle \mathtt{op},\mathtt{v}\rangle}}\Big|_{-\prec\top}, \overline{\mathtt{P}_1}|_{-\prec\top}) = \mathtt{v} \\
& = & \textsc{rval}(\mathcal{F}, \mathtt{G}, \mathtt{P}) \wedge \mathcal{F}(\mathtt{op}, \overline{\mathtt{G}}|_{\mathcal{E}}, \overline{\mathtt{P}_1}|_{\mathcal{E}}) = \mathtt{v}
\end{array}$$

  Now, for every $\mathtt{P}_2 \in \mathtt{sat}(\mathtt{P}, \langle \mathtt{op}, \mathtt{v}\rangle)$ note that $\overline{\mathtt{P}_2}|_{\mathcal{E}} = \overline{\mathtt{P}_1}|_{\mathcal{E}}$ and, consequently, $\mathcal{F}(\mathtt{op}, \overline{\mathtt{G}}|_{\mathcal{E}}, \overline{\mathtt{P}_1}|_{\mathcal{E}}) = \mathtt{v}$ holds. Hence, $\mathtt{P}_2 \in \mathbb{S}(\mathcal{F})(\mathtt{G}_{\mathcal{E}}^{\langle \mathtt{op},\mathtt{v}\rangle})$ holds and the result follows.
- Total. Let us assume that there exist $\mathtt{G}$, $\mathtt{P}$, $\mathcal{E}$, and $\mathtt{op}$ such that

$$\forall \mathtt{G}_1, \mathtt{v}.\ \overline{\mathtt{G}} = \overline{\mathtt{G}_1} \text{ implies } \overline{\mathtt{P}} \notin \overline{\mathbb{S}(\mathcal{F})((\mathtt{G}_1)_{\mathcal{E}}^{\langle \mathtt{op},\mathtt{v}\rangle})}\Big|_{\mathcal{E}_{\mathtt{G}_1}} \tag{1}$$

  Moreover, without loss of generality we can assume that $\mathtt{G}$ is a minimal LDAG that satisfies (1), i.e., that for all $\mathtt{G}'$ strictly contained in $\mathtt{G}$ and for all $\mathtt{P}' \in \mathbb{S}(\mathcal{F})(\mathtt{G}')$ we have

$$\forall \mathcal{E}', \mathtt{op}'.\ \exists \mathtt{G}_1', \mathtt{v}'.\ \overline{\mathtt{G}'} = \overline{\mathtt{G}_1'} \ \wedge\ \overline{\mathtt{P}'} \in \overline{\mathbb{S}(\mathcal{F})((\mathtt{G}_1')_{\mathcal{E}}^{\langle \mathtt{op}',\mathtt{v}'\rangle})}\Big|_{\mathcal{E}_{\mathtt{G}_1'}}$$

  Now, let us consider $\mathtt{G}'$, $\mathcal{E}'$, and $\mathtt{op}'$ such that $\overline{\mathtt{G}} = (\overline{\mathtt{G}'})_{\mathcal{E}'}^{\mathtt{op}'}$ and let $\mathtt{P}' = \mathtt{P}|_{\mathcal{E}_{\mathtt{G}'}}$. Since we proved that $\mathbb{S}(\mathcal{F})$ is saturated we have

$$\exists \langle \mathtt{G}_1', \mathtt{P}_1'\rangle,\ \mathtt{v}'.\ \overline{\mathtt{G}'} = \overline{\mathtt{G}_1'} \ \wedge\ \overline{\mathtt{P}'} = \overline{\mathtt{P}_1'} \ \wedge\ \mathtt{sat}(\mathtt{P}_1', \langle \mathtt{op}', \mathtt{v}'\rangle) \subseteq \mathbb{S}(\mathcal{F})((\mathtt{G}_1')_{\mathcal{E}'}^{\langle \mathtt{op}',\mathtt{v}'\rangle})$$

  Then, let us take $\mathtt{G}_1 = (\mathtt{G}_1')_{\mathcal{E}'}^{\langle \mathtt{op}',\mathtt{v}'\rangle}$ and $\mathtt{P}_1 \in \mathtt{sat}(\mathtt{P}_1', \langle \mathtt{op}', \mathtt{v}'\rangle)$ such that $\overline{\mathtt{P}} = \overline{\mathtt{P}_1}$ and $\mathtt{P}_1 \in \mathbb{S}(\mathcal{F})(\mathtt{G}_1)$. First, note that $\overline{\mathtt{G}} = \overline{\mathtt{G}_1}$ because $\overline{\mathtt{G}} = (\overline{\mathtt{G}'})_{\mathcal{E}'}^{\mathtt{op}'}$, $\overline{\mathtt{G}'} = \overline{\mathtt{G}_1'}$, and $\mathtt{G}_1 = (\mathtt{G}_1')_{\mathcal{E}'}^{\langle \mathtt{op}',\mathtt{v}'\rangle}$. We now show that for each $\mathcal{E}$ and $\mathtt{op}$ there exists $\mathtt{v}$ such that $\mathtt{P}_1{}^{\langle \mathtt{op},\mathtt{v}\rangle} \in \mathbb{S}(\mathcal{F})((\mathtt{G}_1)_{\mathcal{E}}^{\langle \mathtt{op},\mathtt{v}\rangle})$, which is in contradiction with the assumption (1). Given $\mathcal{E}$ and $\mathtt{op}$, take $\mathtt{v}$ such that $\mathcal{F}(\mathtt{op}, \overline{\mathtt{G}_1}|_{\mathcal{E}}, \overline{\mathtt{P}_1}|_{\mathcal{E}}) = \mathtt{v}$ (such v exists because of the definition of $\mathcal{F}$). By following the same reasoning as for saturation we have

$$\begin{array}{rcl}
\mathtt{P}_1{}^{\langle \mathtt{op},\mathtt{v}\rangle} \in \mathbb{S}(\mathcal{F})((\mathtt{G}_1)_{\mathcal{E}}^{\langle \mathtt{op},\mathtt{v}\rangle}) & \overset{\text{Definition 14}}{\equiv} & \mathtt{P}_1{}^{\langle \mathtt{op},\mathtt{v}\rangle} \in \textsc{prval}(\mathcal{F}, (\mathtt{G}_1)_{\mathcal{E}}^{\langle \mathtt{op},\mathtt{v}\rangle}) \\
& \overset{\text{Definition 10}}{\equiv} & \cdots \\
& = & \textsc{rval}(\mathcal{F}, \mathtt{G}_1, \mathtt{P}_1) \wedge \mathcal{F}(\mathtt{op}, \overline{\mathtt{G}_1}|_{\mathcal{E}}, \overline{\mathtt{P}_1}|_{\mathcal{E}}) = \mathtt{v}
\end{array}$$

  Since $\textsc{rval}(\mathcal{F}, \mathtt{G}_1, \mathtt{P}_1)$ coincides with $\mathtt{P}_1 \in \mathbb{S}(\mathcal{F})(\mathtt{G}_1)$, the result follows.
- Deterministic. Let $\mathtt{G}$, $\mathcal{E}$, $\mathtt{op}$, $\mathtt{v}_1$, and $\mathtt{v}_2$ such that $\mathtt{v}_1 \neq \mathtt{v}_2$ and

$$\overline{\mathbb{S}(\mathcal{F})(\mathtt{G}_{\mathcal{E}}^{\langle \mathtt{op},\mathtt{v}_1\rangle})}\Big|_{\mathcal{E}_{\mathtt{G}}} \cap \overline{\mathbb{S}(\mathcal{F})(\mathtt{G}_{\mathcal{E}}^{\langle \mathtt{op},\mathtt{v}_2\rangle})}\Big|_{\mathcal{E}_{\mathtt{G}}} \neq \emptyset$$

  Then, there exist paths $\mathtt{P}_1$, $\mathtt{P}_2$ such that $\mathtt{P}_i \in \mathbb{S}(\mathcal{F})(\mathtt{G}_{\mathcal{E}}^{\langle \mathtt{op},\mathtt{v}_i\rangle})$ and $\overline{\mathtt{P}_1}|_{\mathcal{E}_{\mathtt{G}}} = \overline{\mathtt{P}_2}|_{\mathcal{E}_{\mathtt{G}}}$. By the definition of $\mathbb{S}(\mathcal{F})$ (see Definition 14), the former is equivalent to $\mathtt{P}_i \in \textsc{prval}(\mathcal{F}, \mathtt{G}_{\mathcal{E}}^{\langle \mathtt{op},\mathtt{v}_i\rangle})$ and then to $\textsc{rval}(\mathcal{F}, \mathtt{G}_{\mathcal{E}}^{\langle \mathtt{op},\mathtt{v}_i\rangle}, \mathtt{P}_i)$. By return value consistency (see Definition 10), we have that $\mathcal{F}(\mathtt{op}, \overline{\mathtt{G}_{\mathcal{E}}^{\langle \mathtt{op},\mathtt{v}_i\rangle}}\Big|_{-\prec\top}, \overline{\mathtt{P}_i}|_{-\prec\top}) = \mathtt{v}_i$, and since $\mathtt{G}_{\mathcal{E}}^{\langle \mathtt{op},\mathtt{v}_i\rangle}\Big|_{-\prec\top} = \mathtt{G}|_{\mathcal{E}}$ and $\mathtt{P}_i|_{-\prec\top} = \mathtt{P}_i|_{\mathcal{E}}$, it results in a contradiction.
- Local. For every $\mathtt{P}$ such that $\overline{\mathtt{P}} \in \overline{\mathbb{S}(\mathcal{F})(\mathtt{G}_{\mathcal{E}}^{\langle \mathtt{op},\mathtt{v}\rangle})}\Big|_{-\preccurlyeq\top}$ there exists $\mathtt{P}_1 \in \mathbb{S}(\mathcal{F})(\mathtt{G}_{\mathcal{E}}^{\langle \mathtt{op},\mathtt{v}\rangle})$ such that $\overline{\mathtt{P}} = \overline{\mathtt{P}_1}|_{-\preccurlyeq\top}$. Then, by following the same reasoning as for saturation we have

$$\begin{array}{rcl}
\mathtt{P}_1 \in \mathbb{S}(\mathcal{F})(\mathtt{G}_{\mathcal{E}}^{\langle \mathtt{op},\mathtt{v}\rangle}) & \overset{\text{Definition 14}}{\equiv} & \mathtt{P}_1 \in \textsc{prval}(\mathcal{F}, \mathtt{G}_{\mathcal{E}}^{\langle \mathtt{op},\mathtt{v}\rangle}) \\
& \overset{\text{Definition 10}}{\equiv} & \forall \mathsf{e} \in \mathcal{E}_{\mathtt{G}}.\ \lambda(\mathsf{e}) = \langle \mathtt{op}', \mathtt{v}'\rangle \text{ implies } \mathcal{F}(\mathtt{op}', \overline{\mathtt{G}}|_{-\prec\mathsf{e}}, \overline{\mathtt{P}_1}|_{-\prec\mathsf{e}}) \\
& = & \mathtt{v}' \wedge \mathcal{F}(\mathtt{op}, \overline{\mathtt{G}}|_{\mathcal{E}}, \overline{\mathtt{P}_1}|_{\mathcal{E}}) = \mathtt{v}
\end{array}$$

Since $\mathcal{F}$ is a function, for every $\mathtt{G}_1$ such that $\overline{\mathtt{G}}\big|_{\mathcal{I}} = \overline{\mathtt{G}_1}$ the following holds

$$\forall \mathsf{e} \in \mathcal{E}_{\mathtt{G}_1}, \mathtt{op}'.\exists \mathtt{v}''.\mathcal{F}(\mathtt{op}', \overline{\mathtt{G}_1}\big|_{-\prec \mathsf{e}}, \overline{\mathtt{P}_1}\big|_{-\prec \mathsf{e}}) = \mathtt{v}''$$

Consequently, by coherence there exists $\mathtt{P}_2 \in \mathbb{S}(\mathcal{F})(\mathtt{G}_1)$ such that $\overline{\mathtt{P}_1}\big|_{\mathcal{E}_{\mathtt{G}_1}} = \overline{\mathtt{P}_2}$. Moreover, $\mathcal{F}(\mathtt{op}, \overline{\mathtt{G}}\big|_{\mathcal{I}}, \overline{\mathtt{P}_1}\big|_{\mathcal{I}}) = \mathtt{v}$ and by saturation this implies that $\mathtt{sat}(\mathtt{P}_2, \langle \mathtt{op}, \mathtt{v}\rangle) \subseteq \mathbb{S}(\mathcal{F})(\mathtt{G}_1{}^{\langle \mathtt{op},\mathtt{v}\rangle})$. Therefore, $\overline{\mathtt{P}} \in \overline{\mathcal{S}(\mathtt{G}_1{}^{\langle \mathtt{op},\mathtt{v}\rangle})}$. □

Proof of Lemma 5. Since $\mathcal{S}$ is total, there exists at least a value for every triple $\mathtt{op}$, $\overline{\mathtt{G}}$, and $\overline{\mathtt{P}}$. Let us now assume that there are more than two values, i.e., that there exist $\mathtt{op}$, $\mathtt{G}_i$, and $\mathtt{v}_i$ such that

$$\overline{\mathtt{G}} = \overline{\mathtt{G}_1} = \overline{\mathtt{G}_2} \ \wedge\ \mathtt{v}_1 \neq \mathtt{v}_2 \ \wedge\ \overline{\mathtt{P}} \in \overline{\mathcal{S}(\mathtt{G}_i{}^{\langle \mathtt{op},\mathtt{v}_i\rangle})}\Big|_{\mathcal{E}_{\mathtt{G}_i}}$$

Since $\mathcal{S}$ is deterministic and $\overline{\mathtt{G}_1} = \overline{\mathtt{G}_2}$, two cases may occur by Lemma 2

- $\mathtt{G}_1 = \mathtt{G}_2$. By hypothesis we have that $\overline{\mathtt{P}} \in \overline{\mathcal{S}(\mathtt{G}_i{}^{\langle \mathtt{op},\mathtt{v}_i\rangle})}\Big|_{\mathcal{E}_{\mathtt{G}_i}}$, hence

  $$\overline{\mathcal{S}(\mathtt{G}_1{}^{\langle \mathtt{op},\mathtt{v}_1\rangle})}\Big|_{\mathcal{E}_{\mathtt{G}_1}} \cap \overline{\mathcal{S}(\mathtt{G}_1{}^{\langle \mathtt{op},\mathtt{v}_2\rangle})}\Big|_{\mathcal{E}_{\mathtt{G}_1}} \neq \emptyset$$

  Again by being $\mathcal{S}$ deterministic, it follows that $\mathtt{v}_1 = \mathtt{v}_2$.
- $\overline{\mathcal{S}(\mathtt{G}_1)} \cap \overline{\mathcal{S}(\mathtt{G}_2)} = \emptyset$. Since $\mathcal{S}$ is coherent and saturated and by hypothesis $\overline{\mathtt{P}} \in \overline{\mathcal{S}(\mathtt{G}_i{}^{\langle \mathtt{op},\mathtt{v}_i\rangle})}\Big|_{\mathcal{E}_{\mathtt{G}_i}}$, we have that $\overline{\mathtt{P}} \in \overline{\mathcal{S}(\mathtt{G}_i)}$, which leads to a contradiction. □

We state an auxiliary result that will be used in the proof of Theorem 1.

Lemma 9. *Let $\mathcal{S}$ be a coherent, functional, and saturated specification and $\langle \mathtt{G}, \mathtt{P}\rangle$ a context. If*

$$\forall \mathsf{e} \in \mathcal{E}_{\mathtt{G}}.\ \lambda(\mathsf{e}) = \langle \mathtt{op}, \mathtt{v}\rangle \textit{ implies } \exists \mathtt{G}_1.\ \overline{\mathtt{G}}\big|_{-\prec \mathsf{e}} = \overline{\mathtt{G}_1} \ \wedge\ \overline{\mathtt{P}}\big|_{-\prec \mathsf{e}} \in \overline{\mathcal{S}(\mathtt{G}_1{}^{\langle \mathtt{op},\mathtt{v}\rangle})}\Big|_{\mathcal{E}_{\mathtt{G}_1}} \tag{2}$$

then $\forall \mathsf{e} \in \mathcal{E}_{\mathtt{G}}.\ \mathtt{P}|_{-\prec^* \mathsf{e}} \in \mathcal{S}(\mathtt{G}|_{-\prec^* \mathsf{e}})$.

Moreover, let $\mathcal{S}$ be a local and coherent specification. Then, the vice versa holds.

Proof. ($\Leftarrow$) Let $\bar{\mathsf{e}}$ such that $\lambda(\bar{\mathsf{e}}) = \langle \mathtt{op}, \mathtt{v}\rangle$. By hypothesis $\mathtt{P}|_{-\prec^* \bar{\mathsf{e}}} \in \mathcal{S}(\mathtt{G}|_{-\prec^* \bar{\mathsf{e}}})$, by coherence $\mathtt{P}|_{-\prec^+ \bar{\mathsf{e}}} \in \mathcal{S}(\mathtt{G}|_{-\prec^+ \bar{\mathsf{e}}})$, and by locality

$$\exists \mathtt{G}_1.\ (\overline{\mathtt{G}}\big|_{-\prec^+ \bar{\mathsf{e}}})\Big|_{-\prec \bar{\mathsf{e}}} = \overline{\mathtt{G}_1} \ \wedge\ \overline{\mathcal{S}((\mathtt{G}|_{-\prec^+ \bar{\mathsf{e}}})^{\langle \mathtt{op},\mathtt{v}\rangle}_{-\prec \bar{\mathsf{e}}})}\Big|_{-\preccurlyeq \bar{\mathsf{e}}} \subseteq \overline{\mathcal{S}(\mathtt{G}_1{}^{\langle \mathtt{op},\mathtt{v}\rangle})}$$

This is in turn equivalent to

$$\exists \mathtt{G}_1.\ \overline{\mathtt{G}}\big|_{-\prec \bar{\mathsf{e}}} = \overline{\mathtt{G}_1} \ \wedge\ \overline{\mathcal{S}(\mathtt{G}|_{-\prec^* \bar{\mathsf{e}}})}\Big|_{-\preccurlyeq \bar{\mathsf{e}}} \subseteq \overline{\mathcal{S}(\mathtt{G}_1{}^{\langle \mathtt{op},\mathtt{v}\rangle})}$$

and again by the hypothesis $\mathtt{P}|_{-\prec^* \bar{\mathsf{e}}} \in \mathcal{S}(\mathtt{G}|_{-\prec^* \bar{\mathsf{e}}})$ the result follows.

($\Rightarrow$) By contradiction, let us assume that there exist $\bar{\mathsf{e}} \in \mathcal{E}_{\mathtt{G}}$ such that the equation (2) holds but $\mathtt{P}|_{-\prec^* \bar{\mathsf{e}}} \notin \mathcal{S}(\mathtt{G}|_{-\prec^* \mathsf{e}})$. Without loss of generality, assume e is minimal, i.e., for all e' such that $\mathsf{e}' \prec^+ \bar{\mathsf{e}}$ we have

$$\mathtt{P}|_{-\prec^* \mathsf{e}'} \in \mathcal{S}(\mathtt{G}|_{-\prec^* \mathsf{e}'})$$

By coherence we have

$$\mathtt{P}|_{-\prec^+ \bar{\mathsf{e}}} \in \mathcal{S}(\mathtt{G}|_{-\prec^+ \bar{\mathsf{e}}})$$

Assuming $\lambda(\bar{\mathsf{e}}) = \langle \mathtt{op}, \mathtt{v}\rangle$, by totality it follows

$$\exists \mathtt{G}_1, \mathtt{v}_1.\ \overline{\mathtt{G}_1} = \overline{\mathtt{G}}\big|_{-\prec^+ \bar{\mathsf{e}}} \ \wedge\ \overline{\mathtt{P}}\big|_{-\prec^+ \bar{\mathsf{e}}} \in \overline{\mathcal{S}((\mathtt{G}_1)^{\langle \mathtt{op},\mathtt{v}_1\rangle}_{-\prec \bar{\mathsf{e}}})}\Big|_{-\prec^+ \bar{\mathsf{e}}}$$

Now, by saturation and coherence we have that $\overline{\mathtt{P}}\big|_{-\prec^+ \bar{\mathsf{e}}} \in \overline{\mathcal{S}(\mathtt{G}_1)}$, thus by determinism $\mathtt{G}_1 = \mathtt{G}|_{-\prec^+ \bar{\mathsf{e}}}$, so that it holds

$$\exists \mathtt{v}_1.\ \overline{\mathtt{P}}\big|_{-\prec^+ \bar{\mathsf{e}}} \in \overline{\mathcal{S}((\mathtt{G}|_{-\prec^+ \bar{\mathsf{e}}})^{\langle \mathtt{op},\mathtt{v}_1\rangle}_{-\prec \bar{\mathsf{e}}})}\Big|_{-\prec^+ \bar{\mathsf{e}}}$$

Now, by saturation and (2)

$$\exists \mathtt{G}_2.\ \left(\overline{\mathtt{G}}\big|_{-\prec^+\bar{\mathtt{e}}}\right)\Big|_{-\prec\bar{\mathtt{e}}} = \overline{\mathtt{G}_2} \ \wedge\ \overline{\mathtt{P}}\big|_{-\prec\bar{\mathtt{e}}} \in \overline{\mathcal{S}((\mathtt{G}_2)^{\langle \mathtt{op},\mathtt{v}_1\rangle})}\Big|_{-\prec\bar{\mathtt{e}}}$$

However, applying (2) to $\mathtt{G}$ tells that

$$\exists \mathtt{G}_3.\ \overline{\mathtt{G}}\big|_{-\prec\bar{\mathtt{e}}} = \overline{\mathtt{G}_3} \ \wedge\ \overline{\mathtt{P}}\big|_{-\prec\bar{\mathtt{e}}} \in \overline{\mathcal{S}((\mathtt{G}_3)^{\langle \mathtt{op},\mathtt{v}\rangle})}\Big|_{-\prec\bar{\mathtt{e}}}$$

and by saturation and determinism $(\mathtt{G}_2)^{\langle \mathtt{op},\mathtt{v}_1\rangle} = (\mathtt{G}_3)^{\langle \mathtt{op},\mathtt{v}\rangle}$, so that $\mathtt{v} = \mathtt{v}_1$.

Thus now we have

$$\overline{\mathtt{P}}\big|_{-\prec^+\bar{\mathtt{e}}} \in \overline{\mathcal{S}(\mathtt{G}|_{-\prec^*\bar{\mathtt{e}}})}\Big|_{-\prec^+\bar{\mathtt{e}}}$$

and by saturation it holds that

$$\overline{\mathtt{P}}\big|_{-\prec^*\bar{\mathtt{e}}} \in \overline{\mathcal{S}(\mathtt{G}|_{-\prec^*\mathtt{e}})}$$

and since by hypothesis

$$\mathtt{P}|_{-\prec^+\bar{\mathtt{e}}} \in \mathcal{S}(\mathtt{G}|_{-\prec^+\mathtt{e}})$$

we have that

$$\mathtt{P}|_{-\prec^*\bar{\mathtt{e}}} \in \mathcal{S}(\mathtt{G}|_{-\prec^*\mathtt{e}})$$

which is in contradiction with the assumptions. □

Proof of Theorem 1. We have to prove that for any $\mathtt{G}$ it holds $\mathcal{S}(\mathtt{G}) = \mathbb{S}(\mathbb{F}(\mathcal{S}))(\mathtt{G})$.

$$\begin{array}{ll}
\mathtt{P} \in \mathbb{S}(\mathbb{F}(\mathcal{S}))(\mathtt{G}) & \\
\overset{\text{Definition 14}}{\equiv} & \mathtt{P} \in \mathbf{PRVAL}(\mathbb{F}(\mathcal{S}), \mathtt{G}) \\
\overset{\text{Definition 10}}{\equiv} & \mathbf{RVAL}(\mathbb{F}(\mathcal{S}), \mathtt{G}, \mathtt{P}) \\
\overset{\text{Definition 10}}{\equiv} & \forall \mathtt{e} \in \mathcal{E}_{\mathtt{G}}.\ \lambda(\mathtt{e}) = \langle \mathtt{op}, \mathtt{v}\rangle \text{ implies } \mathbb{F}(\mathcal{S})(\mathtt{op}, \overline{\mathtt{G}}\big|_{-\prec\mathtt{e}}, \overline{\mathtt{P}}\big|_{-\prec\mathtt{e}}) = \mathtt{v} \\
\overset{\text{Definition 15}}{\equiv} & \forall \mathtt{e} \in \mathcal{E}_{\mathtt{G}}.\ \lambda(\mathtt{e}) = \langle \mathtt{op}, \mathtt{v}\rangle \text{ implies } \exists \mathtt{G}_1.\ \overline{\mathtt{G}}\big|_{-\prec\mathtt{e}} = \overline{\mathtt{G}_1} \ \wedge\ \overline{\mathtt{P}}\big|_{-\prec\mathtt{e}} \in \overline{\mathcal{S}(\mathtt{G}_1{}^{\langle \mathtt{op},\mathtt{v}\rangle})}\Big|_{\mathcal{E}_{\mathtt{G}_1}} \\
\overset{\text{Lemma 9}}{\equiv} & \forall \mathtt{e} \in \mathcal{E}_{\mathtt{G}}.\ \ \mathtt{P}|_{-\prec^*\mathtt{e}} \in \mathcal{S}(\mathtt{G}|_{-\prec^*\mathtt{e}}) \\
\overset{\text{Definition 3}}{\equiv} & \mathtt{P} \in \bigotimes_{\mathtt{e}\in\mathcal{E}_{\mathtt{G}}} \mathcal{S}(\mathtt{G}|_{-\prec^*\mathtt{e}}) \\
\overset{\text{coh}}{\equiv} & \mathtt{P} \in \mathcal{S}(\mathtt{G}) \qquad \square
\end{array}$$

Proof of Theorem 2. We prove that $\mathcal{F}(\mathtt{op}, \overline{\mathtt{G}}, \overline{\mathtt{P}}) = \mathbb{F}(\mathbb{S}(\mathcal{F}))(\mathtt{op}, \overline{\mathtt{G}}, \overline{\mathtt{P}})$ for any $\langle \overline{\mathtt{G}}, \overline{\mathtt{P}}\rangle$ and $\mathtt{op}$.

$$\begin{array}{ll}
\mathbb{F}(\mathbb{S}(\mathcal{F}))(\mathtt{op}, \overline{\mathtt{G}}, \overline{\mathtt{P}}) = \mathtt{v} & \\
\overset{\text{Definition 15}}{\equiv} & \exists \langle \mathtt{G}_1, \mathtt{P}_1\rangle.\ \overline{\mathtt{G}} = \overline{\mathtt{G}_1} \wedge \overline{\mathtt{P}} = \overline{\mathtt{P}_1} \wedge \mathtt{P}_1 \in \mathbb{S}(\mathcal{F})(\mathtt{G}_1^{\langle \mathtt{op},\mathtt{v}\rangle})\Big|_{\mathcal{E}_{\mathtt{G}_1}} \\
\overset{\text{sat}}{\Rightarrow} & \exists \langle \mathtt{G}_1,\ \mathtt{P}_1\rangle.\ \overline{\mathtt{G}_1} = \overline{\mathtt{G}} \wedge \overline{\mathtt{P}} = \overline{\mathtt{P}_1} \wedge \mathtt{P}_1^{\langle \mathtt{op},\mathtt{v}\rangle} \in \mathbb{S}(\mathcal{F})(\mathtt{G}_1^{\langle \mathtt{op},\mathtt{v}\rangle}) \\
\overset{\text{Definition 15}}{\equiv} & \ldots \wedge \mathtt{P}_1^{\langle \mathtt{op},\mathtt{v}\rangle} \in \mathbf{PRVAL}(\mathcal{F}, \mathtt{G}_1^{\langle \mathtt{op},\mathtt{v}\rangle}) \\
\overset{\text{Definition 10}}{\equiv} & \ldots \wedge \mathbf{RVAL}(\mathcal{F}, \mathtt{G}_1^{\langle \mathtt{op},\mathtt{v}\rangle}, \mathtt{P}_1^{\langle \mathtt{op},\mathtt{v}\rangle}) \\
\overset{\text{Definition 10}}{\equiv} & \ldots \wedge \mathbf{RVAL}(\mathcal{F}, \mathtt{G}_1, \mathtt{P}_1) \wedge \mathcal{F}(\mathtt{op}, \overline{(\mathtt{G}_1)^{\langle \mathtt{op},\mathtt{v}\rangle}}\Big|_{-\prec\top}, \overline{(\mathtt{P}_1)^{\langle \mathtt{op},\mathtt{v}\rangle}}\Big|_{-\prec\top}) = \mathtt{v} \\
\equiv & \ldots \wedge \mathcal{F}(\mathtt{op}, \overline{\mathtt{G}_1}, \overline{\mathtt{P}_1}) = \mathtt{v} \\
\Rightarrow & \mathcal{F}(\mathtt{op}, \overline{\mathtt{G}}, \overline{\mathtt{P}}) = \mathtt{v}
\end{array}$$

Then the result holds, since $\mathcal{F}$ and $\mathbb{F}(\mathbb{S}(\mathcal{F}))$ are total functions. □

Appendix B. Proof of the results in Section 5

Proof of Lemma 6. It follows from the fact that $\mathcal{S}$ is coherent. □

Proof of Example 16. By case analysis on the derivation of $\sigma \xrightarrow{\alpha} \sigma'$. Consider $\sigma = \langle \mathtt{r}_\mathtt{j}, \mathtt{v}\rangle$ and $(\sigma, \langle \mathtt{G}, \mathtt{P}\rangle) \in \mathcal{I}$. Consequently, there exists $f : \mathcal{E}_{\mathtt{G}} \to \mathcal{R}$ such that

$$\forall \mathtt{r} \in \mathcal{R}.\mathtt{v}(\mathtt{r}) = \#\{\mathtt{e} \mid f(\mathtt{e}) = \mathtt{r} \text{ and } \lambda(\mathtt{e}) = \langle \mathtt{inc}, \mathtt{ok}\rangle\} \tag{3}$$

- (READ) Hence, $\sigma = \langle \mathsf{r}_j, \mathsf{v}\rangle \xrightarrow{\mathtt{rd},\mathtt{k}} \langle \mathsf{r}_j, \mathsf{v}\rangle = \sigma'$ and $\mathtt{k} = \sum_{\mathsf{r}\in\mathcal{R}} \mathsf{v}(\mathsf{r})$. From (3), $\mathtt{k} = \sum_{\mathsf{r}\in\mathcal{R}} \mathsf{v}(\mathsf{r}) = \#\{\mathsf{e} \mid \mathsf{e} \in \mathcal{E}_{\mathtt{G}} \ \wedge \ \lambda(\mathsf{e}) = \langle \mathtt{inc}, \mathtt{ok}\rangle\}$. Define $\mathtt{G}' = \mathtt{G}^{\langle \mathtt{rd},\mathtt{k}\rangle}$ and $f' : \mathcal{E}_{\mathtt{G}'} \to \mathcal{R}$ as follows

$$f'(\mathsf{e}) = \begin{cases} f(\mathsf{e}) & \text{if } \mathsf{e} \in \mathcal{E}_{\mathtt{G}} \\ \mathsf{r}_j & \text{if } \mathsf{e} = \top \end{cases}$$

It is immediate to check that

$$\forall \mathsf{r} \in \mathcal{R}.\mathsf{v}(\mathsf{r}) = \#\{\mathsf{e} \mid f'(\mathsf{e}) = \mathsf{r} \text{ and } \lambda(\mathsf{e}) = \langle \mathtt{inc}, \mathtt{ok}\rangle\}$$

Hence,

$$\mathtt{k} = \sum_{\mathsf{r}\in\mathcal{R}} \mathsf{v}(\mathsf{r}) = \#\{\mathsf{e} \mid \mathsf{e} \in \mathcal{E}_{\mathtt{G}'} \ \wedge \ \lambda(\mathsf{e}) = \langle \mathtt{inc}, \mathtt{ok}\rangle\}$$

Consequently, $\mathtt{sat}(\mathtt{P}, \langle \mathtt{rd}, \mathtt{k}\rangle) \subseteq \mathcal{S}_{\mathit{Ctr}}(\mathtt{G}')$. Then, for any $\mathtt{P}' \in \mathtt{sat}(\mathtt{P}, \langle \mathtt{rd}, \mathtt{k}\rangle)$, we have $(\langle \mathsf{r}_j, \mathsf{v}\rangle, \langle \mathtt{G}^{\langle \mathtt{rd},\mathtt{k}\rangle}, \mathtt{P}'\rangle) \in \mathcal{I}$.
- (INC) Then, $\sigma = \langle \mathsf{r}_j, \mathsf{v}\rangle \xrightarrow{\mathtt{inc},\mathtt{ok}} \langle \mathsf{r}_j, \mathsf{v}[\mathsf{r}_j \mapsto \mathsf{v}(\mathsf{r}_j) + 1]\rangle = \sigma'$. Take $\mathtt{G}' = \mathtt{G}^{\langle \mathtt{inc},\mathtt{ok}\rangle}$. It is immediate to check that $\mathtt{sat}(\mathtt{P}, \langle \mathtt{inc}, \mathtt{ok}\rangle) \subseteq \mathcal{S}_{\mathit{Ctr}}(\mathtt{G}')$. Then, define $f' : \mathcal{E}_{\mathtt{G}'} \to \mathcal{R}$ as follows

$$f'(\mathsf{e}) = \begin{cases} f(\mathsf{e}) & \text{if } \mathsf{e} \in \mathcal{E}_{\mathtt{G}} \\ \mathsf{r}_j & \text{if } \mathsf{e} = \top \end{cases}$$

Hence, $\#\{\mathsf{e} \mid f'(\mathsf{e}) = \mathsf{r} \text{ and } \lambda(\mathsf{e}) = \langle \mathtt{inc}, \mathtt{ok}\rangle\} = \mathsf{v}(\mathsf{r})$ for all $\mathsf{r} \neq \mathsf{r}_j$, and

$$\#\{\mathsf{e} \mid f'(\mathsf{e}) = \mathsf{r}_j \text{ and } \lambda(\mathsf{e}) = \langle \mathtt{inc}, \mathtt{ok}\rangle\} = \mathsf{v}(\mathsf{r}_j) + 1$$

Consequently, for any $\mathtt{P}' \in \mathtt{sat}(\mathtt{P}, \langle \mathtt{inc}, \mathtt{ok}\rangle)$ the following holds

$$(\langle \mathsf{r}_j, \mathsf{v}[\mathsf{r}_j \mapsto \mathsf{v}(\mathsf{r}_j) + 1]\rangle, \langle \mathtt{G}^{\langle \mathtt{inc},\mathtt{ok}\rangle}, \mathtt{P}'\rangle) \in \mathcal{I}$$

- (RECEIVE) Then, $\sigma = \langle \mathsf{r}, \mathsf{v}\rangle \xrightarrow{\mathtt{rcv},\langle \mathsf{r}',\mathsf{v}''\rangle} \langle \mathsf{r}, \mathsf{v}'\rangle = \sigma'$ such that $\mathsf{v}' = \max\{\mathsf{v}, \mathsf{v}''\}$. Define $\mathtt{G}' = \langle \mathcal{F}, \prec', \lambda'\rangle$ and $f' : \mathcal{E}_{\mathtt{G}'} \to \mathcal{R}$ such that $\forall \mathsf{r} \in \mathcal{R}$
  - $\mathsf{v}''(\mathsf{r}) = \#\{\mathsf{e} \mid f'(\mathsf{e}) = \mathsf{r} \text{ and } \lambda'(\mathsf{e}) = \langle \mathtt{inc}, \mathtt{ok}\rangle\}$
  - $\mathit{min}\{\mathsf{v}(\mathsf{r}), \mathsf{v}''(\mathsf{r})\} = \#\{\mathsf{e} \mid f(\mathsf{e}) = \mathsf{r}\} \cap \#\{\mathsf{e} \mid f'(\mathsf{e}) = \mathsf{r}\}$
  - $\prec_{\mathtt{G}'}|_{\mathcal{E}_{\mathtt{G}}\cap\mathcal{E}_{\mathtt{G}'}} = \prec_{\mathtt{G}}|_{\mathcal{E}_{\mathtt{G}}\cap\mathcal{E}_{\mathtt{G}'}}$,
  - $\lambda_{\mathtt{G}'}|_{\mathcal{E}_{\mathtt{G}}\cap\mathcal{E}_{\mathtt{G}'}} = \lambda_{\mathtt{G}}|_{\mathcal{E}_{\mathtt{G}}\cap\mathcal{E}_{\mathtt{G}'}}$.

  It is immediate to note that there exists $\mathtt{P}'$ such that $(\langle \mathsf{r}', \mathsf{v}''\rangle, \langle \mathtt{G}', \mathtt{P}'\rangle) \in \mathcal{I}$. Moreover, $\mathtt{G} \sqcup \mathtt{G}'$ is defined under the above conditions and for any $\mathtt{P}' \in \mathcal{S}_{\mathit{Ctr}}(\mathtt{G}')$ we have $\mathtt{P} \otimes \mathtt{P}' \subseteq \mathcal{S}_{\mathit{Ctr}}(\mathtt{G} \sqcup \mathtt{G}')$. Hence, it suffices to take $\mathtt{P}'' \in \mathtt{P} \otimes \mathtt{P}'$.
  Define $f'' : \mathcal{E}_{\mathtt{G}\sqcup\mathtt{G}'} \to \mathcal{R}$ as

$$f''(\mathsf{e}) = \begin{cases} f(\mathsf{e}) & \text{if } \mathsf{e} \in \mathcal{E}_{\mathtt{G}} \\ f'(\mathsf{e}) = \mathsf{r}_j & \text{otherwise} \end{cases}$$

  From the definition of $\mathtt{G}'$ and f'' it follows that for all $\mathsf{r} \in \mathcal{R}$

$$\#\{\mathsf{e} \mid \mathsf{e} \in \mathcal{E}_{\mathtt{G}\sqcup\mathtt{G}'} \text{ and } \lambda(\mathsf{e}) = \langle \mathtt{inc}, \mathtt{ok}\rangle\} = \mathit{max}\{\mathsf{v}(\mathsf{r}), \mathsf{v}''(\mathsf{r})\} = \mathsf{v}'(\mathsf{r})$$

  Consequently, $(\langle \mathsf{r}, \mathsf{v}'\rangle, \langle \mathtt{G} \sqcup \mathtt{G}', \mathtt{P}''\rangle) \in \mathcal{I}$.
- (SEND) Then, $\sigma = \langle \mathsf{r}, \mathsf{v}\rangle \xrightarrow{\mathtt{send},\langle \mathsf{r},\mathsf{v}\rangle} \langle \mathsf{r}, \mathsf{v}\rangle = \sigma'$. It follows immediately, because $(\langle \mathsf{r}, \mathsf{v}\rangle, \langle \mathtt{G}, \mathtt{P}\rangle) \in \mathcal{I}$ holds by hypothesis. □

Proof of Example 17. By case analysis on the applied rule for the derivation of $\sigma \xrightarrow{\alpha} \sigma'$. Consider $\sigma = \langle \mathsf{r}_j, \mathtt{V}, \mathtt{W}\rangle$ and $(\sigma, \langle \mathtt{G}, \mathtt{P}\rangle) \in \mathcal{I}$. Hence, there exists $f : \mathcal{E}_{\mathtt{G}} \to \mathcal{R}$ and

$$\forall \mathtt{k}.(\exists \mathsf{r}.\mathtt{W}(\mathtt{k}, \mathsf{r}) > 0 \iff \exists\, \mathtt{S}.\ \mathcal{S}_{\mathit{Or\text{-}Set}}(\mathtt{G}^{\langle \mathtt{lookup},\mathtt{S}\rangle}) \neq \emptyset \ \wedge \ \mathtt{k} \in \mathtt{S}) \tag{4}$$

- (LOOKUP) Hence, $\sigma \xrightarrow{\mathtt{lookup},\mathtt{S}} \sigma' = \sigma$. Define $\mathtt{G}' = \mathtt{G}^{\langle \mathtt{lookup},\mathtt{S}\rangle}$ and take $f' : \mathcal{E}_{\mathtt{G}'} \to \mathcal{R}$ defined as follows

$$f'(\mathsf{e}) = \begin{cases} f(\mathsf{e}) & \text{if } \mathsf{e} \in \mathcal{E}_{\mathtt{G}} \\ \mathsf{r}_j & \text{if } \mathsf{e} = \top \end{cases}$$

  Note that

$$\forall \mathtt{k}.(\exists \mathsf{r}.\mathtt{W}(\mathtt{k}, \mathsf{r}) > 0 \iff \exists\, \mathtt{S}.\ \mathcal{S}_{\mathit{Or\text{-}Set}}(\mathtt{G}'^{\langle \mathtt{lookup},\mathtt{S}\rangle}) \neq \emptyset \ \wedge \ \mathtt{k} \in \mathtt{S})$$

  follows from the definition of $\mathcal{S}_{\mathit{Or\text{-}Set}}$ and (4). Moreover, we conclude that $\mathtt{sat}(\mathtt{P}, \langle \mathtt{lookup}, \mathtt{S}\rangle) \subseteq \mathcal{S}_{\mathit{Or\text{-}Set}}(\mathtt{G}'^{\langle \mathtt{lookup},\mathtt{S}\rangle})$. Therefore, $(\sigma', \langle \mathtt{G}', \mathtt{P}'\rangle) \in \mathcal{I}$ for any $\mathtt{P}' \in \mathtt{sat}(\mathtt{P}, \langle \mathtt{lookup}, \mathtt{S}\rangle)$.

- (ADD) Then, $\sigma \xrightarrow{\mathtt{add}(\mathtt{k}'),\mathtt{ok}} \langle \mathsf{r}_j, \mathtt{V}[\mathsf{r}_j \mapsto \mathtt{V}(\mathsf{r}_j)+1], \mathtt{W}[(\mathtt{k}', \mathsf{r}_j) \mapsto \mathtt{V}(\mathsf{r}_j)+1]\rangle = \sigma'$. Define $\mathtt{G}' = \mathtt{G}^{\langle \mathtt{add}(\mathtt{k}'),\mathtt{ok}\rangle}$ and take $f' : \mathcal{E}_{\mathtt{G}'} \to \mathcal{R}$ defined as follows

$$f'(\mathsf{e}) = \begin{cases} f(\mathsf{e}) & \text{if } \mathsf{e} \in \mathcal{E}_{\mathtt{G}} \\ \mathsf{r}_j & \text{if } \mathsf{e} = \top \end{cases}$$

Now, we show that the following holds

$$\forall \mathtt{k}(\exists \mathsf{r}.\mathtt{W}[(\mathtt{k}', \mathsf{r}_j) \mapsto \mathtt{V}(\mathsf{r}_j)+1](\mathtt{k}, \mathsf{r}) > 0 \iff \exists\, \mathtt{S}.\, \mathcal{S}_{\mathit{Or\text{-}Set}}(\mathtt{G}'^{\langle \mathtt{lookup},\mathtt{S}\rangle}) \neq \emptyset \ \wedge\ \mathtt{k} \in \mathtt{S}) \tag{5}$$

For all $\mathtt{k} \neq \mathtt{k}'$, the condition holds by (4)

$$\mathtt{W}[(\mathtt{k}', \mathsf{r}_j) \mapsto \mathtt{V}(\mathsf{r}_j)+1](\mathtt{k}, \mathsf{r}) > 0 \iff \exists \mathtt{S}.\mathcal{S}_{\mathit{Or\text{-}Set}}(\mathtt{G}^{\langle \mathtt{lookup},\mathtt{S}'\rangle}) \neq \emptyset \ \wedge\ \mathtt{k} \in \mathtt{S}$$

Then, for $\mathtt{k} = \mathtt{k}'$ it is enough to take $\mathsf{r} = \mathsf{r}_j$ to conclude that the double implication holds.
By the definition of $\mathcal{S}_{\mathit{Or\text{-}Set}}$, we have that $\mathcal{S}_{\mathit{Or\text{-}Set}}(\mathtt{G}^{\langle \mathtt{lookup},\mathtt{S}'\rangle}) \neq \emptyset$ implies $\mathcal{S}_{\mathit{Or\text{-}Set}}(\mathtt{G}'^{\langle \mathtt{lookup},\mathtt{S}\cup\{\mathtt{k}'\}\rangle}) \neq \emptyset$. Additionally, $\mathtt{sat}(\mathtt{P}, \langle \mathtt{add}(\mathtt{k}), \mathtt{ok}\rangle) \subseteq \mathcal{S}_{\mathit{Or\text{-}Set}}(\mathtt{G}')$ because $\mathtt{P} \in \mathcal{S}_{\mathit{Or\text{-}Set}}(\mathtt{G})$. Then, for any $\mathtt{P}' \in \mathtt{sat}(\mathtt{P}, \langle \mathtt{add}(\mathtt{k}), \mathtt{ok}\rangle)$, we have $(\sigma', \langle \mathtt{G}', \mathtt{P}'\rangle) \in \mathcal{I}$.
- (REM) Then, $\sigma \xrightarrow{\mathtt{rem}(\mathtt{k}'),\mathtt{ok}} \langle \mathsf{r}_j, \mathtt{V}, \mathtt{W}'\rangle = \sigma'$, and $\forall \mathsf{s}.\mathtt{W}'(\mathtt{k}', \mathsf{s}) = 0$, and $\forall \mathsf{s}, \mathtt{k}'' \neq \mathtt{k}'.\mathtt{W}'(\mathtt{k}'', \mathsf{s}) = \mathtt{W}(\mathtt{k}'', \mathsf{s})$. Define $\mathtt{G}' = \mathtt{G}^{\langle \mathtt{rem}(\mathtt{k}'),\mathtt{ok}\rangle}$ and take $f' : \mathcal{E}_{\mathtt{G}'} \to \mathcal{R}$ defined as follows

$$f'(\mathsf{e}) = \begin{cases} f(\mathsf{e}) & \text{if } \mathsf{e} \in \mathcal{E}_{\mathtt{G}} \\ \mathsf{r}_j & \text{if } \mathsf{e} = \top \end{cases}$$

As in the previous case, we conclude that the following holds for all $\mathtt{k} \neq \mathtt{k}'$

$$\exists \mathsf{r}.\mathtt{W}'(\mathtt{k}, \mathsf{r}) > 0 \iff \exists \mathtt{S}.\mathcal{S}_{\mathit{Or\text{-}Set}}(\mathtt{G}'^{\langle \mathtt{lookup},\mathtt{S}\rangle}) \neq \emptyset \ \wedge\ \mathtt{k} \in \mathtt{S}$$

from (4) and the fact that $\mathtt{W}'(\mathtt{k}'', \mathsf{r}) = \mathtt{W}(\mathtt{k}'', \mathsf{r})$ holds for all $\mathtt{k}'' \neq \mathtt{k}'$ and r. By $\mathcal{S}_{\mathit{Or\text{-}Set}}$, $\mathcal{S}_{\mathit{Or\text{-}Set}}(\mathtt{G}'^{\langle \mathtt{lookup},\mathtt{S}\rangle}) \neq \emptyset$ implies $\{\mathtt{k}'\} \notin \mathtt{S}$. Then, the proof is completed by noting that $\forall \mathsf{r}.\mathtt{W}'(\mathtt{k}', \mathsf{r}) = 0$.
- (SEND) Then, $\sigma \xrightarrow{\mathtt{send},\sigma} \sigma$. It follows because $(\sigma, \langle \mathtt{G}, \mathtt{P}\rangle) \in \mathcal{I}$ by hypothesis.
- (RECEIVE) Then, $\sigma \xrightarrow{\mathtt{receive}\langle \mathsf{r}_k, \mathtt{V}', \mathtt{W}'\rangle} \langle \mathsf{r}_j, \max\{\mathtt{V}, \mathtt{V}'\}, (\mathtt{V}, \mathtt{W}) \oplus (\mathtt{V}', \mathtt{W}')\rangle = \sigma'$. Define $\mathtt{G}' = \langle \mathcal{E}_{\mathtt{G}'}, \prec_{\mathtt{G}'}, \lambda_{\mathtt{G}'}\rangle$ where
  - $\mathcal{E}_{\mathtt{G}'} = \biguplus_{\mathsf{r} \in \mathcal{R}, \mathtt{k} \in \mathbb{N}} \mathcal{F}_{\mathsf{r}}^{\mathtt{k}}$ (where $\uplus$ stands for disjoint union) and

$$\mathcal{F}_{\mathsf{r}}^{\mathtt{k}} = \begin{cases} \mathcal{E}_{\mathsf{r}}^{\mathtt{k}} & \text{if } \mathtt{W}'(\mathtt{k}, \mathsf{r}) \neq 0 \ \wedge\ \mathtt{W}'(\mathtt{k}, \mathsf{r}) \leq \mathtt{V}(\mathsf{r}) \quad \text{(i)} \\ \mathcal{E}_{\mathsf{r}}^{\mathtt{k}} \uplus \{\mathsf{a}_{\mathsf{r}}^{\mathtt{k}}\} & \text{if } \mathtt{W}'(\mathtt{k}, \mathsf{r}) \neq 0 \ \wedge\ \mathtt{W}'(\mathtt{k}, \mathsf{r}) > \mathtt{V}(\mathsf{r}) \quad \text{(ii)} \\ \mathcal{E}_{\mathsf{r}}^{\mathtt{k}} \uplus \{\mathsf{r}_{\mathsf{r}}^{\mathtt{k}}\} & \text{if } \mathtt{W}'(\mathtt{k}, \mathsf{r}) = 0 \ \wedge\ \mathtt{W}(\mathtt{k}, \mathsf{r}) \leq \mathtt{V}'(\mathsf{r}) \quad \text{(iii)} \\ \emptyset & \text{if } \mathtt{W}'(\mathtt{k}, \mathsf{r}) = 0 \ \wedge\ \mathtt{W}(\mathtt{k}, \mathsf{r}) > \mathtt{V}'(\mathsf{r}) \quad \text{(iv)} \end{cases}$$

$$\text{where } \mathcal{E}_{\mathsf{r}}^{\mathtt{k}} = \{\mathsf{e} \mid \mathsf{e} \in \mathcal{E}_{\mathtt{G}} \ \wedge\ f(\mathsf{e}) = \mathsf{r} \ \wedge\ \lambda_{\mathtt{G}}(\mathsf{e}) = \langle \mathtt{add}(\mathtt{k}), \mathtt{ok}\rangle\}$$

  - $\lambda_{\mathtt{G}'}$ is defined such that

$$\lambda_{\mathtt{G}'}(\mathsf{e}) = \begin{cases} \langle \mathtt{add}(\mathtt{k}), \mathtt{ok}\rangle & \text{if } \mathsf{e} \in \mathcal{F}_{\mathsf{r}}^{\mathtt{k}} \wedge \mathsf{e} \neq \mathsf{r}_{\mathsf{r}}^{\mathtt{k}} \\ \langle \mathtt{rem}(\mathtt{k}), \mathtt{ok}\rangle & \text{if } \mathsf{e} = \mathsf{r}_{\mathsf{r}}^{\mathtt{k}} \end{cases}$$

  - $\prec_{\mathtt{G}'}$ defined such that

$$\prec_{\mathtt{G}'}|_{\mathcal{F}_{\mathsf{r}}^{\mathtt{k}}} = \begin{cases} \prec|_{\mathcal{E}_{\mathsf{r}}^{\mathtt{k}}} & \text{if } \mathcal{F}_{\mathsf{r}}^{\mathtt{k}} = \mathcal{E}_{\mathsf{r}}^{\mathtt{k}} \\ \prec|_{\mathcal{E}_{\mathsf{r}}^{\mathtt{k}}} & \text{if } \mathcal{F}_{\mathsf{r}}^{\mathtt{k}} = \mathcal{E}_{\mathsf{r}}^{\mathtt{k}} \uplus \{\mathsf{a}_{\mathsf{r}}^{\mathtt{k}}\} \\ \prec|_{\mathcal{E}_{\mathsf{r}}^{\mathtt{k}}} \cup (\mathcal{E}_{\mathsf{r}}^{\mathtt{k}} \times \{\mathsf{r}^{\mathtt{k}}\}) & \text{if } \mathcal{F}_{\mathsf{r}}^{\mathtt{k}} = \mathcal{E}_{\mathsf{r}}^{\mathtt{k}} \uplus \{\mathsf{r}_{\mathsf{r}}^{\mathtt{k}}\} \\ \emptyset & \text{if } \mathcal{F}_{\mathsf{r}}^{\mathtt{k}} = \emptyset \end{cases}$$

Now we check that $(\langle \mathsf{r}_k, \mathtt{V}', \mathtt{W}'\rangle, \langle \mathtt{G}', \mathtt{P}'\rangle) \in \mathcal{I}$. By definition of $\mathcal{S}_{\mathit{Or\text{-}Set}}$ in Example 8, we may conclude that $\mathcal{S}_{\mathit{Or\text{-}Set}}(\mathtt{G}'^{\langle \mathtt{lookup},\mathtt{S}'\rangle}) \neq \emptyset$ whenever

$$\mathtt{S}' = \{\mathtt{k} \mid \exists \mathsf{e}' \in \mathcal{E}_{\mathtt{G}'}.\exists \mathtt{k} \in \mathbb{N}.\lambda_{\mathtt{G}'}(\mathsf{e}') = \langle \mathtt{add}(\mathtt{k}), \mathtt{ok}\rangle \text{ and } \forall \mathsf{e}''.\mathsf{e}' \prec_{\mathtt{G}'} \mathsf{e}'' \text{ implies } \lambda_{\mathtt{G}'}(\mathsf{e}'') \neq \langle \mathtt{rem}(\mathtt{k}), \mathtt{ok}\rangle\}$$

From the definition of $\mathtt{G}'$, it is immediate that $\mathtt{k} \in \mathtt{S}'$ iff $\exists \mathsf{r}.\mathtt{W}'(\mathtt{k}, \mathsf{r}) > 0$ (the cases (i) and (ii) on the definition of $\mathcal{F}_{\mathsf{r}}^{\mathtt{k}}$ apply). Then, for any $\mathtt{P}' \in \mathcal{S}_{\mathit{Or\text{-}Set}}(\mathtt{G}')$, we have $(\langle \mathsf{r}_k, \mathtt{V}', \mathtt{W}'\rangle, \langle \mathtt{G}', \mathtt{P}'\rangle) \in \mathcal{I}$ by taking $f'' : \mathcal{E}_{\mathtt{G}'} \to \mathcal{R}$ defined such that $f''(\mathsf{e}) = \mathsf{r}$ if there exists $\mathtt{k} \in \mathbb{N}$ s.t. $\mathsf{e} \in \mathcal{F}_{\mathsf{r}}^{\mathtt{k}}$.
Note that $\mathtt{G} \sqcup \mathtt{G}'$ is well-defined because $\mathtt{G}'$ is defined such that $\lambda_{\mathtt{G}'}|_{\mathcal{E}_{\mathtt{G}} \cap \mathcal{E}_{\mathtt{G}'}} = \lambda_{\mathtt{G}}|_{\mathcal{E}_{\mathtt{G}} \cap \mathcal{E}_{\mathtt{G}'}}$ and $\prec_{\mathtt{G}'}|_{\mathcal{E}_{\mathtt{G}} \cap \mathcal{E}_{\mathtt{G}'}} = \prec_{\mathtt{G}}|_{\mathcal{E}_{\mathtt{G}} \cap \mathcal{E}_{\mathtt{G}'}}$. Moreover, $\mathtt{P} \otimes \mathtt{P}' \neq \emptyset$ by the definition of $\mathtt{G}'$. It remains to prove that $(\sigma', \langle \mathtt{G} \sqcup \mathtt{G}', \mathtt{P}''\rangle) \in \mathcal{I}$ for any $\mathtt{P}'' \in \mathtt{P} \otimes \mathtt{P}'$.

Now define $f' : \mathcal{E}_{G \sqcup G'} \to \mathcal{R}$ such that

$$f'(e) = \begin{cases} f(e) & \text{if } e \in \mathcal{E}_G \\ f''(e) & \text{if } e \in \mathcal{E}_{G'} \end{cases}$$

Note that f' is well-defined because $\{\mathcal{F}'^{k}_{r}\}_{r \in \mathcal{R}, k \in \mathbb{N}}$ is a partition of $\mathcal{E}_{G'}$, and the definition of G' ensures that $\mathcal{E}^{k}_{r} \cap \mathcal{F}'^{k'}_{s} = \emptyset$ for all k, k' and $r \neq s$. Finally, we show that for all k, the following holds

$$\exists r.(V, W) \oplus (V', W')(k, r) > 0 \iff \exists\ S.\ \mathcal{S}((G \sqcup G')^{\langle \mathtt{lookup}, S \rangle}) \neq \emptyset \wedge k \in S$$

For $\Rightarrow$), assume that $(V, W) \oplus (V', W')(k, r) > 0$ for some r. By the definition of $\oplus$, $\neg(\textsc{IsREM}(W, W', V, k, r) \ \vee \textsc{IsREM}(W', W, V', k, r))$. Consequently,

$$\neg\ (\ (W(k, r) = 0 \ \wedge\ W'(k, r) \leq V(r)) \quad \vee \quad (W'(k, r) = 0 \ \wedge\ W(k, r) \leq V'(r))\ )$$

By rearranging terms,

$$(W(k, r) \neq 0 \wedge W'(k, r) \neq 0) \quad \vee \quad (W(k, r) \neq 0 \wedge W(k, r) > V'(r)) \quad \vee$$
$$(W'(k, r) > V(r) \wedge W'(k, r) \neq 0) \quad \vee \quad (W'(k, r) > V(r) \wedge W(k, r) > V'(r))$$

We proceed by case analysis:

a) $W(k, r) \neq 0 \wedge W'(k, r) \neq 0$. Since $(\langle r, V, W \rangle, \langle G, P \rangle) \in \mathcal{I}$, $W(k, r) > 0$ implies $\exists\ S.\ \mathcal{S}_{\textit{Or-Set}}(G^{\langle \mathtt{lookup}, S \rangle}) \neq \emptyset \ \wedge\ k \in S$. By the definition of $\mathcal{S}_{\textit{Or-Set}}$, $\exists e' \in \mathcal{E}_G$ such that $\lambda_G(e') = \langle \mathtt{add}(k), \mathtt{ok} \rangle$ and $\forall e''.e' \prec_G e''$ implies $\lambda_G(e'') \neq \langle \mathtt{rem}(k), \mathtt{ok} \rangle$. Moreover, since $W'(k, r) \neq 0$, the definition of G' is such that the cases (i) and (ii) in $\mathcal{F}'^{k_r}$ only apply. Consequently, $\forall e''.e' \prec_{G \sqcup G'} e''$ implies $\lambda_{G \sqcup G'}(e'') \neq \langle \mathtt{rem}(k), \mathtt{ok} \rangle$. Therefore, $\mathcal{S}_{\textit{Or-Set}}((G \sqcup G')^{\langle \mathtt{lookup}, S \rangle}) \neq \emptyset$ implies $k \in S$.

b) $W(k, r) \neq 0 \wedge W(k, r) > V'(r)$. If $W'(k, r) \neq 0$, then the proof follows as in the previous case. Otherwise $(W'(k, r) = 0)$, the only possible case in the definition of $\mathcal{F}'^{k}_{r}$ is (iv) and we reason analogously to the previous case to conclude that $\mathcal{S}_{\textit{Or-Set}}((G \sqcup G')^{\langle \mathtt{lookup}, S \rangle}) \neq \emptyset$ implies $k \in S$.

c) $W'(k, r) > V(r) \wedge W'(k, r) \neq 0$. Hence, only the case (ii) in the definition of $\mathcal{F}'^{k}_{r}$ applies. Consequently, for a^{k}_{r} it holds that $\forall e''.a^{k}_{r} \prec_{G \sqcup G'} e''$ implies $\lambda_{G \sqcup G'}(e'') \neq \langle \mathtt{rem}(k), \mathtt{ok} \rangle$. Therefore, $\mathcal{S}_{\textit{Or-Set}}(G^{\langle \mathtt{lookup}, S \rangle}) \neq \emptyset$ implies $k \in S$.

d) $W'(k, r) > V(r) \wedge W(k, r) > V'(r)$. Now note that $W'(k, r) > V(r)$ implies $W'(k, r) \neq 0$ and $W(k, r) > V'(r)$ implies $W(k, r) \neq 0$. Then, the proof follows as in case a).

Now, we consider the case in which $(V, W) \oplus (V', W')(k, r) = 0$ for all r. As before, we conclude that for all r, the following holds:

$$(W(k, r) = 0 \ \wedge\ W'(k, r) \leq V(r)) \quad \vee \quad (W'(k, r) = 0 \ \wedge\ W(k, r) \leq V'(r))$$

We proceed by case analysis for all r

a) $W(k, r) = 0 \wedge W'(k, r) \leq V(r)$. From $W(k, r) = 0$ and $(\langle r, V, W \rangle, \langle G, P \rangle) \in \mathcal{I}$ we deduce that there is not $e' \in \mathcal{E}_G$ such that $\lambda(e') = \langle \mathtt{add}(k), \mathtt{ok} \rangle$ and $\forall e''.e' \prec_G e''$ implies $\lambda(e'') \neq \langle \mathtt{rem}(k), \mathtt{ok} \rangle$. Moreover, the applicable cases in the definition of $\mathcal{F}'^{k}_{r}$ are (i), (iii), and (iv). Consequently, there is not $e' \in \mathcal{E}^{k}_{r} \cup \mathcal{F}'^{k}_{r}$ such that $\lambda_{G \sqcup G'}(e') = \langle \mathtt{add}(k), \mathtt{ok} \rangle$ and for all e'' if $e' \prec_{G \sqcup G'} e''$ then $\lambda_{G \sqcup G'}(e'') \neq \langle \mathtt{rem}(k), \mathtt{ok} \rangle$.

b) $W'(k, r) = 0 \wedge W(k, r) \leq V'(r)$. This case corresponds to (iii) in the definition of $\mathcal{F}'^{k}_{r}$. Hence, for all $e' \in \mathcal{E}^{k}_{r} \cup \mathcal{F}'^{k}_{r}$ such that $\lambda(e') = \langle \mathtt{add}(k), \mathtt{ok} \rangle$ we have that $e' \prec_{G \sqcup G'} r^{k}$. Consequently, there is not $e' \in \mathcal{E}^{k}_{r} \cup \mathcal{F}'^{k}_{r}$ such that $\lambda_{G \sqcup G'}(e') = \langle \mathtt{add}(k), \mathtt{ok} \rangle$ and for all e'' if $e' \prec_{G \sqcup G'} e''$ then $\lambda_{G \sqcup G'}(e'') \neq \langle \mathtt{rem}(k), \mathtt{ok} \rangle$.

Since the above two cases hold for all k and r, we conclude that

$$\forall r.(V, W) \oplus (V', W')(k, r) = 0 \Rightarrow \forall\ S.\ \mathcal{S}((G \sqcup G')^{\langle \mathtt{lookup}, S \rangle}) = \emptyset \vee k \notin S$$

For $\Leftarrow$), assume that there exists S and k such that $\mathcal{S}((G \sqcup G')^{\langle \mathtt{lookup}, S \rangle}) \neq \emptyset \wedge k \in S$. Then, there exists e' such that $\lambda_{G \sqcup G'}(e') = \langle \mathtt{add}(k), \mathtt{ok} \rangle$ and $\forall e''.e' \prec_{G \sqcup G'} e''$ implies $\lambda_{G \sqcup G'}(e'') \neq \langle \mathtt{rem}(k), \mathtt{ok} \rangle$. There are two cases:

a) $e' \in \mathcal{E}_G$. By definition of $G \sqcup G'$, $\lambda_G(e') = \langle \mathtt{add}(k), \mathtt{ok} \rangle$ and $\forall e''.e' \prec_G e''$ implies $\lambda_G(e'') \neq \langle \mathtt{rem}(k), \mathtt{ok} \rangle$. Since, $(\langle r, V, W \rangle, \langle G, P \rangle) \in \mathcal{I}$, then there exists r such that $W(k, r) > 0$. Consequently, $\neg\textsc{IsREM}(W, W', V, k, r)$ holds. Since $\forall e''.e' \prec_{G \sqcup G'} e''$ implies $\lambda_{G \sqcup G'}(e'') \neq \langle \mathtt{rem}(k), \mathtt{ok} \rangle$, we conclude that $\mathcal{F}'^{k}_{r}$ is such that either (i), (ii) or (iv) holds. For (i) or (ii), note that $W'(k, s) \neq 0$ and, hence, $\neg\textsc{IsREM}(W', W, V', k, r)$ holds. For (iii), $W(k, r) \leq V'(r)$ implies $\neg\textsc{IsREM}(W', W, V', k, r)$. Consequently, $\neg\textsc{IsREM}(W, W', V, k, r)$ and $\neg\textsc{IsREM}(W', W, V', k, r)$. Hence,

$$(V, W) \oplus (V', W')(k, r) = \max\{W(k, r), W'(k, r)\}$$

Since $W(k, r) > 0$, $(V, W) \oplus (V', W')(k, r) > 0$ holds.

b) $e' \notin \mathcal{E}_G$. Then, $e' \notin \mathcal{E}_{G'}$. The only possibility is $e' = a^{k}_{r}$ for some $\mathcal{F}'^{k}_{r}$ (case (ii)). Hence, $W'(k, r) \neq 0$ and $W'(k, r) > V(r)$. The case follows by noting that $W'(k, r) \neq 0$ implies $\neg\textsc{IsREM}(W', W, V', k, r)$ and $W'(k, r) > V(r)$ implies $\neg\textsc{IsREM}(W, W', V, k, r)$.

Now, consider that there exists $\mathtt{k}$ such that for all $\mathtt{S}$, if $\mathcal{S}((\mathtt{G} \sqcup \mathtt{G}')^{\langle \mathtt{lookup}, \mathtt{S} \rangle}) \neq \emptyset$ then $\mathtt{k} \notin \mathtt{S}$. Consequently, for all $\mathtt{e}' \in \mathcal{E}_{\mathtt{GLIG}'}$ if $\lambda_{\mathtt{GLIG}'}(\mathtt{e}') = \langle \mathtt{add}(\mathtt{k}), \mathtt{ok} \rangle$ then there exists $\mathtt{e}''$ such that $\mathtt{e}' \prec_{\mathtt{GLIG}'} \mathtt{e}''$ and $\lambda_{\mathtt{GLIG}'}(\mathtt{e}'') = \langle \mathtt{rem}(\mathtt{k}), \mathtt{ok} \rangle$. Then, $\mathcal{F}^{\mathtt{k}}_{\mathtt{r}}$ was obtained by using either (i), (iii) or (iv). In case (iii) is used, then $\mathtt{W}'(\mathtt{k}, \mathtt{r}) = 0$ and $\mathtt{W}(\mathtt{k}, \mathtt{r}) \leq \mathtt{V}'(\mathtt{r})$. Therefore, $\textsc{IsRem}(\mathtt{W}', \mathtt{W}, \mathtt{V}', \mathtt{k}, \mathtt{r})$ holds and $(\mathtt{V}, \mathtt{W}) \oplus (\mathtt{V}', \mathtt{W}')(\mathtt{k}, \mathtt{r}) = 0$. For cases (i) and (iv), we first note that $\mathtt{e}'' \in \mathcal{E}_{\mathtt{G}}$. Hence, for all $\mathtt{S}$, if $\mathcal{S}(\mathtt{G}^{\langle \mathtt{lookup}, \mathtt{S} \rangle}) \neq \emptyset$ then $\mathtt{k} \notin \mathtt{S}$. Since $(\langle \mathtt{r}, \mathtt{V}, \mathtt{W} \rangle, \langle \mathtt{G}, \mathtt{P} \rangle) \in \mathcal{I}$, we have $\mathtt{W}(\mathtt{k}, \mathtt{r}) = 0$ for all r. Additionally, (i) implies $\mathtt{W}'(\mathtt{k}, \mathtt{r}) \leq \mathtt{V}(\mathtt{r})$ and, hence, $\textsc{IsRem}(\mathtt{W}, \mathtt{W}', \mathtt{V}, \mathtt{k}, \mathtt{r})$ holds. Consequently, $(\mathtt{V}, \mathtt{W}) \oplus (\mathtt{V}', \mathtt{W}')(\mathtt{k}, \mathtt{r}) = 0$. Case (iv) also implies $\mathtt{W}'(\mathtt{k}, \mathtt{r}) = 0$. Hence, $\mathtt{max}\{\mathtt{W}(\mathtt{k}, \mathtt{r}), \mathtt{W}'(\mathtt{k}, \mathtt{r})\} = 0$ and $(\mathtt{V}, \mathtt{W}) \oplus (\mathtt{V}', \mathtt{W}')(\mathtt{k}, \mathtt{r}) = 0$.
Therefore, $(\langle \mathtt{r}, \mathtt{max}\{\mathtt{V}, \mathtt{V}'\}, (\mathtt{W}, \mathtt{V}) \oplus (\mathtt{W}', \mathtt{V}') \rangle, \langle \mathtt{G} \sqcup \mathtt{G}', \mathtt{P}'' \rangle) \in \mathcal{I}$ holds. □

Proof of Lemma 7. We show that the following relation satisfies Definition 21.

$$\mathcal{I} = \{(\sigma_1 \| \sigma_2, \langle \mathtt{G}_1 \sqcup \mathtt{G}_2, \mathtt{P} \rangle) \mid \sigma_1 \sim^{o}_{\mathcal{S}} \langle \mathtt{G}_1, \mathtt{P}_1 \rangle \text{ and } \sigma_2 \sim^{o}_{\mathcal{S}} \langle \mathtt{G}_2, \mathtt{P}_2 \rangle \text{ and } \mathtt{P} \in \mathtt{P}_1 \otimes \mathtt{P}_2\}$$

We proceed by case analysis on the transitions of $\sigma_1 \parallel \sigma_2$.

- (PARL). Since $\sigma_1 \sim^{o}_{\mathcal{S}} \langle \mathtt{G}_1, \mathtt{P}_1 \rangle$, we have that $\langle \mathtt{G}_1, \mathtt{P}|_{\mathcal{E}_{\mathtt{G}_1}} \rangle \xrightarrow{\ell} \langle \mathtt{G}'_1, \mathtt{P}'_1 \rangle$. Then, the case follows by Lemma 6.
- (COMM). Since $\sigma_1 \xrightarrow{\mathtt{send}, \sigma} \sigma'_1$ and $\sigma_1 \sim^{o}_{\mathcal{S}} \langle \mathtt{G}_1, \mathtt{P}_1 \rangle$, we have (i) $\sigma'_1 = \sigma_1$, (ii) $\sigma \sim^{o}_{\mathcal{S}} \langle \mathtt{G}', \mathtt{P}' \rangle$, (iii) $\mathtt{G}_1 \sqcup \mathtt{G}' = \mathtt{G}_1$, and (iv) $\mathtt{P}_1 \otimes \mathtt{P}' = \mathtt{P}_1$. Additionally, $\sigma_2 \xrightarrow{\mathtt{rcv}, \sigma} \sigma'_2$ implies there exist $\mathtt{G}''$, $\mathtt{P}''$ such that (v) $\sigma \sim^{o}_{\mathcal{S}} \langle \mathtt{G}'', \mathtt{P}'' \rangle$, (vi) $\mathtt{P}'_2 \in \mathtt{P}_2 \otimes \mathtt{P}''$ and (vii) $\sigma'_2 \sim^{o}_{\mathcal{S}} \langle \mathtt{G}_2 \sqcup \mathtt{G}'', \mathtt{P}'_2 \rangle$. From (ii), (v) and (vii) we conclude that $\sigma'_2 \sim^{o}_{\mathcal{S}} \langle \mathtt{G}_2 \sqcup \mathtt{G}', \mathtt{P}'_2 \rangle$. Moreover, $\mathtt{G}_1 \sqcup (\mathtt{G}_2 \sqcup \mathtt{G}') = \mathtt{G}_1 \sqcup \mathtt{G}_2$ because of (iii). Hence, $(\sigma'_1 \parallel \sigma'_2, \langle \mathtt{G}_1 \sqcup \mathtt{G}_2, \mathtt{P} \rangle) \in I$. □

Proof of Prop. 1. It follows as in the previous lemma by showing that $\sim^{m}_{\mathcal{S}}$ is preserved by parallel composition $\parallel$ of states. □

Proof of Lemma 8. Proof follows analogously to Lemma 7 by showing that the following relation satisfies Definition 21.

$$I = \{(\sigma_1 \mid_{\mathtt{B}} \sigma_2, \langle \mathtt{G}_1 \sqcup \mathtt{G}_2, \mathtt{P} \rangle) \mid \sigma_1 \sim^{o}_{\mathcal{S}} \langle \mathtt{G}_1, \mathtt{P}_1 \rangle \text{ and } \sigma_2 \sim^{o}_{\mathcal{S}} \langle \mathtt{G}_2, \mathtt{P}_2 \rangle \text{ and } \mathtt{P} \in \mathtt{P}_1 \otimes \mathtt{P}_2$$
$$\text{and } \forall \sigma \in \mathtt{B}. \exists \mathtt{G}. (\sigma \sim^{o}_{\mathcal{S}} \langle \mathtt{G}, \mathtt{P} \rangle \text{ and } (\mathtt{G}_1 \sqcup \mathtt{G}_2) \sqcup \mathtt{G} \text{ defined})\} \quad \square$$

Proof of Prop. 2. It follows as in the previous lemma by showing that $\sim^{m}_{\mathcal{S}}$ is preserved by parallel composition $\mid_{\mathtt{B}}$ of states. □

References

[1] S. Gilbert, N. Lynch, Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services, SIGACT News 33 (2) (2002) 51–59.
[2] M. Shapiro, N.M. Preguiça, C. Baquero, M. Zawirski, Conflict-free replicated data types, in: X. Défago, F. Petit, V. Villain (Eds.), SSS 2011, in: Lect. Notes Comput. Sci., vol. 6976, Springer, 2011, pp. 386–400.
[3] S. Burckhardt, A. Gotsman, H. Yang, M. Zawirski, Replicated data types: specification, verification, optimality, in: S. Jagannathan, P. Sewell (Eds.), POPL 2014, ACM, 2014, pp. 271–284.
[4] S. Burckhardt, Principles of eventual consistency, Found. Trends Program. Lang. 1 (1–2) (2014) 1–150.
[5] S. Burckhardt, A. Gotsman, H. Yang, Understanding eventual consistency, Tech. Rep. MSR-TR-2013-39, Microsoft Research, 2013.
[6] A. Gotsman, H. Yang, Composite replicated data types, in: J. Vitek (Ed.), ESOP 2015, in: Lect. Notes Comput. Sci., vol. 9032, Springer, 2015, pp. 585–609.
[7] A. Cerone, G. Bernardi, A. Gotsman, A framework for transactional consistency models with atomic visibility, in: L. Aceto, D. de Frutos-Escrig (Eds.), CONCUR 2015, in: LIPIcs, vol. 42, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2015, pp. 58–71.
[8] K.C. Sivaramakrishnan, G. Kaki, S. Jagannathan, Declarative programming over eventually consistent data stores, in: D. Grove, S. Blackburn (Eds.), PLDI 2015, ACM, 2015, pp. 413–424.
[9] A. Gotsman, H. Yang, C. Ferreira, M. Najafzadeh, M. Shapiro, Cause I'm strong enough: reasoning about consistency choices in distributed systems, in: R. Bodík, R. Majumdar (Eds.), POPL 2016, ACM, 2016, pp. 371–384.
[10] M. Wirsing, Algebraic specification, in: Handbook of Theoretical Computer Science, Volume B: Formal Models and Semantics (B), Elsevier, 1990, pp. 675–788.
[11] J.V. Guttag, E. Horowitz, D.R. Musser, The design of data type specifications, in: R.T. Yeh, C.V. Ramamoorthy (Eds.), ICSE 1976, IEEE Computer Society Press, 1976, pp. 414–420.
[12] H.-D. Ehrich, On the theory of specification, implementation, and parametrization of abstract data types, J. ACM 29 (1) (1982) 206–227.
[13] F. Gadducci, H.C. Melgratti, C. Roldán, A denotational view of replicated data types, in: J. Jacquet, M. Massink (Eds.), COORDINATION 2017, in: Lect. Notes Comput. Sci., vol. 10319, Springer, 2017, pp. 138–156.
[14] M. Shapiro, N. Preguiça, C. Baquero, M. Zawirski, A comprehensive study of convergent and commutative replicated data types, Tech. Rep. RR-7506, Inria-Centre Paris-Rocquencourt, 2011.
[15] D.B. Terry, M. Theimer, K. Petersen, A.J. Demers, M. Spreitzer, C. Hauser, Managing update conflicts in Bayou, a weakly connected replicated storage system, in: M.B. Jones (Ed.), SOSP 1995, ACM, 1995, pp. 172–183.
[16] R. Jagadeesan, J. Riely, From sequential specifications to eventual consistency, in: M.M. Halldórsson, K. Iwama, N. Kobayashi, B. Speckmann (Eds.), ICALP 2015, in: Lect. Notes Comput. Sci., vol. 9135, Springer, 2015, pp. 247–259.
[17] A. Bouajjani, C. Enea, J. Hamza, Verifying eventual consistency of optimistic replication systems, in: S. Jagannathan, P. Sewell (Eds.), POPL 2014, ACM, 2014, pp. 285–296.
[18] K. von Gleissenthall, A. Rybalchenko, An epistemic perspective on consistency of concurrent computations, in: P.R. D'Argenio, H.C. Melgratti (Eds.), CONCUR 2013, in: Lect. Notes Comput. Sci., vol. 8052, Springer, 2013, pp. 212–226.
[19] P. Zeller, A. Bieniusa, A. Poetzsch-Heffter, Formal specification and verification of CRDTs, in: E. Ábrahám, C. Palamidessi (Eds.), FORTE 2014, in: Lect. Notes Comput. Sci., vol. 8461, Springer, 2014, pp. 33–48.