Replicated Database Systems under the Crash-Recovery Failure Model: Correctness Conditions for 1-Copy Equivalence

J. R. Juárez-Rodríguez[1], I. Arrieta-Salinas[1], J. R. González de Mendívil[1], F. D. Muñoz-Escoí[2]

[1]Depto. de Ing. Matemática e Informática
Univ. Pública de Navarra
31006 Pamplona, Spain

[2] Inst. Univ. Mixto Tecnológico de Informática
Univ. Politécnica de Valencia
46022 Valencia, Spain

{jr.juarez,itziar.arrieta,mendivil}@unavarra.es fmunyoz@iti.upv.es

Technical Report TR-ITI-SIDI-2011/002

Replicated Database Systems under the Crash-Recovery Failure Model: Correctness Conditions for 1-Copy Equivalence

J. R. Juárez-Rodríguez[1], I. Arrieta-Salinas[1], J. R. González de Mendívil[1], F. D. Muñoz-Escoí[2]

[1]Depto. de Ing. Matemática e Informática
Univ. Pública de Navarra
31006 Pamplona, Spain

[2] Inst. Univ. Mixto Tecnológico de Informática
Univ. Politécnica de Valencia
46022 Valencia, Spain

Technical Report TR-ITI-SIDI-2011/002

e-mail: {jr.juarez,itziar.arrieta,mendivil}@unavarra.es fmunyoz@iti.upv.es

April 12, 2011

Abstract

The correct design and implementation of replicated database servers is a challenging task. A replication protocol has to ensure consistency among different data replicas even in case of failure, as well as to retransmit lost data if a replica recovers after a failure. One-copy equivalence is the traditional criterion to study the correctness of this kind of systems. Formal tecniques using simple and modular models are necessary to provide clear specifications of distributed data systems, although they are rarely used. In fact, up to our knowledge, no previous work has tried to gather multiple isolation levels, integrity constraints, crash failures and recovery in a same formal model. This paper presents a general framework for modelling and verifying database replication schemes under the crash-recovery failure model that covers a wide range of protocols. Using this model, we introduce a one-copy equivalence notion, in which transactions can be executed under different isolation levels, and we provide some criteria to simplify the task of proving this equivalence for full replicated protocols. Additionally, these criteria are also applicable to partial replication protocols.

1 Introduction

In the last years, high availability and load balancing in database servers have become increasingly important. The challenge in the design of these systems is to keep multiple copies of data items in different physical locations despite failures. Most replication techniques usually follow a Read One Write All (ROWA) approach: transactions read from one replica, whereas their write operations have to be applied at all replicas. The primary copy approach is the easiest replication strategy, in which a replica, called primary, is in charge of executing all the update transactions and the rest of replicas are used as backups that can only perform read-only transactions. In contrast, the update everywhere strategy allows to perform updates at any copy but makes coordination more complex. Several protocols are based on deferred-update replication: each update transaction executes first all its operations in one replica, called delegate, and then the write operations are propagated to all replicas.

In order to ensure consistency among replicas, transaction executions have to be synchronized. Latest solutions of database replication [1, 9, 14, 17, 18, 19, 21, 25, 29, 33, 40] make use of Group Communication Systems (GCS) to order transactions. These systems allow to keep track of the active replicas and to communicate using a variety of reliability and ordering properties, which are very useful for replication and fault-tolerance.

When a server site fails, clients can redirect their transactions to another available one, but possibility of recovering sites that have failed must exist. The recovery task basically consists in transferring the information lost during the failure interval, from one or more active replicas to one or more recovering sites, without interfering with the normal service of the system.

Motivation. When developing crash-recovery applications in a distributed system, its correctness must be proven somehow. The correctness of replicated database systems is traditionally linked to the notion of one-copy equivalence [8], since a user should see a replicated system working as a single database. The main idea is that every transaction in the replicated system behaves as if it had been executed in a logical copy of the database maintaining its isolation level and respecting the integrity constraints in case it is committed. Currently, there exist different definitions of the one-equivalence concept depending on the concrete isolation level considered, such as one-copy serializability [8] or one-copy snapshot isolation [25]. Moreover, many one-copy equivalence definitions do not consider any liveness properties, which entails that if no events occurred in the system, it would be considered as correct.

The adoption of formal design techniques to provide precise specifications of the problem, their solutions and proof of correctness is an important issue to be addressed, more so when the vast number of algorithms and protocols for database replication shows us the great complexity of the problem. It is highly desirable to use a simple and compact mathematical model to provide clear specifications and formal verifications of the critical properties of distributed systems. Besides, system specifications should be clear and modular enough to describe precisely all the aspects of a complex distributed system. However, formal methods and modular descriptions are rarely used in replicated database systems.

Several solutions have been proposed to provide data replication. These solutions usually assume only a single isolation level and do not manage integrity constraints [2, 13, 14, 22, 25, 28, 29, 34]; a few works support multiple isolation levels, but again without considering integrity constraints [6, 37]; and a last set considers integrity constraints, but only related to either a replication protocol family [31, 35] or a single isolation level [24]. Some of these works [2, 24, 25, 33] consider also the effect of crash failures, but only a few of them presents a strict formal model to reason about them. Besides, other works [5, 11, 18, 20] deal with how to recover from replica failures, although they simply propose some recovery mechanisms and do not provide any formal reasoning about their correctness.

Up to our knowledge, no previous work has tried to gather all these details together in a formal model. Furthermore, many correctness proofs either do not consider failures or no strict formal proofs are presented. Besides, the correctness of the recovery process is normally presented as a patch over the proof for the replication protocol, although the recovery should be actually integrated into the replication protocol. Usually, stronger conditions are assumed to simplify the study of failures, instead of considering the peculiarities of the recovery process from the start.

Contributions. This paper tries to fill this void by presenting a general model for a replicated database system under the crash-recover failure model that supports both multiple isolation levels and integrity constraints. The model considers a distributed system where each site has a local DBMS that guarantees the *ACID* properties [16]. A *full replication* model is followed; i.e., each replica holds a full copy of the database and all replicas share the same schema. Transactions may be started at any time and any site, and may read or write any item. In this model, transactions may be executed under any isolation level supported by the DBMSs. Thus, applications may combine sets of transactions with different isolation levels. Moreover, databases admit the declaration of integrity constraints that must be respected by committed transactions. Furthermore, this general model accommodates any kind of replication protocol that fulfills a very simple set of properties, which most existing replication approaches actually do. With the aim to provide a rigorous and modular specification, we make use of the I/O automaton model [26, 27].

Using this general model, we provide the one-copy equivalence notion based on a legal relation that transforms the behaviors of the replicated system, in which a transaction is executed in different sites, into a one-copy behavior, in which the transaction appears to be executed in a single database. Thus, to prove the correctness of a particular solution, we only need to check whether it is possible to find this relation. However, proving the correctness of a particular system in this way can be an arduous task; but it can be simplified if some simple properties that the system behaviors have to fulfill are provided.

Thus, in order to narrow the gap between the implementation of replication and recovery protocols and their formal proofs, this paper proposes a set of four correctness criteria, which are proven to be the necessary and sufficient conditions to be imposed on a replicated database system (using ROWA replication)

supporting the crash-recover model for achieving one-copy equivalence. These criteria, consisting of both safety and liveness properties, can serve as a basis for formally proving the correctness of protocols under the considered model in a modular and well-defined way. These criteria require that: (*C1 - local transaction progress*) every transaction that starts in its delegate site eventually gives a termination response (committed or aborted) at some site of the system unless that site crashes; (*C2 - uniform decision*) if a transaction is committed (aborted) at one site, then it cannot be aborted (committed) at other sites; (*C3 - uniform prefix order consistency*) for every two distinct sites, the sequence of committed update transactions at one site is a prefix of the sequence of the committed update transactions at the other site or vice versa providing that the writesets of remote and local transactions are the same; and (*C4 - non-contradiction*) if a remote transaction is committed, then it does not conflict with any of the transactions that were committed at its delegate replica between the beginning and commitment of it. Although they are quite intuitive, they have never been formalized or proved as valid for such a general model of data replication.

Related works. Previous works with a similar aim have not reached the level of generality proposed in this paper. Some works [3, 24] based on histories of operations executed on SI databases provide some sufficient conditions for a replicated history of a ROWA system to be one-copy equivalent. Additionally, in [24], integrity constraints and crash failures are also considered. However, failures are studied as a separate case of the replication process and hence the replication and recovery processes are considered individually although they are very interrelated. Another work [38] based on TLA+ [23] provides a general serializable database specification to study the serializability of deferred-update protocols. Thus, they propose an abstract algorithm that makes it easy to think about sufficient and necessary requirements for them to work correctly. Their system model is based on serializable databases and it does not consider either integrity constraints nor crash failures. Finally, in [2], we tried to propose some criteria to achieve the one-copy equivalence of a replicated database system, considering crash failures from the beginning, but the system model was based on SI replicas and it did not consider integrity constraints. Nevertheless, these criteria failed again to discuss the uncertainty of the crash failure despite its importance, which requires every remote transaction not to conflict with any of the transactions that were committed at its delegate replica.

Structure of the paper. The rest of this paper is organized as follows. Section 2 introduces the specification framework used throughout this work. Section 3 explains the basic definitions for understanding single database systems by means of the specification of a generic single database system with no failures. In Section 4, the previous system is extended to support the crash-recover failure model. Section 5 presents the specification of an abstract replicated database system with the crash-recover model. Section 6 formalizes the notion of one-copy equivalence. Section 7 is devoted to the correctness criteria that must be imposed in order to achieve one-copy equivalence, along with the formal proof that shows that such criteria are the necessary and sufficient conditions for obtaining one-copy equivalence. Section 8 extends the model presented in previous sections to include partial replication. Finally, Section 9 presents some concluding remarks.

2 Specification Framework

This paper makes use of the *I/O automaton model* [27] with the aim to provide a rigorous framework. In order to promote a modular design, each component is modeled as an I/O automaton module [27]. Each module M is specified by its external signature $sig(M)$ and a set of behaviors $behs(M)$ delimited by safety and liveness properties.

The signature of a module M consists of two different kinds of actions that allow M to communicate with other modules: input actions ($in(M)$) and output actions ($out(M)$). Thus, $sig(M) = (in(M), out(M))$. The set comprising all the possible actions of M is $acts(M) = in(M) \cup out(M)$.

An infinite (finite) behavior β of a module M is denoted by $\beta = \pi_1 \cdot \pi_2 ... \pi_m ...$ ($\beta = \pi_1 \cdot \pi_2 ... \pi_m$) with $\pi_i \in acts(M)$. The set of all acceptable behaviors of M is denoted by $behs(M)$. We say that π_i *is in* β if the i-th event in β is π_i, and that π *is in* β if there exists an index k such that $\pi_k = \pi$ and π_k is in β. For any $0 \leq j \leq |\beta|$ (where $|\beta|$ stands for the length of β), $\beta(j)$ represents the finite prefix (denoted '$\preceq$') of length j of β, i.e., $\beta(j) \preceq \beta$ and $|\beta(j)| = j$. By its definition, $\beta(0) =$ empty. Moreover, we define a function named $end()$ that returns the last element of a given sequence, e.g., $end(\beta(j)) = \pi_j$.

Let $\varphi \subseteq acts(M)$, $\beta|\varphi$ is the subsequence of β including only the actions of φ in β, i.e., $\beta|\varphi = \pi_{i_1} \cdot \pi_{i_2} ... \pi_{i_k} ...$ such that π_{i_k} is in β and $\pi_{i_k} \in \varphi$. Note that we can also use the original indexes of the actions of β when describing the sequence of actions of $\beta|\varphi$. When $\varphi = acts(M')$ for a module M', we simply write $\beta|M'$.

In this paper, a replicated system is represented as the composition of a set of compatible modules [27]. A composition operation of several modules M_i whose signatures are compatible results in a module $M = \Pi_i(M_i)$ which has a signature composed by the set of M_i signatures and a set of behaviors $behs(M)$ such that each behavior $\beta \in behs(M)$ satisfies that $\beta|M_i \in behs(M_i)$, i.e., the behavior of the composition satisfies the properties of each of its components.

On the other hand, if a module M is a more detailed refinement of another module M', M must satisfy M' in the sense that $sig(M)$=$sig(M')$ and $behs(M)$⊆$behs(M')$. Thus, the properties satisfied by M' will also be satisfied by M.

Finally, although some variables used in the properties of the behaviors may be unbounded, it is understood that they are universally quantified in their domains for the entire scope of the formulas, unless we explicitly specify them for a better comprehension.

3 Database System Model

This section presents the specification of a generic single database system with no failures, which is modeled by means of a module denoted by $DB(\mathcal{I}, \mathcal{T})$. This module is needed for introducing some basic definitions and notations which are used throughout this work.

3.1 Database Transactions

A *database* consists of a set of items that can be accessed by concurrent transactions. Let $\mathcal{I}$ be the set of database items. The identifier of each item is assumed to be unique. The set of possible values for each item $x \in \mathcal{I}$ is represented by V_x.

Let $\mathcal{T}$ be the set of all possible transaction identifiers in any module, where the identifier of each transaction is assumed to be unique. A transaction $t \in \mathcal{T}$ is a sequence of read and write operations over the database items, starting with a $begin$ operation (denoted by $B(t)$) and ending with an $abort$ or $commit$ operation. Each operation *op* is actually a ($request\_op$,$response\_op$) pair. The response of a commit operation corresponding to a transaction t is either a $committed$ or an $aborted$ notification ($C(t)$ and $A(t)$ respectively), whereas the response of an $abort$ operation (i.e., a rollback request) always reports an $aborted$ notification.

If t completes a write operation on an item x by setting its value to v and is committed afterwards, a new version (x, v, t) is installed on the database. Thus, a *version* (x, v, t) relates an item $x \in \mathcal{I}$ to the value $v \in V_x$ installed by a committed transaction $t \in \mathcal{T}$. Let $\mathcal{V}$ the set of all possible versions of the database. For each item $x \in \mathcal{I}$, its initial version is the first version installed by the first committed transaction creating it. The model assumes that several versions of the same data item can be available in the database. Therefore, when a transaction completes a read operation on an item x, it can get any version $(x, v, t') \in \mathcal{V}$ previously installed by t'.

An execution of a set of concurrent transactions over the database is usually represented by an interleaved sequence of completed transaction operations. In order to delimit which of the executions are the possible valid executions on the database, some obligations have to be imposed to define that set of correct executions [8, 32]. However, from a mathematical perspective, it is also possible to provide a complete description of the execution without specifying the sequence of all individual operations for each transaction in the execution. To this end, we can include certain parameters in the notification of commit and abort operations, so as to obtain a complete semantic description of each transaction. More precisely, each transaction t that commits in a certain behavior must specify its readset (i.e. the set of versions read by t) and its writeset (i.e. the set of versions installed by t). A transaction is said to be read-only in a behavior β if its writeset is an empty set throughout β; otherwise, it is called an update transaction. Moreover, databases require a certain amount of information to establish whether a transaction can be committed or not. This control information may be the readset or the writeset themselves, it may be inferred from these sets (e.g.

the items of the writeset) or it may even be related with other information of the execution. To represent this information in a general way, we define a set $\mathcal{E}$ that contains all the possible control information elements, which will depend on the kind of control information used.

Thus, the notification of a committed transaction t taking place in an execution is now denoted by $C(t,d)$, where d includes the sets $d.rs \in 2^{\mathcal{V}}$, $d.ws \in 2^{\mathcal{V}}$ and $d.inf \in \mathcal{E}$ (which stand for the readset, writeset and control information used for establishing why t has been committed). In the following, $\mathcal{D} = 2^{\mathcal{V}} \times 2^{\mathcal{V}} \times \mathcal{E}$; hence $d \in \mathcal{D}$.

It would also be possible to change the aborted notification $A(t)$ by $A(t,c)$ where parameter c would explain the abort cause of t. However, it is unnecessary in this paper.

3.2 A Generic Single Database Module

The module $DB(\mathcal{I},\mathcal{T})$ is defined by its external action signature and the set of its possible behaviors, $behs(DB(\mathcal{I},\mathcal{T}))$. We omit $(\mathcal{I},\mathcal{T})$ when it is clear in the context. The external signature of DB is defined in such a way that $in(DB)$ is arbitrary and $\{B(t), C(t,d), A(t) : t \in \mathcal{T}, d \in \mathcal{D}\} \subseteq out(DB)$. By means of $B(t)$, the DB module notifies the beginning of a new transaction t. Actions $C(t,d)$ and $A(t)$ represent the database's final decision on the transaction effects.

In the following, we present the main definitions used for restricting the set of possible behaviors of the rest of modules specified in this paper.

Definition 3.1. (Well-formedness)

- *A transaction $t \in \mathcal{T}$ is said to be well-formed in a behavior $\beta \in behs(DB)$ if the sequence $\beta|\{B(t), A(t), C(t,d) : d \in \mathcal{D}\}$ is a prefix of one of the following sequences: $B(t) \cdot C(t,d)$ for some $d \in \mathcal{D}$, or $B(t) \cdot A(t)$.*
- *A behavior $\beta \in behs(DB)$ is said to be well-formed if every $t \in \mathcal{T}$ is well-formed in β.*

If a behavior $\beta \in behs(DB)$ is well-formed, then after the beginning of a transaction it can only be either committed or aborted, and such actions can only appear at most once in β.

The database specification is based on the concept of committed state, also called snapshot. A snapshot provides a view of the installed versions of the database items existing at a certain time in a behavior.

To determine the versions that comprise the snapshot, we define the log of a behavior for a certain set of items. Given a writeset $ws \in 2^{\mathcal{V}}$, and a subset of items $\mathcal{I}_k \subseteq \mathcal{I}$, $ws(\mathcal{I}_k)$ represents the elements of ws related to the items of $\mathcal{I}_k$, i.e. $ws(\mathcal{I}_k) = \{(x,v,t) : (x,v,t) \in ws \wedge x \in \mathcal{I}_k\}$. The log of DB for a subset of items I_k is defined as follows.

Definition 3.2. (Log of DB for $\mathcal{I}_k$) *Let β be a well-formed behavior of $DB(\mathcal{I},\mathcal{T})$ and $\mathcal{I}_k$ be a subset of $\mathcal{I}$. For each prefix $\beta(j)$ of β, with $0 \leq j \leq |\beta|$, the log of $\beta(j)$ for the subset of items $\mathcal{I}_k$ is defined as $log(\beta(j),\mathcal{I}_k) = log(\beta(j-1),\mathcal{I}_k) \cdot \langle d.ws(\mathcal{I}_k)\rangle \Leftrightarrow (\pi_j = C(t,d) \wedge d.ws(\mathcal{I}_k) \neq \emptyset \wedge j > 0)$. Otherwise, $log(\beta(j),\mathcal{I}_k) = log(\beta(j-1),\mathcal{I}_k)$, being $log(\beta(0),\mathcal{I}_k) = empty$.*

When $\mathcal{I}_k = \mathcal{I}$, $log(\beta(j),\mathcal{I})$ represents the sequence of writesets that were installed by the all the committed update transactions in $\beta(j)$. We can omit the $\mathcal{I}_k$ parameter when representing the log for the whole set of items $\mathcal{I}$, i.e., $log(\beta(j)) = log(\beta(j),\mathcal{I})$.

By making use of the previous definition, the database snapshot is defined as the set of the latest installed versions existing at a certain time in a behavior.

Definition 3.3. (Database Snapshot) *Let β be a well-formed behavior of DB. For each prefix $\beta(j)$ of β, with $0 \leq j \leq |\beta|$, the snapshot of $\beta(j)$ is defined as $\mathcal{S}(\beta(j)) = \bigcup_{x \in \mathcal{I}} end(log(\beta(j),\{x\}))$*[1].

Legal Database Behaviors. A database management system must guarantee all the ACID properties [15] for each transaction. The log represents the set of versions that have been persistently installed on the database, which can be seen as an abstraction of data durability. In order to guarantee atomicity, the model establishes that aborted transactions must never interfere with committed transactions, i.e., the operations

[1]For simplicity reasons, it is assumed that $end(log(\beta(j),\{x\}))$ can be interpreted as a set. Thus, $end(log(\beta(j),\{x\}))$ will only contain the latest version of item x, regardless of the number of versions of x installed by a given writeset.

of aborted transactions are appropriately rolled back. The presented definitions satisfy atomicity. In addition, real database management systems admit the definition of a variety of isolation levels under which transactions can be executed, and it is also possible to specify a whole range of integrity constraints to maintain data consistency. Instead of assuming a specific isolation level for each transaction, the presented database model considers weak conditions from which multiple isolation levels can be derived (within the limits of the proposed mathematical formulation). The definitions of predicates $compatible()$, $isolated()$ and $consistent()$ allow us to achieve this degree of generality.

First, we define the semantic data of a transaction t in a behavior β by means of the following function, which specifies the point at which the readset, writeset and control information of a transaction in a certain behavior become meaningful and never change from then on.

Definition 3.4. *Let $\delta : \mathcal{T} \times behs(DB) \rightarrow \mathcal{D}$. Given a well-formed behavior $\beta \in behs(DB)$ and a transaction $t \in \mathcal{T}$, then $\delta(t, \beta) = d$ if and only if $C(t, d)$ is in β. Otherwise, $\delta(t, \beta)$ is undefined.*

Definition 3.5. *Let β be a well-formed behavior of DB, $t', t \in \mathcal{T}$ be two transactions and i, j be two indexes of β such that $0 \leq i < j \leq |\beta|$:*

- $compatible(t, i, \beta(j)) \Rightarrow$
  $\delta(t, \beta(j)).rs \subseteq (\bigcup_{i \leq k \leq j} \mathcal{S}(\beta(k))) \cup \delta(t, \beta(j)).ws$
- $isolated(t', t, i, \beta(j)) \equiv \exists k\colon i < k < j\colon \pi_k = C(t', d') \Rightarrow P(d'.inf, \delta(t, \beta(j)).inf)$
- $consistent(t, \beta(j)) \equiv \forall z\colon K_z(\mathcal{S}(\beta(j-1)), \delta(t, \beta(j)).ws)$ *where $K_z()$ is an integrity constraint defined in the database.*

Predicate $compatible(t, i, \beta(j))$ shows that the versions that can belong to the readset $\delta(t, \beta(j)).rs$ must have been installed on the database between indexes i and j of β or must be its own writes $\delta(t, \beta(j)).ws$.

On the other hand, $isolated(t', t, i, \beta(j))$ determines the conditions that may happen in the context of a transaction t between indexes i and j of β with regard to another transaction t' that may be concurrently committed in that context. If those conditions happen, then t is able to reach the committed status (see Definition 3.6 below). By its definition, if t' is not committed between i and j, then it will never conflict with t. Note that, in this case, the predicate becomes true. If this happens for every transaction $t' \in \mathcal{T}$, it entails that t has been executed completely isolated from the rest of transactions between i and j. Otherwise, the control information of the involved transactions will determine if their isolation level permits them to be concurrently committed, by means of $P(d'.inf, \delta(t, \beta(j)).inf)$ in $isolated(t', t, i, \beta(j))$.

Finally, $consistent(t, \beta(j))$ holds if and only if the writeset of t, $\delta(t, \beta(j)).ws$, does not infringe any integrity constraint demanded by the database at j. Each constraint depends on the previous committed state (snapshot) of the database and the writeset to be installed.

By making use of the aforementioned predicates, Definition 3.6 provides the obligations for every committed transaction in a *legal behavior*.

Definition 3.6. (Legal Behavior) *A well-formed behavior $\beta \in behs(DB)$ is legal, if for each $t \in \mathcal{T}$ such that $\pi_i = B(t)$ and $\pi_j = C(t, d)$ are in β, the following conditions hold:*

(a) $compatible(t, i, \beta(j))$
(b) $isolated(t', t, i, \beta(j))$, *for all $t' \in \mathcal{T}$*
(c) $consistent(t, \beta(j))$

Thus, Definition 3.6 establishes that in a legal behavior, if a transaction t is committed: (a) its readset is obtained from the committed states seen within its context; (b) there is no other transaction t' conflicting with t; and (c) all the integrity constraints hold at the time the transaction is committed.

Table 1 shows several isolation levels that can be defined as particular cases of Definition 3.6.

The following example displays a legal behavior with concurrent transactions executed under different isolation levels.

Example 1. *Let us consider four update transactions $\{t_1, t_2, t_3, t_4\}$ such that t_1 is executed under Weak Read Committed, t_2 under Snapshot Isolation, t_3 under Dynamic-Serializable and t_4 under Serial. Assuming that transactions satisfy all the integrity constraints and are compatible, one of the possible behaviors could be, as shown in Figure 1:*

	$P(d'.inf, d.inf)$ in $isolated(t', t, i, \beta(j))$	$compatible(t, i, \beta(j))$
Weak Read Committed [7]	true	$d.rs \subseteq \bigcup_{i \leq k \leq j} \{\mathcal{S}(\beta(k))\} \cup d.ws$
Snapshot Isolation [14, 25]	$items(d'.ws) \cap items(d.ws) = \emptyset$	$d.rs \subseteq (\mathcal{S}(\beta(i)) \cup d.ws)$
Dynamic-Serializable [14]	$items(d'.ws) \cap (items(d.ws) \cup items(d.rs)) = \emptyset$	$d.rs \subseteq (\mathcal{S}(\beta(i)) \cup d.ws)$
Serial	false	$d.rs \subseteq (\mathcal{S}(\beta(i)) \cup d.ws)$

Table 1: Predicates depending on the isolation level considered (where $\pi_i = B(t)$, $\pi_j = C(t, d)$ and $\pi_k = C(t', d')$ are in β with $i < k < j$)

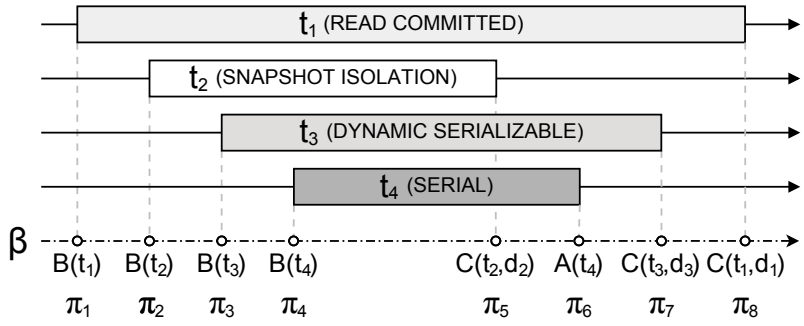

Figure 1: Example of a behavior with transactions executed concurrently under different isolation levels.

$\triangleright\ \beta = B(t_1) \cdot B(t_2) \cdot B(t_3) \cdot B(t_4) \cdot C(t_2, d_2) \cdot A(t_4) \cdot C(t_3, d_3) \cdot C(t_1, d_1)$

Since transaction t_1 is executed under Weak Read Committed, its $isolated$ predicate is always true. As for t_2, which is executed under Snapshot Isolation, $isolated(t', t_2, 2, \beta(5))$ is also true, because there is no transaction t' that commits during its execution. Transaction t_3, which runs under Dynamic-Serializable, would make $isolated(t_2, t_3, 3, \beta(7))$ false in case $items(d_2.ws) \cap (items(d_3.ws) \cup items(d_3.rs)) \neq \emptyset$, since $\pi_5 = C(t_2, d_2)$. However, we can infer that there is no such intersection as both t_2 and t_3 are committed. Finally, t_4 can never be commited, as $isolated(t_2, t_4, 4, \beta(6))$ is false because $\pi_5 = C(t_2, d_2)$.

3.3 Generalized Legal Behavior

The definition for legal behaviors can be generalized in a simple way to make it suitable for replicated settings. In a generalized legal behavior, a transaction is allowed to perform operations with stale information about database versions, as if it had been started before the time it actually did. This idea was originally introduced by [14] for Snapshot Isolation under the name of Generalized Snapshot Isolation. We extend this notion to make it valid under other isolation levels.

Definition 3.7. (Generalized Legal Behavior) *A well-formed behavior $\beta \in behs(DB)$ is a generalized legal behavior, if for each $t \in \mathcal{T}$ such that $\pi_i = B(t)$ and $\pi_j = C(t)$ are in β, there exists $0 \leq s \leq i$ such that the following conditions hold:*

(a) $compatible(t, s, \beta(j))$
(b) $isolated(t', t, s, \beta(j))$, *for all* $t' \in \mathcal{T}$
(c) $consistent(t, \beta(j))$

4 A Crash-Recovery Database Module

The generic module $DB(\mathcal{I}, \mathcal{T})$ can be adapted to support the crash-recovery failure model. The $DB^{CR}(\mathcal{I}, \mathcal{T})$ module, defined in Figure 2, has $crash$ and $recover$ as input actions. The new system is crash-prone: it

may fail and stop its execution at any time. Furthermore, the system may recover after crashing and resume the execution of transactions.

- Signature:
  $in(DB^{CR}) = \{crash, recover\}$
  $out(DB^{CR}) = \{B(t), C(t,d), A(t) : t \in \mathcal{T}, d \in \mathcal{D}\}$
- A set $behs(DB^{CR})$ of behaviors such that every $\beta \in behs(DB^{CR})$ is *well-formed* (Definition 3.1), *legal* (Definition 3.6), *progressive* (Property 4.1), and fulfills *execution integrity* (Property 4.2).

Figure 2: Module $DB^{CR}(\mathcal{I}, \mathcal{T})$

Apart from being well-formed and legal, the behaviors of DB^{CR} must fulfill the following liveness property, which states that if a transaction begins and does not provide any output, then the system must have crashed.

Property 4.1. (Progressive Behavior) *Every behavior $\beta \in behs(DB^{CR})$ is progressive: for every $t \in \mathcal{T}$, $\pi_i = B(t) \Rightarrow \exists j : j > i : \pi_j \in \{C(t,d), A(t), crash : d \in \mathcal{D}\}$.*

Moreover, we need to adequately model the behavior of actions $crash$ and $recover$, as specified in Property 4.2.

Property 4.2. (Execution Integrity) *Every $\beta \in behs(DB^{CR})$ holds the two following conditions:*
(a) $\pi_j = crash \Rightarrow (\beta = \beta(j) \vee \pi_{j+1} = recover)$
(b) $\pi_j = recover \Rightarrow \pi_{j-1} = crash$

Property 4.2 ensures that: (a) after a $crash$ action, either the system stops its activity and no further actions are performed, or a $recover$ action is executed after the system crashed to restart the system; and (b) the immediate predecessor of a $recover$ action is always a $crash$ action. Thus, no actions can occur between a $crash$ and a $recover$ action.

5 A Replicated Database System

This section provides the specification of an abstract replicated database system supporting the crash-recovery failure model, named RS. The components of the RS module are a group of databases $DB_n^{CR}(\mathcal{I}, \mathcal{T})$, being $n \in \mathcal{N}$ (where $\mathcal{N} = \{1..N\}$ represents the set of site identifiers). All the databases in the system have the same set of items $\mathcal{I}$ and the same set of values V_x for each item $x \in \mathcal{I}$, as well as the same integrity constraints. Thus, they all have the same set of possible versions $\mathcal{V}$ for the set of transactions $\mathcal{T}$. Under these conditions, *full database replication* is assumed. From this point onwards, we omit $\mathcal{I}$ and $\mathcal{T}$ in $DB_n^{CR}(\mathcal{I}, \mathcal{T})$.

The DB_n^{CR} module is the same as the DB^{CR} module of Section 4, with the only difference that, since the DB_n^{CR} is intended for replicated settings, its specification is subject to its site identifier, i.e., the actions of its signature are labeled with the site identifier. Thus, $in(DB_n^{CR})$=$\{crash_n, recover_n\}$; $out(DB_n^{CR})$=$\{B_n(t), C_n(t,d), A_n(t) : t \in \mathcal{T}, d \in \mathcal{D}\}$; and every $\beta \in behs(DB_n^{CR})$ is well-formed, legal, progressive and fulfills execution integrity.

The RS module results from the module composition [27] of the group of DB_n^{CR} modules: $RS = \Pi_{n \in \mathcal{N}} DB_n^{CR}$. Therefore, the signature of RS has $in(RS) = \left(\bigcup_{n \in \mathcal{N}} in(DB_n^{CR})\right)$ and $out(RS) = \left(\bigcup_{n \in \mathcal{N}} out(DB_n^{CR})\right)$ as input and output actions.

The signature of the RS module is well-defined, as the signatures of the component modules are compatible [27]. Moreover, by the definition of module composition, every $\beta \in behs(RS)$ satisfies $\beta | DB_n^{CR} \in behs(DB_n^{CR})$.

Several works [10, 29, 36] point out the convenience of providing databases with extended features to simplify the replication task. These features are modeled by the DB_n^{CR} in an abstract way, regardless of their implementation. Thus, in the resulting RS, the database at each site handles transactions with independence from the rest of sites. As depicted in Figure 3, a replication protocol will be responsible

for coordinating them adequately, and a recovery protocol will manage sites that recover from a crash, so that they can catch up with the rest of sites. By leaving the replication and recovery protocols in charge of these tasks, the RS focuses on the valid behaviors that transactions can have, and hides the distribution and communication issues that depend on the concrete implementation of the protocols.

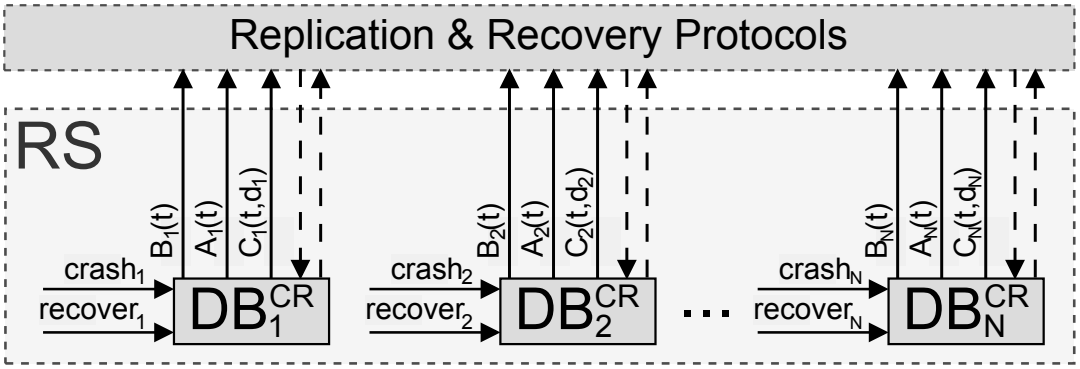

Figure 3: Replicated Database System. Solid arrows indicate the observable actions of this model, whereas dashed arrows represent hidden actions.

This paper focuses on protocols in which each transaction starts at a certain system site, called *delegate site* of that transaction, which is in charge of executing all read operations. In contrast, the effects of write operations of committed transactions must be applied at all sites (which entails that the writeset of each transaction t is the same at all sites that commit t). Similarly, the control information of a transaction in a certain behavior is the same at all sites that commit that transaction.

Transactions are said to be *local transactions* at their respective delegate sites, and *remote transactions* at the other sites. Each local transaction and all its associated remote transactions share the same transaction identifier. In order to distinguish between local and remote transactions, we assume a function $site : \mathcal{T} \rightarrow \mathcal{N}$ such that $site(t)$ (the delegate site of t) is unique: $(site(t) = n \wedge site(t) = n') \Leftrightarrow n = n'$. There are no further assumptions restricting the way in which local transactions can appear in the system; therefore, they may begin anytime at any site and read/write any item under any isolation level.

We define a new module, named RS^P, which is a refinement of the RS. Hence, $sig(RS^P) = sig(RS)$ and $behs(RS^P) \subseteq behs(RS)$. The behaviors of the RS^P module are restricted by Property 5.1, which establishes the conditions that model the considered replication protocols. This property states that a remote transaction can begin only after the corresponding local transaction began at the delegate site (thus avoiding the spontaneous creation of remote transactions). Moreover, the semantic data regarding a transaction is the same at all sites in which it is committed, with the exception of the readset (which is empty at remote sites).

Property 5.1. (P: Protocol Abstraction) *For every behavior $\beta \in behs(RS^P)$ and every transaction $t \in \mathcal{T}$, the following conditions hold:*

(a) $\pi_i = B_n(t) \Rightarrow \exists j : j \leq i : \pi_j = B_{site(t)}(t)$
(b) $\pi_i = C_n(t, d) \wedge \pi_j = C_{n'}(t, d') \Rightarrow d.ws = d'.ws \wedge d.inf = d'.inf$
(c) $\pi_i = C_n(t, d) \wedge n \neq site(t) \Rightarrow d.rs = \emptyset$

We now define the semantic data of a transaction t in a behavior of the RS^P, which is the union of the semantics of t at all system sites.

Definition 5.1. *Given a behavior $\beta \in behs(RS^P)$ and a transaction $t \in \mathcal{T}$, $\delta(t, \beta).rs = \bigcup_{n \in \mathcal{N}}(\delta(t, \beta | DB_n^{CR}).rs)$, $\delta(t, \beta).ws = \bigcup_{n \in \mathcal{N}}(\delta(t, \beta | DB_n^{CR}).ws)$ and $\delta(t, \beta).inf = \bigcup_{n \in \mathcal{N}}(\delta(t, \beta | DB_n^{CR}).inf)$.*

The following remark allows us to ignore read-only transactions for the rest of the paper.

Remark 5.1 (Read-only transactions)**.** *If t is a read-only transaction, by its definition, if $C_{site(t)}(t, d)$ is in a behavior of the RS^P, it holds that $d.ws = \emptyset$ (therefore it does not appear at the log of $site(t)$). Moreover,*

by Property 5.1(b-c), for any $n \neq site(t)$, if $C_{site(t)}(t, d')$ is in a behavior of the RS, then $d'.rs = d'.ws = \emptyset$. However, in order for a read-only transaction t to be purely local, $isolated(t, t', i, \beta(j))$ must be true for any t'. Assuming that read-only transactions never conflict with other transactions, no control information has to be sent to remote transactions, thus they can be executed locally. Consequently, read-only transactions can be ignored. From now on, we consider that every transaction $t \in \mathcal{T}$ is an update transaction.

The RS^P is just the composition of $N \geq 1$ databases with a basic property to model the replication protocol abstraction. Since there are not any other global restrictions, any pattern is possible, e.g., although a remote transaction is committed, its local transaction may be aborted or may not give any response because of a crash. Therefore, other global conditions must be demanded in order to obtain a correct replicated database system.

6 One-Copy Equivalence

The correctness criterion commonly used to prove that a replicated database system works correctly is the one-copy equivalence notion [8]. Following this notion, a committed or aborted transaction is also committed or aborted in the one-copy database, but if no response is produced for a transaction which started in the replicated system because of a crash, then the same happens in the one-copy database. In this paper, the one-copy database attempts to provide an explanation for each transaction in a behavior of the replicated system.

6.1 The 1CDB Module

The $1CDB$ module is defined in Figure 4. This module bears some similarity to the DB^{CR} module presented in Section 3. However, as the $1CDB$ module represents a system with N replicas, instead of the execution integrity specified by Property 4.2, the behaviors of the $1CDB$ module must satisfy Property 6.1, which models the occurrence of $crash$ and $recover$ actions. It ensures that the number of $crash$ actions cannot be lower than the number of $recover$ actions in any prefix of a behavior (i.e. every $recover$ must have a matching $crash$ action that happened before, to represent that a replica recovers after having crashed). Moreover, Property 6.1 states that the number of $crash$ actions without a corresponding $recover$ cannot be higher than N (as there can be at most N crashed replicas), and that when there are N $crash$ actions without a matching $recover$, the $1CDB$ module cannot perform any more actions unless the next action is a $recover$ (thus representing that all replicas are crashed and one of them recovers, although in the $1CDB$ model there is no notion of system sites).

- Signature:
  $in(1CDB) = \{crash, recover\}$
  $out(1CDB) = \{B(t), A(t), C(t, d) \colon t \in \mathcal{T}, d \in \mathcal{D}\}$
- A set $behs(1CDB)$ such that every $\gamma \in behs(1CDB)$ is a *well-formed* (Definition 3.1), *generalized legal* (Definition 3.7), *progressive* (in the sense given in Property 4.1), and *N-crash-stop* (Property 6.1) behavior.

Figure 4: Module 1CDB

Property 6.1. (N-crash-stop Behaviors) *For every behavior $\gamma \in behs(1CDB)$, being $\#crash(\gamma(j)) = |(\gamma(j)|\{crash\})|$ and $\#recover(\gamma(j)) = |(\gamma(j)|\{recover\})|$, for any $j : 0 \leq j \leq |\gamma|$, the following conditions hold:*

- $0 \leq \#crash(\gamma(j)) - \#recover(\gamma(j)) \leq N$
- $\#crash(\gamma(j)) - \#recover(\gamma(j)) = N \Rightarrow (\pi_{j+1} = recover \vee \gamma = \gamma(j))$

Note also that the $1CDB$ module considers generalized legal behaviors instead of legal behaviors. This is due to the fact that a transaction that begins in a site may not see the most current versions of the database

items in the whole system, and the transaction may be committed locally by working with stale versions of those items.

The $1CDB$ module is the most abstract specification of a correct replicated database system. It does not impose any conditions regarding its implementability. For instance, there are no restrictions on the number of sites that may be simultaneously crashed. The additional conditions imposed on replicated systems are necessary to achieve one-copy equivalence, but are not part of the one-copy model itself.

6.2 One-copy Equivalence Definition

Given a list of conditions φ, the RS^P_φ module is defined as $sig(RS^P_\varphi)$=$sig(RS^P)$ and $behs(RS^P_\varphi)$=$\{\beta : \beta \in behs(RS^P)$ and β satisfies all the conditions in $\varphi\}$. Any RS^P_φ is called a (refined) module of the RS^P, since $behs(RS^P_\varphi) \subseteq behs(RS^P)$.

The RS^P_φ module represents all possible systems that can be built using the RS^P module as a basis. We now have to state when an RS^P_φ module is correct according to the notion of one-copy equivalence. In an RS^P_φ module, a transaction $t \in T$ may appear in $\beta \in behs(RS^P_\varphi)$ as either a local or a remote transaction. However, in the $1CDB$ module each $t \in T$ can only appear once without making reference to any site. Thus, it is necessary to relate the actions of a transaction t in both modules and also their semantics. Consequently, in order to study the one-copy equivalence we have to define a relation between the behaviors of an RS^P_φ and the $1CDB$.

Definition 6.1. (Legal Relation) *Let RS^P_φ be a refined module of RS^P. Let Γ be a relation in $behs(RS^P_\varphi)$ $\times behs(1CDB)$. Γ is a legal relation if for each $\beta \in behs(RS^P_\varphi)$ there exists at least one $\gamma \in behs(1CDB)$ such that:*

(1) $\delta(t, \beta) = \delta(t, \gamma)$
(2) $\exists n \in \mathcal{N}$: $B_n(t)$ *is in* $\beta \Leftrightarrow B(t)$ *is in* γ
(3) $\exists n \in \mathcal{N}$: $C_n(t, d)$ *is in* β *for some* $d \in \mathcal{D} \Leftrightarrow C(t, d')$ *is in* γ *for some* $d' \in \mathcal{D}$.
(4) $\exists n \in \mathcal{N}$: $A_n(t)$ *is in* $\beta \Leftrightarrow A(t)$ *is in* γ
(5) $|(\beta|\{crash_n : n \in \mathcal{N}\})| = |(\gamma|\{crash\})|$
(6) $|(\beta|\{recover_n : n \in \mathcal{N}\})| = |(\gamma|\{recover\})|$

Note that, according to Definition 6.1(1), the semantic data of a given transaction in behaviors β and γ is the same. By the definition of $\delta(t, \beta)$ and Property 5.1, it holds that $\delta(t, \beta).ws = d.ws$ and $\delta(t, \beta).inf = d.inf$ in case $C_n(t, d)$ exists in β. Moreover, if $n = site(t)$, $\delta(t, \beta).rs = d.rs$. On the other hand, if a transaction t is committed in γ, $C(t, d')$ will happen only once in γ, as it is well-formed. Thus, d' must contain all the semantic data of t, i.e. $\delta(t, \gamma) = d'$.

To define the relation in a more general way, Definition 6.1 permits to choose arbitrarily the order of the actions in γ regardless of the order established by β for these actions. By its definition, the image of β by the legal relation Γ, denoted $\Gamma(\beta)$, satisfies $\Gamma(\beta) \subseteq behs(1CDB)$. Since transactions in $\gamma \in \Gamma(\beta)$ have the same readset, writeset and control information as in β, as well as the same isolation level as in β, this legal relation can be somehow considered a general one-copy equivalence notion between a system characterized by $behs(RS^P_\varphi)$ and the $1CDB$ module. This allows us to define the one-copy equivalence between an RS^P_φ module and the $1CDB$ module.

Definition 6.2. (One-Copy Equivalence) *Let RS^P_φ be a refined module of RS^P. The RS^P_φ module is one-copy equivalent to the $1CDB$ module if and only if there exists a legal relation $\Gamma \subseteq behs(RS^P_\varphi) \times behs(1CDB)$.*

Figure 5 shows an example of a behavior β of an RS_φ module and a behavior γ of the $1CDB$ module obtained from β. It is worth noting that a transaction t that begins in a site n that crashes before producing a response for t can be committed (or aborted) after that transaction recovers (as in Figure 5, where site 1 crashes before executing $C_1(t_1, d^1_1)$). In fact, when a site n crashes before executing the $C_n(t, d)$ action for a pending transaction t, the model does not distinguish between the case when t was not committed in n before it crashed and the case when t was actually committed but n crashed before producing a response. The recovery protocol will be in charge of appropriately finishing transactions that were active before the

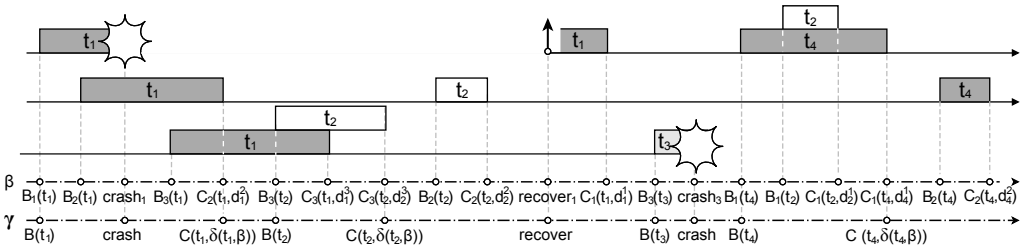

Figure 5: Example of a behavior $\beta \in behs(RS^P)$ and its corresponding one-copy equivalent behavior $\gamma \in \Gamma(\beta)$.

site crashed (e.g. if messages corresponding to pending transactions are persistently delivered [30], the recovery protocol will be able to apply transactions that were delivered at the replica but were not applied before crashing, by accessing the queue of delivered messages).

7 Correctness Criteria

This section details the set of conditions φ that must be fulfilled by the behaviors of the RS^P_φ module to provide one-copy equivalence (see Definition 6.2). As it will be proven, the proposed conditions are necessary and sufficient to guarantee one-copy equivalence. For this reason, they are correctness criteria.

Criterion 1 (C1: Local Transaction Progress)**.** *Let $\beta \in behs(RS^P)$, it holds that $\pi_i = B_{site(t)}(t) \Rightarrow \exists\, n\colon n \in \mathcal{N}\colon \exists\, k\colon k > i\colon \pi_k \in \{C_n(t,d),\, A_n(t),\, crash_n : d \in \mathcal{D}\}$*

Criterion C1 indicates that if a transaction begins, then it will be committed or aborted at least at one site, or that site will crash otherwise. This liveness criterion entails that if a transaction begins and does not provide any output then there must have been at least one crash in the system. Note that every formal specification requires such kind of properties, since a system in which nothing happens is always safe.

Criterion 2 (C2: Non-Contradictory Decision)**.** *Let $\beta \in behs(RS^P)$, it holds that $\neg(\pi_i = C_n(t,d) \wedge \pi_j = A_{n'}(t))$ in β, for any $t \in \mathcal{T}$, $d \in \mathcal{D}$ and $n, n' \in \mathcal{N}$.*

Criterion C2 states that if a transaction is committed (aborted) at one site, either correct or faulty, it cannot be aborted (committed) at any site. Thus, C2 guarantees that the decision on the outcome of a transaction is not contradictory in the system.

Criterion 3 (C3: Uniform Prefix Order Consistency)**.** *Let $\beta \in behs(RS^P)$. For every $\beta(j) \preceq \beta$, it holds that $log(\beta(j)|DB^{CR}_n) \preceq log(\beta(j)|DB^{CR}_{n'})$ or vice versa.*

Criterion C3 forces the system to build the same snapshots at all the databases. If a database fails, this criterion ensures that the last installed snapshot is also a valid snapshot for the rest of correct sites.

When it comes to considering crash failures, the previous criteria may not prevent some undesirable behaviors. For example, if a local transaction t was going to be aborted locally by another conflicting transaction, but the delegate site crashed before producing the abort notification, a remote transaction of t may be committed at other site. Criterion C4 avoids such behaviors by ensuring that the behavior of remote transactions is equivalent to that of their local transactions even if the local ones fail to notify their termination. To simplify the formulation, we define $last(i, n, \beta)$ as the last transaction that committed in a site n before an action π_i in a behavior $\beta \in behs(RS^P)$.

Definition 7.1 (Last Transaction)**.** *Let β be a behavior of RS^P. The last committed transaction of β at a site n before π_i is defined as $last(i, n, \beta) = t_{last}$ if and only if $\exists\, j\colon j < i\colon \pi_j = C_n(t_{last}, d)$ (for some $d \in \mathcal{D}$) $\wedge \forall\, k\colon j < k < i\colon \pi_k \notin \{C_n(t, d')\colon t \in T, d' \in \mathcal{D}\}$. Otherwise, $last(i, n, \beta) = f_0$.*

By Definition 7.1, either there exists a transaction t_{last} which is the last committed one just before action π_i in β, or there does not exist a previous committed transaction yet. In the latter case, in order to

simplify some of the proofs, we assume that for all $n \in \mathcal{N}$, if $\pi_j = C_n(f_0)$ then $j = 0$; i.e., a fictitious transaction f_0 has been committed at every site at the initial point.

Criterion 4 (C4: Remote Equivalence)**.** *Let $\beta \in behs(RS^P)$, it holds that $\pi_i = B_{site(t)}(t) \wedge \pi_j = C_{n'}(last(i, site(t), \beta), d') \wedge \pi_k = C_{n'}(t, d) \wedge n' \neq site(t) \Rightarrow \forall\, t'' \in T\colon\ isolated(t'', t, j, \beta(k) | DB^{CR}_{n'})$.*

It is worth noting that Criterion C4 is only necessary when isolation levels may cause conflicts.

We denote by RS^P_{CC} the module RS^P_φ whose behaviors satisfy C1-C4. Such criteria are proven to be the necessary and sufficient conditions that must be imposed on the RS^P to be one-copy equivalent to the $1CDB$ module in the next subsections.

7.1 Proof of Necessity

In order to study whether Criteria C1-C4 are necessary conditions to get the $1CDB$ equivalent system, we assume the existence of a legal relation Γ such that a refined module of RS^P is one-copy equivalent to $1CDB$, and prove by contradiction that such equivalence is not possible when supposing that each criterion does not hold.

Theorem 7.1. *Let RS^P_φ be a refined module of RS^P. If either $C1$, C2, C3 or $C4$ does not hold for $behs(RS^P_\varphi)$, then RS^P_φ is not one-copy equivalent to $1CDB$.*

Proof. We prove that when any of the correctness criteria does not hold, there exists no legal relation Γ (so by Definition 6.2 RS^P_φ is not one-copy equivalent to $1CDB$), i.e. there is at least one $\beta \in behs(RS^P_\varphi)$ such that any γ obtained using the conditions of Definition 6.1 from any possible relation Γ satisfies $\gamma \notin behs(1CDB)$.

• If C1 does not hold, then there exists $\beta \in behs(RS^P_\varphi)$ such that for some $t \in T\colon\ \pi_i = B_{site(t)}(t) \wedge \forall\, n \in \mathcal{N}\colon \forall\, k\colon k > i\colon \pi_k \notin \{C_n(t,d),\, A_n(t),\, crash_n : d \in \mathcal{D}\}$. Since Γ is a legal relation, every $\gamma \in \Gamma(\beta)$ holds that $\gamma | \{C_n(t,d),\, A_n(t),\, crash_n : d \in \mathcal{D}\} = B(t)$. Thus, $\gamma \notin behs(1CDB)$ because it is not progressive with regard to t, so there cannot exist a legal relation Γ, i.e. C1 is necessary.

• In case C2 does not hold, then there exists $\beta \in behs(RS^P_\varphi)$ such that for some transaction $t \in T$ and some $d \in \mathcal{D}$, $\pi_i = C_n(t,d)$ and $\pi_j = A_{n'}(t)$ are in β, with $n \neq n'$. For any $\gamma \in \Gamma(\beta)$, $\gamma | \{B(t), C(t,d), A(t) : d \in \mathcal{D}\}$ is either $B(t) \cdot C(t,d) \cdot A(t)$ or $B(t) \cdot A(t) \cdot C(t,d)$ for some $d \in \mathcal{D}$; therefore, γ is not well-formed. Thus, C2 is a necessary condition.

• If C3 does not hold, then there exists a finite $\beta \in behs(RS^P_\varphi)$ such that $log(\beta | DB^{CR}_n, \mathcal{I}) \not\preceq log(\beta | DB^{CR}_{n'}, \mathcal{I})$ with $n \neq n'$. Let us consider the β of Figure 6.

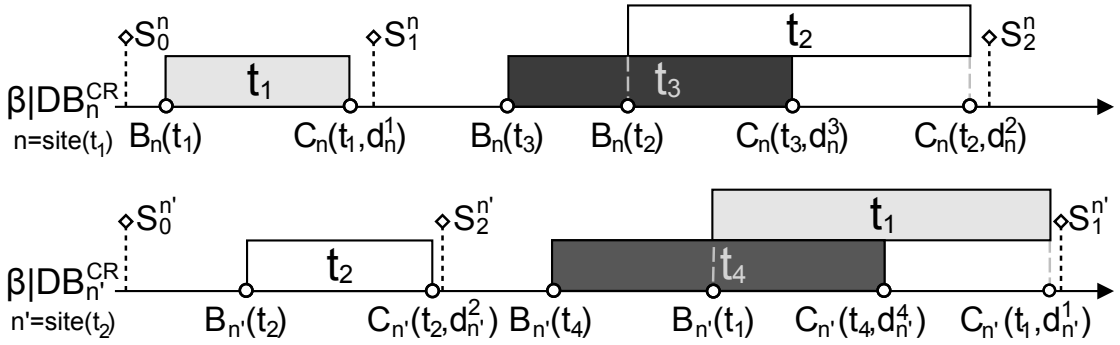

Figure 6: Example of a behavior in which transactions are committed in different order at two sites.

According to the system model, transactions can be executed under any isolation level and they can read/write any item at any time. Then, we establish the following additional conditions to the considered behavior β:

(1) $d^1_n.rs \subseteq S^n_0$ and $d^2_{n'}.rs \subseteq S^{n'}_0$ with $S^n_0 = S^{n'}_0$

(2) $items(d^1_n.ws) \nsubseteq items(d^2_{n'}.ws)$ and $items(d^2_{n'}.ws) \nsubseteq items(d^1_n.ws)$

(3) $d_n^3.rs \subseteq S_1^n$ and $d_{n'}^4.rs \subseteq S_2^{n'}$
(4) $items(d_n^3.rs) \cap items(d_n^1.ws) \neq \emptyset \wedge items(d_n^3.rs) \cap items(d_{n'}^2.ws) \neq \emptyset$
(5) $items(d_{n'}^4.rs) \cap items(d_n^1.ws) \neq \emptyset \wedge items(d_{n'}^4.rs) \cap items(d_{n'}^2.ws) \neq \emptyset$

Recall that, by Property 5.1, $d_n^1.ws = d_{n'}^1.ws$ and $d_n^2.ws = d_{n'}^2.ws$. Any possible $\gamma \in behs(1CDB)$ considering the transformation of Definition 6.1 produces one of the snapshot sequences $S_2^{n'} \cdot S_1^{n'}$ or $S_1^n \cdot S_2^n$. Since $S_1^n \neq S_1^{n'}$ and $S_2^{n'} \neq S_2^n$, for any $\gamma' \in \Gamma(\beta)$ that could be built including t_3 and t_4 over γ, either t_3 is incompatible or t_4 is incompatible and therefore $\gamma' \notin behs(1CDB)$.

• As Criteria C1-C3 have already been proven to be necessary conditions, any RS_φ^P which is one-copy equivalent to $1CDB$ must fulfill $\varphi \supseteq \{C1,\ C2,\ C3\}$. Thus, we can consider a one-copy equivalent system that fulfills these properties. First, we strengthen C1 by considering the following progress condition:

(C'1: Transaction Progress) For every $\beta \in behs(RS^P)$: $\pi_i = B_n(t) \Rightarrow \exists k : k > i : \pi_k \in \{C_n(t,d),\ A_n(t),\ crash_n : d \in \mathcal{D}\}$

Let $RS_{\{C'1,C2,C3\}}^P$ be a module RS^P whose behaviors satisfy Criteria C'1, C2 and C3. The $RS_{\{C'1,C2,C3\}}^P$ module satisfies Criteria C1-C3 (as C'1 implies C1 trivially). Let us assume that that $RS_{\{C'1,C2,C3\}}^P$ is one-copy equivalent to $1CDB$ and that C4 does not hold for $RS_{\{C'1,C2,C3\}}^P$. Therefore, there exists $\beta \in behs(RS_{\{C'1,C2,C3\}}^P)$ (and $\beta \in behs(RS_{\{C1,C2,C3\}}^P)$) such that $\pi_i = B_{site(t)}(t) \wedge \pi_j = C_{n'}(last(i, site(t), \beta), d') \wedge \pi_k = C_{n'}(t,d) \wedge site(t) \neq n' \wedge \exists t'' \in T\colon \neg isolated(t'', t, j, \beta(k)|DB_{n'}^{CR})$. The following proof shows that this is a contradiction, since every $\beta \in behs(RS_{\{C'1,C2,C3\}}^P)$ fulfills C4 when for any $n \in \mathcal{N}$ there is no action $crash_n$ in β.

By C'1, $\exists\, s\colon s > i\colon \pi_s \in \{C_{site(t)}(t,d), A_{site(t)}(t) : d \in \mathcal{D}\}$ in β. By C2: $\pi_s = C_{site(t)}(t,d)$ since $\pi_k = C_{n'}(t,d')$.

By its definition there exists a $\pi_l = C_{site(t)}(last(i, site(t), \beta), d_{last})$. Since every $\beta|DB_{site(t)}^{CR}$ is legal, $\forall\, t'' \in T\colon isolated(t'', t, i, \beta(s)|DB_{site(t)}^{CR})$, and then $\forall\, t'' \in T\colon isolated(t'', t, l, \beta(s)|DB_{site(t)}^{CR})$. By C3, $log(\beta(k)|DB_{n'}^{CR}, \mathcal{I}) = log(\beta(s)|DB_{site(t)}^{CR}, \mathcal{I})$.

If there exists a transaction $t_m \in T$ such that $l < m < s$ and $\pi_m = C_{site(t)}(t_m, d_m)$, then there exists an m' such that $j < m' < k$ and $\pi_{m'} = C_{n'}(t_m, d'_m)$. Recall that $d_m.ws = d'_m.ws$ and $d_m.inf = d'_m.inf$ for every site. Thus, if $isolated(t_m, t, j, \beta(k)|DB_{n'}^{CR})$ is false, then a contradiction is obtained because $isolated(t_m, t, l, \beta(s)|DB_{site(t)}^{CR})$ is true.

Thus, if C4 does not hold, then $behs(RS_{\{C'1,C2,C3\}}^P)$ is not one-copy equivalent to $1CDB$, and neither is $behs(RS_{\{C1,C2,C3\}}^P)$. As a consequence, C4 is a necessary condition for the behaviors of RS_φ^P to be one-copy equivalent to $1CDB$.

□

Remark 7.1. *Although Criterion C1 is the weakest liveness condition for a correct system, the* RS^P *module actually fulfills Criterion* $C'1$*, which is stronger than C1, due to the fact that the behaviors of every* DB_n^{CR} *are progressive in the sense of Property 4.1.*

7.2 Proof of Sufficiency

In order to prove that Criteria C1 to C4 are sufficient conditions for obtaining one-copy equivalence, the criteria must ensure that any behavior β of the RS_{CC}^P module can be transformed in such a way that the result is a behavior γ of the $1CDB$ module.

We now study the structure of a transaction in a behavior β. Let β_t be the subsequence $\beta_t = \beta|\{A_n(t), B_{site(t)}(t), C_n(t,d)\colon n \in \mathcal{N}, d \in \mathcal{D}\}$. For each transaction $t \in T$, the β_t sequence will always be one of the sequences defined in Theorem 7.2 due to the conditions enforced by C2.

Theorem 7.2. *Let* $\beta \in behs(RS_{CC}^P)$*. For each transaction* $t \in T$*, the sequence* β_t *is one of the following sequences (where* $(n_1, \ldots, n_N)$ *is a permutation of the site identifiers,* $1..N$*):*

- $\beta_t =$ *empty*
- $\beta_t = B_{site(t)}(t) \cdot \gamma_{c_t}$ *with* $\gamma_{c_t} \preceq C_{n_1}(t,d_1) \ldots C_{n_N}(t,d_N)$ *(with* $d_1 \ldots d_N \in \mathcal{D}$*)*

• $\beta_t = B_{site(t)}(t) \cdot \gamma_{a_t}$ *with* $\gamma_{a_t} \preceq A_{n_1}(t) \ldots A_{n_N}(t)$

Proof. By Definition 3.1 and Property 5.1(a), for every $t \in \mathcal{T}$, either β_t is empty (in case $B_{site(t)}(t)$ is not in β), or $B_{site(t)}(t)$ is the first action involving t in β, i.e., $B_{site(t)}(t) \preceq \beta_t$. If $\gamma_{c_t} = \gamma_{a_t} =$ empty, the theorem holds; if not, β_t will be $B_{site(t)}(t) \cdot \gamma$. Then, let π_i, π_j be in β such that $i < j$. Now suppose that $\pi_i = A_n(t)$ is in γ. By contradiction, we assume that there also exists a $\pi_j = C_{n'}(t,d)$ in γ, for some $d \in \mathcal{D}$. By Definition 3.1, $n \neq n'$. Since β satisfies C2, such γ is not possible. The same happens if $\pi_i = C_n(t,d)$ and $\pi_j = A_{n'}(t)$. Thus, the theorem holds. □

A transaction $t \in T$ is said to be *committed in* a behavior $\beta \in behs(RS^P_{CC})$, denoted by $t \in Committed(\beta)$, if and only if β_t has an action $C_n(t,d)$ for some site $n \in \mathcal{N}$ and $d \in \mathcal{D}$, formally: $\beta_t \preceq B_{site(t)}(t) \cdot \gamma_{c_t}$ with $\gamma_{c_t} \neq empty$. In the same way $t \in T$ *is aborted in* $\beta \in behs(RS^P_{CC})$, ($t \in Aborted(\beta)$), if and only if β_t has an action $A_n(t)$ for any site $n \in \mathcal{N}$, formally: $\beta_t \preceq B_{site(t)}(t) \cdot \gamma_{a_t}$ with $\gamma_{a_t} \neq empty$.

As a result of Theorem 7.2: (i) if $t \in Committed(\beta)$, then $B_{site(t)}(t) \cdot C_{f_c(\beta_t)}(t,d)$ is a prefix of β_t (for some $d \in \mathcal{D}$), where $f_c(\beta_t)$ is the first site at which t is committed; and (ii) if $t \in Aborted(\beta)$, then $B_{site(t)}(t) \cdot A_{a_c(\beta_t)}(t)$ is a prefix of β_t, where $f_a(\beta_t)$ is the first site at which t is aborted. Next, we define the subsequence of β comprising the beginning and the first output (committed or aborted) of each transaction, as well as the crash and recover actions.

Definition 7.2 (Transaction's First-Output Behavior)**.** *Let β be a behavior of RS^P_{CC}. The subsequence β_F is defined as $\beta_F = \beta|F(\beta)$ where $F(\beta) = \{B_{site(t)}(t) \colon t \in T\} \cup \{C_{f_c(\beta_t)}(t,d) \colon t \in Committed(\beta), d \in \mathcal{D}\} \cup \{A_{f_a(\beta_t)}(t) \colon t \in Aborted(\beta)\} \cup \{crash_n \colon n \in \mathcal{N}\} \cup \{recover_n \colon n \in \mathcal{N}\}$.*

The β_F sequence has some useful properties, shown in Lemma 7.1, of which we will make use later.

Lemma 7.1. *Let β be a behavior of RS^P_{CC}. In the subsequence β_F the following conditions hold:*

1. $\pi_i \in \{C_{f_c(\beta_t)}(t,d), A_{f_a(\beta_t)}(t) : d \in \mathcal{D}\} \Rightarrow \exists k \colon k < i \colon \pi_k = B_{site(t)}(t)$
2. $\pi_i = B_{site(t)}(t) \Rightarrow \exists k \colon k > i \colon \pi_k \in \{C_{f_c(\beta_t)}(t, \delta(t,\beta)), A_{f_a(\beta_t)}(t), crash_n : n \in \mathcal{N}, d \in \mathcal{D}\}$
3. *For every $0 \leq j \leq |\beta_F|$, the following conditions hold:*
   - $0 \leq \#crash(\beta(j)_F) - \#recover(\beta(j)_F) \leq N$
   - $\#crash(\beta(j)_F) - \#recover(\beta(j)_F) = N \Rightarrow (\beta(j)_F = \beta_F \vee \pi_{j+1} = recover_n$ *for some* $n \in \mathcal{N})$

Proof. The first condition comes from Definition 7.2 of β_F and Theorem 7.2. Moreover, as $\beta \in behs(RS^P_{CC})$ satisfies C1, then by Definition 7.2 and Theorem 7.2 the second condition holds. Finally, the third condition is proven by Property 4.2. Since in every $\beta|DB^{CR}_n$ a $recover_n$ action is always immediately preceded by a $crash_n$ action, by the definition of β_F, $0 \leq \#crash(\beta(j)_F) - \#recover(\beta(j)_F)$. Furthermore, in every $\beta|DB^{CR}_n$ the action following a $crash_n$ action, if any, is $recover_n$. As a result, $\#crash(\beta(j)_F) - \#recover(\beta(j)_F) \leq N$; and in case $\#crash(\beta(j)_F) - \#recover(\beta(j)_F) = N$, by Property 4.2 no action is possible after π_j at any site of the RS^P_{CC} unless $\pi_{j+1} = recover_n$ for some $n \in \mathcal{N}$. □

The β_F sequence also satisfies that $log(\beta|DB^{CR}_n) \preceq log(\beta_F)$ for all $n \in \mathcal{N}$, as stated in Lemma 7.2 (recall that $log(\beta) = log(\beta, \mathcal{I})$). This means that β_F installs the same snapshots, and in the same order, as the ones installed at each replica of the replicated system. This consideration is possible due to β_F being trivially well-formed in the sense given by Definition 3.1 (see Lemma 7.1) and the fact that the writeset of a transaction is the same for all sites that commit it by Property 5.1.

Lemma 7.2. *Let $\beta \in behs(RS^P_{CC})$. It holds that $log(\beta(j)|DB^{CR}_n) \preceq log(\beta(j)_F)$ for every prefix $\beta(j) \preceq \beta$ and every $n \in \mathcal{N}$.*

Proof. Let $\beta(j)$ be a finite prefix of β for some index $j \in \mathbb{Z}^+$. By induction over $j \geq 0$.

- *Basis*: $j = 0$. $\beta(0)|DB^{CR}_n = \beta(0)_F =$empty and, by definition, $log(\beta(0)|DB^{CR}_n) = log(\beta(0)_F) =$empty.
- *Hypothesis*: $log(\beta(j)|DB^{CR}_n) \preceq log(\beta(j)_F)$ and $j > 0$.

- *Induction Step*: we only consider the events π_{j+1} affecting the lemma statement.

- $\pi_{j+1} = C_{f_c(\beta_t)}(t,d)$ (for some $d \in \mathcal{D}$) and $f_c(\beta_t) = n$. By Hypothesis, $log(\beta(j)|DB_n^{CR}) \preceq log(\beta(j)_F)$. The only possible case from the Hypothesis is $log(\beta(j)|DB_n^{CR}) = log(\beta(j)_F)$. Consider $log(\beta(j)|DB_n^{CR}) \prec log(\beta(j)_F)$. There is at least one different element $\langle d'.ws \rangle$ in $log(\beta(j)_F)$. Thus, $\beta(j)$ includes an action $\pi_{j'} = C_{f_c(\beta_{t'})}(t',d')$ with $j' < j$ which appears in $\beta(j)_F$ but not in $\beta(j)|DB_n^{CR}$. By C3, there is some replica ($f_c(\beta_{t'}) = n'$) $n' \neq n$ such that $log(\beta(j')|DB_n^{CR}) \prec log(\beta(j')|DB_{n'}^{CR})$. Then, $log(\beta(j)|DB_n^{CR}) \prec log(\beta(j)|DB_{n'}^{CR})$. By the definition of log (Definition 3.2), as $\beta(j+1)|DB_n^{CR} = \beta(j)|DB_n^{CR} \cdot \pi_{j+1}$ and $\beta(j+1)|DB_{n'}^{CR} = \beta(j)|DB_{n'}^{CR}$ then $log(\beta(j+1)|DB_n^{CR}) \not\preceq log(\beta(j+1)|DB_{n'}^{CR})$, which leads to a contradiction with C3. As a conclusion, $log(\beta(j)|DB_n^{CR}) = log(\beta(j)_F)$. As $\beta(j+1)|DB_n^{CR} = \beta(j)|DB_n^{CR} \cdot \pi_{j+1}$ and $\beta(j+1)_F = \beta(j)_F \cdot \pi_{j+1}$, by the log definition (Definition 3.2) $log(\beta(j+1)|DB_n^{CR}) = log(\beta(j+1)_F)$ holds.
- $\pi_{j+1} = C_{f_c(\beta_t)}(t,d)$ and $f_c(\beta_t) \neq n$. By Hypothesis, $log(\beta(j)|DB_n^{CR}) \preceq log(\beta(j)_F)$. Since $\beta(j+1)|DB_n^{CR} = \beta(j)|DB_n^{CR}$ and $\beta(j+1)_F = \beta(j)_F \cdot \pi_{j+1}$, then by the log definition, $log(\beta(j+1)|DB_n^{CR}) \prec log(\beta(j+1)_F)$ holds.
- $\pi_{j+1} = C_{n_k}(t,d)$ and $n_k = n$, being $n_k \neq f_c(\beta_t)$. By Theorem 7.2, there exists $j' < j$ such that $\pi_{j'} = C_{f_c(\beta_t)}(t,d')$ is in $\beta(j)_F$. This action is in $\beta(j)_F$ but not in $\beta(j)|DB_n^{CR}$. Moreover, by Property 5.1, $d.ws = d'.ws$. By induction Hypothesis, $log(\beta(j')|DB_n^{CR}) \prec log(\beta(j')_F)$ and also $log(\beta(j)|DB_n^{CR}) \prec log(\beta(j)_F)$. Thus, as $\beta(j+1)|DB_n^{CR} = \beta(j)|DB_n^{CR} \cdot \pi_{j+1}$ and $\beta(j+1)_F = \beta(j)_F$, by the log Definition 3.2, $log(\beta(j+1)|DB_n^{CR}) \preceq log(\beta(j+1)_F)$.
- $\pi_{j+1} = C_{n_k}(t,d)$ and $n_k \neq n$, being $n_k \neq f_c(\beta_t)$. In this case, $\beta(j+1)|DB_n^{CR} = \beta(j)|DB_n^{CR}$ and $\beta(j+1)_F = \beta(j)_F$. Thus, trivially $log(\beta(j+1)|DB_n^{CR}) \preceq log(\beta(j+1)_F)$ by induction Hypothesis.

Thus, the lemma holds. □

Since our aim is to obtain by construction a behavior that is a one-copy view of β, we slightly modify β_F according to Definition 6.1(1), in order to include the complete semantic of t given by $\delta(t,\beta)$.

Definition 7.3. *Let β be a behavior of RS_{CC}^P. The behavior β_{1C} is defined as: $\pi_i = C_{f_c(\beta_t)}(t, \delta(t,\beta))$ is in $\beta_{1C} \Leftrightarrow \pi_i = C_{f_c(\beta_t)}(t,d)$ is in β_F for some $d \in \mathcal{D}$; otherwise π_j is in $\beta_{1C} \Leftrightarrow \pi_j$ is in β_F.*

Recall that, due to Property 5.1 and the definition of $\delta(t,\beta)$, if $\pi_i = C_n(t,d)$ is in β for any $n \in \mathcal{N}$, $\delta(t,\beta).ws = d.ws$ and $\delta(t,\beta).inf = d.inf$. Moreover, if $n = site(t)$, then $\delta(t,\beta).rs = d.rs$ (otherwise $d.rs$ is empty, and if $\pi_i = C_{site(t)}(t,d)$ is not in β, then $\delta(t,\beta).rs$ is undefined). Thus, by the definition of β_{1C} and Property 5.1, Lemmas 7.1 and 7.2 hold trivially for β_{1C}.

As an example, Figure 7 shows the β_F and β_{1C} of a certain behavior. Note that the actions in β_F for each transaction $t \in T$ (i.e., $B_{site(t)}(t)$, $C_{f_c(\beta_t)}(t,d)$ and $A_{f_a(\beta_t)}(t)$) are unique. The same happens for β_{1C}.

The actions of sequence β_{1C} compose a behavior which somehow represents the way in which transactions behave in the replicated system. This behavior is not strictly the same as the one of a single database system (Definition 3.6), but it satisfies the generalized legal behavior of Definition 3.7, in which transactions may obtain older snapshots prior to their beginning. Theorem 7.3 covers this issue.

Theorem 7.3. *Let β be a behavior of RS_{CC}^P. For each transaction $t \in T$ such that $\pi_i = B_{site(t)}(t)$ and $\pi_j = C_{f_c(\beta_t)}(t, \delta(t,\beta))$ are in β_{1C}, there exists $0 \leq s \leq i$ such that the following conditions hold:*

(a) $compatible(t, s, \beta(j)_{1C})$

(b) $isolated(t', t, s, \beta(j)_{1C})$, *for all* $t' \in T$

(c) $consistent(t, \beta(j)_{1C})$

Proof. By the Definition 7.3 of β_{1C}, both $\pi_i = B_{site(t)}(t)$ and $\pi_j = C_{f_c(\beta_t)}(t,d)$ with $i < j$ exist in β for each considered $t \in T$, being $\delta(t,\beta).ws = d.ws$ and $\delta(t,\beta).inf = d.inf$. We can use the indexes i, j of β in β_{1C}, although the properties to be proven are related only with β_{1C}. Let $t_0 \in T$ be the transaction such that $t_0 = last(i, site(t), \beta)$ and $\pi_{i_0} = C_{site(t)}(t_0, d_0)$ is in β. By Theorem 7.2, there exists $\pi_{i'_0} = C_{f_c(\beta_t)}(t_0, d'_0)$ with $i'_0 < j$, as $log(\beta(j)|DB_{site(t)}^{CR}) \preceq log(\beta(j)|DB_{f_c(\beta_t)}^{CR})$. Figure 8 depicts the sequence of actions that are used in this proof.

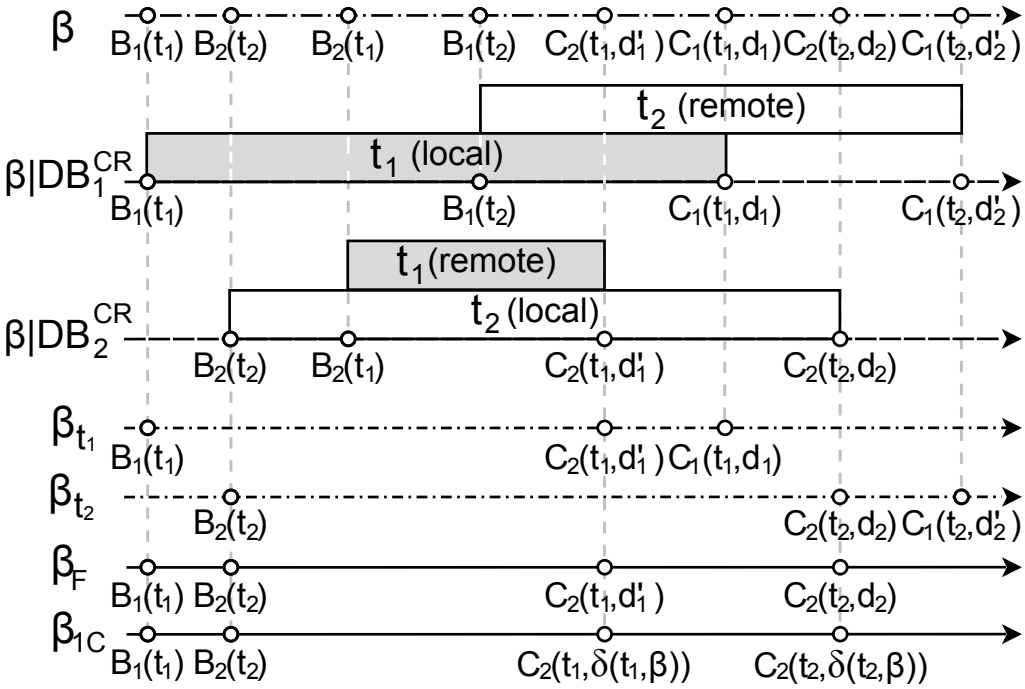

Figure 7: The β_F and β_{1C} for a behavior β

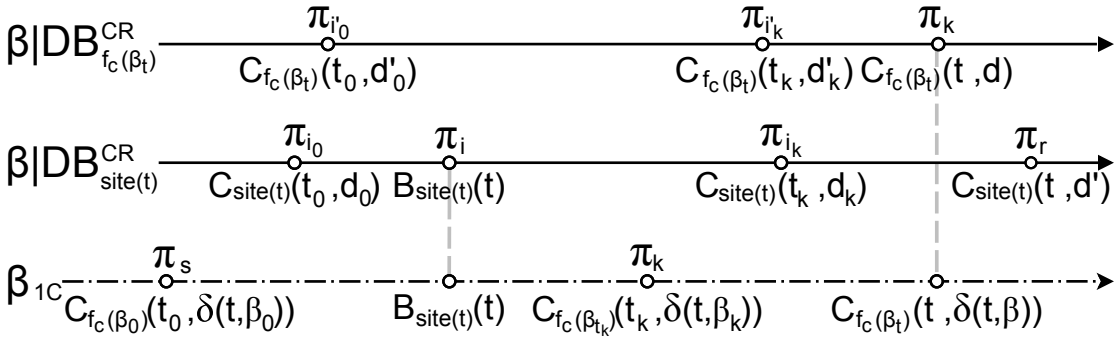

Figure 8: Actions used for the proof of Theorem 7.3.

- Proof of Condition (a):
If $\pi_r = C_{site(t)}(t, d')$ is not in β, then by the definition of $\delta(t, \beta)$ and Property 5.1, $\delta(t, \beta).rs = \emptyset$. Thus, $compatible(t, s, \beta(j)_{1C})$ is trivially true for any s. Otherwise, $\pi_r = C_{site(t)}(t, d')$ is in β, thus $\delta(t, \beta).rs = d'.rs$. Since $\beta|DB^{CR}_{site(t)}$ is a legal behavior, it holds that $compatible(t, i, \beta(r)|DB^{CR}_{site(t)})$. Note that, by its Definition 3.3, the snapshot only changes when a transaction is committed. Thus, $\mathcal{S}(\beta(i)|DB^{CR}_{site(t)}) = \mathcal{S}(\beta(i_0)|DB^{CR}_{site(t)})$. Then, we have to be concerned about the transactions t_k which were committed between the commitment of t_0 and t at $f_c(\beta_t)$ (where t was committed first); i.e., $t_k \in T$ such that $\pi_{i'_k} = C_{f_c(\beta_t)}(t_k, d'_k)$ is in β and $i'_0 < i'_k < j$.
By C3, $log(\beta(r)|DB^{CR}_{site(t)}) = log(\beta(j)|DB^{CR}_{f_c(\beta_t)})$. Let $\pi_{i_k} = C_{site(t)}(t_k, d_k)$ be the committed actions at $site(t)$ of each t_k with i_k: $i_0, i_1, \ldots, i_k, \ldots, i_m$ and $i_m < r$. The sequence of snapshots in $site(t)$ which makes $compatible(t, i, \beta(r)|DB^{CR}_{site(t)})$ true is: $\mathcal{S}(\beta(i_0)|DB^{CR}_{site(t)})\mathcal{S}(\beta(i_1)|DB^{CR}_{site(t)})$ $\ldots\mathcal{S}(\beta(i_k)|DB^{CR}_{site(t)})\ldots\mathcal{S}(\beta(i_m)|DB^{CR}_{site(t)})$. Note that t_0 and each t_k have been committed in β and also in β_{1C} (by the definition of β_{1C}). Then, let $\pi_s = C_{f_c(\beta_{t_0})}(t_0, \delta(t_0, \beta))$ and $\pi_k = C_{f_c(\beta_{t_k})}(t_k, \delta(t_k, \beta))$ be in β_{1C}, it is satisfied that $s \leq i_0 < i$ and, by Lemma 7.2, $s < k < j$ since $log(\beta(j)|DB^{CR}_{f_c(\beta_t)}) \preceq$

$log(\beta(j)_{1C})$.

Therefore, both behaviors have built the same snapshots. By the previous definition of π_s and π_k, then $log(\beta(i_0)|DB^{CR}_{site(t)}) = log(\beta(s)_{1C})$ and $log(\beta(i_k)|DB^{CR}_{site(t)}) = log(\beta(k)_{1C})$ for $k\colon 1 \ldots m$. By the snapshot Definition 3.3, $\mathcal{S}(\beta(s)_{1C}) \cdots \mathcal{S}(\beta(k)_{1C}) \cdots \mathcal{S}(\beta(m)_{1C})$ is the same sequence of snapshots. Then, $compatible(t, s, \beta(m)_{1C})$ holds and trivially $compatible(t, s, \beta(j)_{1C})$ holds too, since $d'.rs$ does not change after action $C_{site(t)}(t, d')$ is executed, and therefore neither does $\delta(t, \beta)$.

- Proof of Condition (b):

Recall that $\pi_{i_0} = C_{site(t)}(t_0, d_0)$ and $\pi_{i'_0} = C_{f_c(\beta_t)}(t_0, d'_0)$ are in β. By Property 5.1(b) and Definition 7.3, $d_0.inf = d'_0.inf = \delta(t_0, \beta).inf$. If $f_c(\beta_t) = site(t)$, then $\pi_{i'_0} = \pi_{i_0}$ and, since every $\beta \in behs(DB^{CR}_{f_c(\beta_t)})$ is legal, it holds that $\forall t'' \in T\colon\ isolated(t'', t, i, \beta(j)|DB^{CR}_{f_c(\beta_t)})$. As t_0 is the last committed transaction in $site(t)$ before $\pi_i = B_{site(t)}(t)$, the $isolated()$ predicate can be extended to i'_0, i.e., it holds that $\forall t'' \in T\colon\ isolated(t'', t, i'_0, \beta(j)|DB^{CR}_{f_c(\beta_t)})$. On the other hand, if $f_c(\beta_t) \neq site(t)$, then also, by C4, it holds that $\forall t'' \in T\colon\ isolated(t'', t, i'_0, \beta(j)|DB^{CR}_{f_c(\beta_t)})$. Then, we have to be again concerned about the transactions $t_k \in T$ such that $\pi_{i'_k} = C_{f_c(\beta_t)}(t_k, d'_k)$ and $i'_0 < i'_k < j$. By the definition of $isolated()$, these transactions satisfy $isolated(t_k, t, i'_k, \beta(j)|DB^{CR}_{f_c(\beta_t)})$.

Then, recalling that $\pi_s = C_{f_c(\beta_{t_0})}(t_0, \delta(t_0, \beta))$ and $\pi_k = C_{f_c(\beta_{t_k})}(t_k, \delta(t_k, \beta))$ are in β_{1C}, since $log(\beta(j)|DB^{CR}_{f_c(\beta_t)}) \preceq log(\beta(j)_{1C})$, $s < k < j$. Therefore, $isolated(t_k, t, s, \beta(j)_{1C})$. For the rest of committed transactions t' such that $\pi_{k'} = C_{f_c(\beta_{t'})}(t', \delta(t', \beta))$ and that $s < k' < j$ does not hold, $isolated(t', t, s, \beta(j)_{1C})$ holds too.

- Proof of Condition (c):

Finally, as $\pi_j = C_{f_c(\beta_t)}(t, d)$ is in β, since by definition every $\beta|DB^{CR}_{f_c(\beta_t)}$ is legal, $consistent(t, \beta(j)|DB^{CR}_{f_c(\beta_t)}) = \forall i\colon\ K_i(\mathcal{S}(\beta(j-1)|DB^{CR}_{f_c(\beta_t)}), d.ws)$. By Lemma 7.2, $log(\beta(j)|DB^{CR}_{f_c(\beta_t)}) = log(\beta(j)_{1C})$. Thus, $log(\beta(j-1)|\ DB^{CR}_{f_c(\beta_t)}) = log(\beta(j-1)_{1C})$, since $\pi_j = C_{f_c(\beta_t)}(t, \delta(t, \beta))$ is in β_{1C}, being $\delta(t, \beta).ws = d.ws$. Then, $\mathcal{S}(\beta(j-1)|DB^{CR}_{f_c(\beta_t)}) = \mathcal{S}(\beta(j-1)_{1C})$ and therefore $consistent(t, \beta(j)_{1C})$ holds.

□

One can think that the β_{1C} of each $\beta \in behs(RS^P_{CC})$ keeps the properties that prove the existence of a one-copy equivalence of the RS^P_{CC}. This is the conclusion drawn from Theorem 7.4, which proves that any behavior of the RS^P_{CC} can be transformed to become a behavior of the $1CDB$ module.

Theorem 7.4. *The RS^P_{CC} module satisfying Criteria C1 to C4 is one-copy equivalent to the $1CDB$.*

Proof. For each $\beta \in behs(RS^P_{CC})$, we define the following legal relation $\Gamma\colon behs(RS^P_{CC}) \rightarrow behs(1CDB)$; $\Gamma(\beta) = \mathcal{R}(\beta_{1C})$ where $\mathcal{R}()$ is a renaming function which removes every reference of a site in the actions of β_{1C}. That is, $B(t)$, $C(t, \delta(t, \beta))$, $A(t)$, $crash$ or $recover$ appear in $\mathcal{R}(\beta_{1C})$ when $B_{site(t)}(t)$, $C_{f_c(\beta_t)}(t, \delta(t, \beta))$, $A_{f_a(\beta_t)}(t)$, $crash_n$ or $recover_n$ appear in β_{1C}. Note that $|\beta_{1C}|\{crash_n : n \in \mathcal{N}\}| = |\mathcal{R}(\beta_{1C})|crash|$ and $|\beta_{1C}|\{recover_n : n \in \mathcal{N}\}| = |\mathcal{R}(\beta_{1C})|recover|$. Thus, β_{1C} satisfies Lemma 7.1 and Theorem 7.3; the definition of the semantic data $\delta(t, \beta)$ does not change for the transactions in $\mathcal{R}(\beta_{1C})$; and the actions in $\mathcal{R}(\beta_{1C})$ are included in the $1CDB$ signature, i.e., $acts(1CDB)$. Therefore, $\Gamma(\beta) \in behs(1CDB)$, i.e., $\Gamma(\beta)$ is a well-formed (Lemma 7.1.1), progressive (Lemma 7.1.2), N-crash stop (Lemma 7.1.3), and generalized legal (Theorem 7.3) behavior. □

7.3 Discussion

Abstraction of the recovery process: The aim of the recovery process is to obtain the effects of committed transactions that the recovering site lost during its down-time period, so that the recovering site can apply them and become up-to-date. In the presented mathematical model, the effects of committed transactions are represented by the evolution of $log(\beta(j)|DB^{CR}_n)$ at each of the system sites $n \in \mathcal{N}$. This log is, by Definition 3.2, the sequence of writesets of committed transactions at that site, i.e., it is defined by the actions $C_n(t, d)$ that occur at site n. According to this definition, after a site n executes a $recover_n$ action, the recovery protocol should ensure that all the missed transactions are committed in a certain order at n, so that Criterion C3 is fulfilled at the recovering site. This is the reason why, in the example of Figure 5,

after site 1 recovers, t_4 cannot be committed at site 1 before t_2 is committed, since in site 2 t_2 is committed before t_4. The model does not limit the implementation of correct recovery mechanisms, since the log definition could be adapted to reflect the behavior of the recovery protocol, as long as the recovery protocol provides a certain database snapshot from which the recovering site is considered to be up-to-date and can start executing transactions normally. Put another way, the presented model does not exclude the use of known recovery mechanisms [20], as it only requires that the effects of the recovery process on the database of the recovering site must be the same effects that transactions would have produced on the site if it had not crashed.

It is also worth noting that, since Criterion C3 precludes sites from diverging in the sequence of applied writesets, this criterion does not allow sites to work in different network partitions, i.e., there can be at most one partition processing transactions. The most common way of dealing with this problem consists of considering the primary partition system model [12], in which only a partition with at least $(N/2)+1$ sites, called majority partition, is allowed to work.

On the other hand, Criterion C1 establishes a liveness condition upon transactions (if a transaction begins, either a site crashes or the transaction produces an output). However, there are no liveness conditions regarding the progress of system sites. As a consequence, in this model a site may never recover after crashing, or it may even execute a $recover$ action but never execute all its pending transactions. Such conditions can be added to implementation specifications in order to ensure the termination of the recovery protocol.

8 Extending the model for partial replication

Up until now, we have considered a fully replicated system, in which each database stores a copy of all the possible items $\mathcal{I}$. As shown in this section, it is possible to extend the model to include partially replicated systems.

Instead of holding a copy of the whole set of items $\mathcal{I}$ at each system site, the database of each site $n \in \mathcal{N}$ (denoted by $DB_n^{CR}(\mathcal{I}_n, \mathcal{T})$) in a partially replicated system stores a subset of the set of items, $\mathcal{I}_n \subseteq \mathcal{I}$. The set of transaction identifiers $\mathcal{T}$ is the same for all databases.

The partially replicated system, represented by module PRS, consists of the composition of the $DB_n^{CR}(\mathcal{I}_n, \mathcal{T})$ modules: $PRS = \Pi_{n\in\mathcal{N}} DB_n^{CR}(\mathcal{I}_n, \mathcal{T})$. The set of items of the global system is $\mathcal{I} = \bigcup_{n\in\mathcal{N}} \mathcal{I}_n$. The set of all possible versions of the $DB_n^{CR}(\mathcal{I}_n, \mathcal{T})$ is denoted by $\mathcal{V}_n$, thus the set of versions of the system is $\mathcal{V} = \bigcup_{n\in\mathcal{N}} \mathcal{V}_n$. In the following, $\mathcal{I}_{nn'} = \mathcal{I}_n \cap \mathcal{I}_{n'}$.

Using the PRS module as a basis, we define a basic protocol for partial replication. Since the system is now partially replicated, the delegate site of a transaction t may not contain all the items affected by t. Thus, the delegate site may not be able to serve all the operations of t, which in that case will have to be forwarded to other sites.

In order to model the basic properties of the partially replicated system, we define a refinement of the PRS module, named PRS^{PP}, whose behaviors are restricted by Property 8.1.

Property 8.1. (PP: Partial Replication Abstraction) *For every behavior $\alpha \in behs(PRS^{PP})$ and every transaction $t \in \mathcal{T}$, the following conditions hold:*

(a) $\pi_i = B_n(t) \Rightarrow \exists j : j \leq i : \pi_j = B_{site(t)}(t)$

(b) $\pi_i = C_n(t,d) \wedge \pi_j = C_{n'}(t,d') \Rightarrow d.ws(\mathcal{I}_{nn'}) = d'.ws(\mathcal{I}_{nn'}) \wedge d.inf(\mathcal{I}_{nn'}) = d'.inf(\mathcal{I}_{nn'})$

Similarly to Property 5.1, Property 8.1 states that a remote transaction can begin only after the corresponding local transaction began at the delegate site. Moreover, when a transaction is committed in two different sites, the writeset and control information regarding the items that the two sites share in common must be the same. There are no limitations on the readset, since it may be necessary to execute read operations in different sites by means of remote transactions, and therefore remote transactions may have a non-empty readset.

The definition of the semantic data of a transaction in a behavior of the PRS^{PP} is straightforwardly obtained by replacing RS^P by PRS^{PP} in Definition 5.1.

Note that Remark 5.1 is no longer applicable, as read-only transactions can read items from different sites. However, in the development explained for RS^P_{CC} there are no contradictions if read-only transactions are included in $behs(RS^P_{CC})$.

Let us consider every partial replication protocol than can be built using the PRS^{PP} as a basis. This can be done by specifying a set of properties φ that must be verified by the PRS^{PP}. This is denoted by PRS^{PP}_{φ}

One-copy equivalence can be used as correctness criterion for partially replicated systems. In order to be one-copy equivalent to the $1CDB(\mathcal{I},\mathcal{T})$ module, all transactions in the PRS^{PP}_{φ} must behave as if they were executed in a single database system containing all the items of the whole system. Definitions 6.1 and 6.2 can be adapted by replacing RS^P_{φ} by PRS^{PP}_{φ}. That is, PRS^{PP}_{φ} is one-copy equivalent to the $1CDB(\mathcal{I},\mathcal{T})$ module if and only if there exists a legal relation $\Gamma_{PP} \subseteq behs(PRS^{PP}_{\varphi}) \times behs(1CDB(\mathcal{I},\mathcal{T}))$.

Recall that the $1CDB(\mathcal{I},\mathcal{T})$ module has been used for proving that a fully replicated system is equivalent to it. In particular, the RS^P_{CC} module is one-copy equivalent to the $1CDB(\mathcal{I},\mathcal{T})$ when every database of RS^P_{CC} holds a copy of all the items in $\mathcal{I}$. Moreover, if $\gamma \in behs(1CDB(\mathcal{I},\mathcal{T}))$, then there exists a behavior $\beta \in behs(RS^P_{CC})$ such that $\gamma \in \Gamma_P(\beta)$, where Γ_P is the legal relation between the behaviors of RS^P_{CC} and $1CDB(\mathcal{I},\mathcal{T})$. The proof is straightforward by construction of the β from the γ behavior. As a result, we can state the following theorem:

Theorem 8.1. *Let $\alpha \in behs(PRS^{PP}_{\varphi})$. If PRS^{PP}_{φ} is one-copy equivalent to the $1CDB(\mathcal{I},\mathcal{T})$ module, then there exists a correct fully replicated system RS^P_{CC} that is one-copy equivalent to $1CDB(\mathcal{I},\mathcal{T})$, such that if $\gamma \in \Gamma_{PP}(\alpha)$, then $\gamma \in \Gamma_P(\beta)$ for some $\beta \in behs(RS^P_{CC})$.*

The previous theorem states that a behavior α of a correct partially replicated system is, in some sense, indistinguishable from a behavior β of a correct fully replicated system. Every committed transaction t in α, whose semantic data is $\delta(t,\alpha)$, appears in β with $\delta(t,\alpha) = \delta(t,\beta)$. Thus, regardless of the number of sites that perform read operations of a transaction t in α, all these read operations occur as if they had been executed in RS^P_{CC} (i.e. fully replicated database system with replication protocol fulfilling Property 5.1 and correctness criteria C1-C4).

Obtaining a group of necessary and sufficient conditions in order to guarantee that a PRS^{PP}_{φ} is one-copy equivalent to $1CDB(\mathcal{I},\mathcal{T})$ is not a simple task. However, the previous theorem allows us to notice the dificulties of designing correct partial replication protocols [4, 39]. For instance, if the isolation level of a transaction t requires read operations of t to be able to access the values written by t, the writeset of t will have to be available at sites that execute read operations of t, even if those sites do not store the items of the writeset. Moreover, if the read operations of a transaction retrieve item values from different sites, in order to achieve one-copy equivalence those values have to belong to the same database snapshot.

9 Conclusion

This paper reviews the notion of one-copy equivalence for replicated database systems supporting the crash-recovery failure model. The considered replication technique (in which read operations are executed at a delegate site of the transaction, whereas the effects of write operations are propagated to all sites) has been regularly used in the database replication field, since it is appropriate for update-everywhere approaches. By executing transaction operations in a delegate site and propagating later its writeset to other replicas, this kind of replication protocols ensure a high level of consistency and an asymmetric workload management that boosts performance, since writeset application at remote sites requires less effort than a local transaction service.

Up to our knowledge, there exists no general formalization of this kind of replication protocols that considers transactions running under different isolation levels, supporting both integrity constraints and crash failures. By means of the I/O Automaton model, we have provided a detailed specification of a replicated database system that fills this void. We have established the necessary and sufficient conditions that implementations of the considered model must fulfill in order to be correct under the one-copy equivalence notion. The stated properties can be considered as a new proposal for one-copy equivalence criteria, which can serve as a basis for the development and formal proof of such kind of protocols.

Acknowledgment

This work has been supported by the Spanish Government under research grant TIN2009-14460-C03.

References

[1] Yair Amir and Ciprian Tutu. From total order to database replication. In *ICDCS*, pages 494–, 2002.

[2] José Enrique Armendáriz-Íñigo, José Ramón González de Mendívil, José Ramón Garitagoitia, and Francesc D. Muñoz-Escoí. Correctness proof of a database replication protocol under the perspective of the I/O automaton model. *Acta Inf.*, 46(4):297–330, 2009.

[3] José Enrique Armendáriz-Íñigo, Juárez-Rodríguez, José Ramón González de Mendívil, José Ramón Garitagoitia, Luis Irún-Briz, and Francesc D. Muñoz-Escoí. A formal characterization of SI-based ROWA replication protocols. *Data Knowl. Eng.*, 70(1):21 – 34, 2011.

[4] José Enrique Armendáriz-Iñigo, A. Mauch-Goya, José Ramón González de Mendívil, and Francesc D. Muñoz-Escoí. SIPRe: a partial database replication protocol with si replicas. In Roger L. Wainwright and Hisham Haddad, editors, *SAC*, pages 2181–2185. ACM, 2008.

[5] José Enrique Armendáriz-Iñigo, Francesc D. Muñoz-Escoí, José Ramón Juárez-Rodríguez, José Ramón González de Mendívil, and Bettina Kemme. A recovery protocol for middleware replicated databases providing gsi. In *ARES'07: Proceedings of the 2nd International Conference on Availability, Reliability and Security*, pages 85–92, Washington, DC, USA, 2007. IEEE Computer Society.

[6] Josep M. Bernabé-Gisbert, Raúl Salinas-Monteagudo, Luis Irún-Briz, and Francesc D. Muñoz-Escoí. Managing multiple isolation levels in middleware database replication protocols. In *ISPA '06: Proceedings of th 4th International Symposium on Parallel and Distributed Processing and Applications*, pages 511–523, 2006.

[7] J.M. Bernabé-Gisbert, J.E. Armendáriz-Iñigo, Rubén de Juan-Marín, and F.D. Muñoz-Escoí. Providing read committed isolation level in non-blocking rowa database replication protocols. In *JCSD '07: Proceedings of XV Jornadas de Concurrencia y Sistemas Distribuidos*, 2007.

[8] Philip A. Bernstein, Vassco Hadzilacos, and Nathan Goodman. *Concurrency Control and Recovery in Database Systems*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1987.

[9] Michael J. Carey and Miron Livny. Conflict detection tradeoffs for replicated data. *ACM Trans. Database Syst.*, 16(4):703–746, 1991.

[10] N. Carvalho, A. Correia Jr., J. Pereira, L. Rodrigues, R. Oliveira, and S. Guedes. On the use of a reflective architecture to augment database management systems. *Journal of Universal Computer Science*, 13(8):1110–1135, 2007.

[11] Francisco Castro-Company, Luis Irún-Briz, Félix García-Neiva, and Francesc D. Muñoz-Escoí. Fobr: A version-based recovery protocol for replicated databases. In *PDP*, pages 306–313, 2005.

[12] Gregory V. Chockler, Idid Keidar, and Roman Vitenberg. Group communication specifications: a comprehensive study. *ACM Comput. Surv.*, 33(4):427–469, 2001.

[13] Khuzaima Daudjee and Kenneth Salem. Lazy database replication with snapshot isolation. In *VLDB '06: Proceedings of the 32nd International Conference on Very Large DataBases*, pages 715–726. VLDB Endowment, 2006.

[14] Sameh Elnikety, Willy Zwaenepoel, and Fernando Pedone. Database replication using generalized snapshot isolation. In *IEEE Symposium on Reliable Distributed Systems*, pages 73–84, Washington, DC, USA, 2005. IEEE Computer Society.

[15] Jim Gray and Andreas Reuter. *Transaction Processing: Concepts and Techniques*. Morgan Kaufmann Publishers, San Francisco, CA, USA, 1992.

[16] Theo Härder and Andreas Reuter. Principles of transaction-oriented database recovery. *ACM Comput. Surv.*, 15(4):287–317, 1983.

[17] JoAnne Holliday, Robert C. Steinke, Divyakant Agrawal, and Amr El Abbadi. Epidemic algorithms for replicated databases. *IEEE Trans. Knowl. Data Eng.*, 15(5):1218–1238, 2003.

[18] Ricardo Jiménez-Peris, Marta Patiño-Martínez, and Gustavo Alonso. Non-intrusive, parallel recovery of replicated data. In *SRDS*, pages 150–159. IEEE Computer Society, 2002.

[19] Bettina Kemme and Gustavo Alonso. A new approach to developing and implementing eager database replication protocols. *ACM Trans. Database Syst.*, 25(3):333–379, 2000.

[20] Bettina Kemme, Alberto Bartoli, and Özalp Babaoglu. Online reconfiguration in replicated databases based on group communication. In *DSN*, pages 117–130. IEEE Computer Society, 2001.

[21] Bettina Kemme, Fernando Pedone, Gustavo Alonso, Andre Schiper, and Matthias Wiesmann. Using optimistic atomic broadcast in transaction processing systems. *IEEE Trans. on Knowl. and Data Eng.*, 15(4):1018–1032, 2003.

[22] Konstantinos Krikellas, Sameh Elnikety, Zografoula Vagena, and Orion Hodson. Strongly consistent replication for a bargain. In *ICDE*, pages 52–63, 2010.

[23] Leslie Lamport. *Specifying Systems, The TLA+ Language and Tools for Hardware and Software Engineers*. Addison-Wesley, 2002.

[24] Yi Lin, Bettina Kemme, Ricardo Jiménez-Peris, Marta Patiño-Martínez, and José Enrique Armendáriz-Iñigo. Snapshot isolation and integrity constraints in replicated databases. *ACM Trans. Database Syst.*, 34(2):1–49, 2009.

[25] Yi Lin, Bettina Kemme, Marta Patiño-Martínez, and Ricardo Jiménez-Peris. Middleware based data replication providing snapshot isolation. In *SIGMOD '05: Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data*, pages 419–430, New York, NY, USA, 2005. ACM.

[26] Nancy A. Lynch. *Distributed Systems*. Morgan Kaufmann Publishers, 1996.

[27] Nancy A. Lynch and Mark R. Tuttle. An introduction to input/output automata. *CWI-Quarterly*, 2(3):219–246, 1989.

[28] Takeshi Mishima and Hiroshi Nakamura. Pangea: An eager database replication middleware guaranteeing snapshot isolation without modification of database servers. *PVLDB*, 2(1):1066–1077, 2009.

[29] F. D. Muñoz-Escoí, J. Pla-Civera, M. I. Ruiz-Fuertes, L. Irún-Briz, H. Decker, J. E. Armendáriz-Íñigo, and J. R. González de Mendívil. Managing transaction conflicts in middleware-based database replication architectures. In *IEEE Symposium on Reliable Distributed Systems*, pages 401–420, Washington, DC, USA, 2006. IEEE Computer Society.

[30] Francesc D. Muñoz-Escoí, Ruben de Juan-Marin, José Enrique Armendáriz-Iñigo, and José Ramón González de Mendívil. Persistent logical synchrony. In *NCA '08. Seventh IEEE International Symposium on Network Computing and Applications*, pages 253 –258, July 2008.

[31] Francesc D. Muñoz-Escoí, María Idoia Ruiz-Fuertes, Hendrik Decker, José Enrique Armendáriz-Iñigo, and José Ramón González de Mendívil. Extending middleware protocols for database replication with integrity support. In Robert Meersman and Zahir Tari, editors, *OTM Conferences (1)*, volume 5331 of *Lecture Notes in Computer Science*, pages 607–624. Springer, 2008.

[32] Christos H. Papadimitriou. *The Theory of Database Concurrency Control*. Computer Science Press, 1986.

[33] Marta Patiño-Martínez, Ricardo Jiménez-Peris, Bettina Kemme, and Gustavo Alonso. MIDDLE-R: Consistent database replication at the middleware level. *ACM Trans. Comput. Syst.*, 23(4):375–423, 2005.

[34] Christian Plattner, Gustavo Alonso, and M. Tamer Özsu. Extending DBMSs with satellite databases. *The VLDB Journal*, 17(4):657–682, 2008.

[35] María Idoia Ruiz-Fuertes, Francesc D. Muñoz-Escoí, Hendrik Decker, José Enrique Armendáriz-Iñigo, and José Ramón González de Mendívil. Integrity dangers in certification-based replication protocols. In Robert Meersman, Zahir Tari, and Pilar Herrero, editors, *OTM Workshops*, volume 5333 of *Lecture Notes in Computer Science*, pages 924–933. Springer, 2008.

[36] J. Salas, R. Jiménez-Peris, M. Patiño-Martínez, and B. Kemme. Lightweight reflection for middleware-based database replication. In *IEEE Symposium on Reliable Distributed Systems*, pages 377–390, Washington, DC, USA, 2006. IEEE Computer Society.

[37] R. Salinas, F. D. Muñoz-Escoí, J. E. Armendáriz-Iñigo, and J. R. Mendívil. A performance evaluation of g-bound with a consistency protocol supporting multiple isolation levels. In *OTM '08: Proceedings of the OTM Confederated International Workshops*, pages 914–923, Berlin, Heidelberg, 2008. Springer-Verlag.

[38] Rodrigo Schmidt. *Deferred-Update Database Replication: Theory and Algorithms*. PhD thesis, EPFL, 2008.

[39] Damián Serrano, Marta Patiño-Martínez, Ricardo Jiménez-Peris, and Bettina Kemme. Boosting database replication scalability through partial replication and 1-copy-snapshot-isolation. In *PRDC*, pages 290–297. IEEE Computer Society, 2007.

[40] Shuqing Wu and Bettina Kemme. Postgres-R(SI): Combining replica control with concurrency control based on snapshot isolation. In *ICDE '05: Proceedings of the 2005 IEEE International Conference on Data Engineering*, pages 422–433, Washington, DC, USA, 2005. IEEE Computer Society.