Consistent Query Answering for Atemporal Constraints over Temporal Databases

Jan Chomicki
Dept. of Computer Science and Engineering
SUNY at Buffalo
Buffalo, NY, USA
chomicki@buffalo.edu

Jef Wijsen
Département d'Informatique
University of Mons
Mons, Belgium
jef.wijsen@umons.ac.be

***Abstract*—Consistent query answering is a principled approach to query answering on inconsistent databases: when an inconsistent database has more than one plausible repair, queries are answered by returning the intersection of the query answers over all repairs. In this paper, we study consistent query answering over temporal databases relative to atemporal integrity constraints. A temporal database is conceptually viewed as a sequence of atemporal snapshot databases indexed by time. Two approaches to repairing are presented. In the first approach, each snapshot database is repaired individually and independently of earlier or later snapshots. This independence between snapshots facilitates the computation of consistent query answers. A second approach, which is seemingly more realistic, favors the persistence of attribute values in repairs.**

***Keywords*-databases; query processing**

I. Motivation

A database may be inconsistent with respect to a set of integrity constraints for several reason. For example, when two corporate databases are integrated in the event of a business combination, the result may be inconsistent because the databases stored contradictory information. Database cleaning (or repairing) refers to the process of removing inconsistencies from an inconsistent database. A difficulty in this process is that there may not be a single best way to clean a database. *Consistent query answering* (CQA) [1] provides an elegant way for dealing with this difficulty. In CQA, the *consistent answer* to a query is the intersection of the answers to the query on all repairs. For a Boolean query, the consistent answer is $true$ if the query evaluates to $true$ on every repair. The rationale behind CQA is that if several repairs are plausible, then only the answers that are valid in each repair should be returned.

CQA is attractive but comes at a price in computational complexity. For example, it is known for many years [2] that there exists a Boolean conjunctive query q whose consistent answer with respect to primary key constraints cannot be computed in polynomial time in the size of the database (unless $\mathbf{P} = \mathbf{NP}$). Intuitively, the source of this intractability is the exponential number of repairs. An area of active research has been to identify queries and constraints that allow CQA in polynomial time [3]. For example, for primary key constraints and conjunctive queries without self-joins, the (in)tractability of CQA is by now well understood [4].

So far, CQA has mainly focused on atemporal databases. The first part of this paper explores a simple setting that allows "temporalizing" known CQA results. A temporal database can be conceptually viewed as a sequence of atemporal databases (called *snapshots*) indexed by time. If all integrity constraints are atemporal, then such temporal database can be repaired by repairing each snapshot individually. Next, a class of atemporal Boolean queries can be "temporalized" in a straightforward manner by closing the class under temporal modal operators (like $\Box$, $\Diamond$, $\bigcirc$) and their Boolean combinations. We will study consistent query answering in such simple temporal setting. In particular, we introduce a condition, called *Closeness Under Crossover*, which captures that every snapshot is repaired individually and independently of past or future snapshots. We will show that this condition admits some nice equivalent rewritings of temporal queries under CQA semantics.

Although *Closeness Under Crossover* is theoretically elegant and attractive, this assumption may not be generally plausible. In the second part of the paper, we therefore revisit the assumption that a temporal database can be repaired by repairing each snapshot individually. For example, consider the temporal information encoded in Fig. 1. The first tuple of S encodes that Ed's salary was 10K from time 1 on. The second tuple of S encodes that Ed's salary was 12K during that same time period. Assume a constraint saying that at any one time, no employee can have two distinct salaries. The relation S represents a sequence of snapshots indexed by time, where for each integer $i \in [1, +\infty]$, the ith snapshot contains both $\mathsf{S}(\mathrm{Ed}, 10\mathrm{K})$ and $\mathsf{S}(\mathrm{Ed}, 12\mathrm{K})$. Each snapshot contains two conflicting tuples and can be repaired by deleting either of both tuples. If each snapshot is repaired individually, then for each non-negative integer n, there exists a repair where Ed's salary changed n times. Such repairs are unrealistic for large n. In fact, it seems more likely to assume that either Ed earned 10K throughout the interval $[1, +\infty]$, or Ed earned 12K throughout that interval.

This article reflects on ways of extending CQA to temporal databases, starting with the most obvious case where all integrity constraints are atemporal. It is organized as follows. Section II defines (abstract) temporal databases as well as a very generic notion of repairing. Section III introduces

S	Ename	Sal	Time
	Ed	10K	$[1, +\infty]$
	Ed	12K	$[1, +\infty]$

Figure 1. Concrete temporal database instance $\mathbf{db}$.

CQA for a temporal query language. Section IV shows a theoretical result concerning query rewriting for CQA under the *Closeness Under Crossover* condition, which captures that all snapshots are repaired independently. Section V revisits the latter condition and introduces a novel repair construct. Section VI discusses related work and Section VII concludes the paper.

II. Temporal Data Model

We assume a fixed database schema. An *atemporal database instance* is a relational database instance (over the fixed schema) defined in a standard way. To simplify the technical treatment, we will assume that an atemporal database is a finite set of *facts*, where a fact is an expression of the form $R(a_1, \dots, a_n)$ where R is an n-ary relation name and $a_1, \dots, a_n$ are constants.

A *temporal database instance* is an infinite sequence $\langle I_0, I_1, I_2, \dots \rangle$ where

1) for every $\ell \in \mathbb{N}_0$, we have that I_ℓ is an atemporal database instance (over the fixed schema), called *snapshot*; and
2) *Finite Change Condition:* there exists $m \in \mathbb{N}_0$ such that $I_m = I_{m+1} = I_{m+2} = \cdots$.

Intuitively, the second condition means that from some time point on, the recorded information remains unchanged. This condition is plausible in most temporal database applications, where updates are typically not known before they occur, and current data is assumed to remain true as long as it is not deleted or modified. A practical consequence is that every temporal database instance can be finitely presented by a technique known as tuple-timestamping where every timestamp is either $+\infty$ or a non-negative integer. This representation is common in temporal database systems and will be discussed in Section V.

For example, let I and J be two atemporal database instances and let $\mathcal{I} = \langle I, J, I, J, I, J, \dots \rangle$. Then $\mathcal{I}$ is a temporal database instance if and only if $I = J$. If $I = J$, then the temporal database instance $\mathcal{I}$ can be finitely represented by timestamping each fact $R(\vec{a})$ of I with $R(\vec{a}, [0, +\infty])$, expressing that $R(\vec{a})$ was valid at any time from time 0 on. If $I \neq J$, then the sequence $\mathcal{I}$ violates the *Finite Change Condition* and hence is not a legal temporal database instance.

In practice, if a temporal database instance is inconsistent with respect to some set of integrity constraints, it can be cleaned or repaired to make it consistent. In general, the same temporal database instance can be repaired in multiple ways. To formally capture this, we assume a function ρ (called *repair function*) from the set of temporal database instances to the powerset of the set of temporal database instances. Intuitively, if $\mathcal{I}$ is a temporal database instance, then $\rho(\mathcal{I})$ is the nonempty set of repairs of $\mathcal{I}$. If $\mathcal{R}$ belongs to $\rho(\mathcal{I})$, then we say that $\mathcal{R}$ is a *repair* of $\mathcal{I}$. Typically, if $\mathcal{R}$ is a repair of $\mathcal{I}$, then $\rho(\mathcal{R}) = \{\mathcal{R}\}$, because a repair is consistent.

In most works on database repairing, the function ρ is defined in an implicit way by fixing a set Σ of integrity constraints and a repair construct. For example, we could define a repair of $\mathcal{I}$ as any temporal database instance $\mathcal{R}$ that satisfies Σ and that preserves a maximal number of facts from $\mathcal{I}$. We will postpone such implicit definitions until Section V. It suffices to say here that we have in mind integrity constraints that are atemporal, i.e., that apply to atemporal database instances. That is, given an atemporal database instance I, we can tell whether or not I is consistent. Such notion of consistency naturally carries over to temporal database instances: a temporal database instance $\mathcal{I} = \langle I_0, I_1, \dots \rangle$ is *consistent* if I_j is consistent for every $j \in \mathbb{N}_0$. An atemporal integrity constraint is, for example, that no employee can have two distinct salaries at any one time.

Although our notion of temporal repair may be little surprising, it has some consequences that merit particular attention. First, since every repair $\langle R_0, R_1, \dots \rangle$ is a temporal database instance, there must exist m such that $R_m = R_{m+1} = R_{m+2} = \cdots$. Second, if $\langle R_0, R_1, \dots \rangle$ is a repair of some temporal database instance $\mathcal{I} = \langle I_0, I_1, \dots \rangle$, then $I_\ell = I_{\ell+1}$ does not imply $R_\ell = R_{\ell+1}$ (for any $\ell \in \mathbb{N}_0$). Intuitively, two consecutive atemporal databases instances I_ℓ and $I_{\ell+1}$ may be identical and yet be "repaired" in different ways.

For example, let $\mathcal{I} = \langle I, I, I, \dots \rangle$ be an inconsistent temporal database instance with the same atemporal database instance I at any time. Assume that R_1 and R_2 are two distinct consistent atemporal database instances that are "reasonable" ways of repairing I. Then, $\rho(\mathcal{I})$ may contain $\langle R_1, R_1, R_1, \dots \rangle$ and $\langle R_2, R_2, R_2, \dots \rangle$, but also

$$\langle R_1, R_2, R_1, R_2, \cdots, R_1, R_2, R_1, R_1, R_1, \dots \rangle.$$

This repair first switches finitely many times between R_1 and R_2, and then R_1 holds at all future times. On the other hand $\langle R_1, R_2, R_1, R_2, R_1, R_2, \dots \rangle$ is not a temporal database instance (and hence not a repair) because it falsifies the *Finite Change Condition*.

III. Temporalizing an Existing Query Language

In this section, we will first temporalize an existing query language, and then add a modal operator for consistent query answering.

A. Standard Temporal Semantics

Let $\mathcal{C}$ be a class of (atemporal) Boolean queries. We define $\mathsf{TL}(\mathcal{C})$ as the following family of temporal queries:

- every query in $\mathcal{C}$ belongs to $\mathsf{TL}(\mathcal{C})$;
- if φ_1 and φ_2 belong to $\mathsf{TL}(\mathcal{C})$, then $\varphi_1 \land \varphi_2$ belongs to $\mathsf{TL}(\mathcal{C})$;
- if φ belongs to $\mathsf{TL}(\mathcal{C})$, then $\Diamond\varphi$ and $\Box\varphi$ belong to $\mathsf{TL}(\mathcal{C})$.

The modal operator $\Diamond$ represents "Eventually," and $\Box$ represents "Always."

Example 1. In our running example, we assume a relation schema $\mathsf{E}(\mathsf{Ename}, \mathsf{LivesIn}, \mathsf{WorksFor})$. A fact $\mathsf{E}(e, \ell, w)$ means that employee e lives in the city ℓ and works for company w. We assume that employees are uniquely identified by their name, that no employee lives in two distinct cities, and that no employee works for two distinct companies.

The question *"Is it true that Ed lived in Paris and then in London, in which town he remained?"* can be expressed as follows in $\mathsf{TL}(\mathcal{C})$:

$$\Diamond\left(\exists z_1 \mathsf{E}(\text{'Ed'}, \text{'Paris'}, z_1) \land \Diamond\Box\forall y \forall z_2\left(\mathsf{E}(\text{'Ed'}, y, z_2) \rightarrow y = \text{'London'}\right)\right).$$

In this temporal query, $\exists z_1 \mathsf{E}(\text{'Ed'}, \text{'Paris'}, z_1)$ (henceforth denoted by q_1) and $\forall y \forall z_2 \left(\mathsf{E}(\text{'Ed'}, y, z_2) \rightarrow y = \text{'London'}\right)$ (henceforth denoted by q_2) are understood to be queries in $\mathcal{C}$. $\Box$

Let $\mathcal{I} = \langle I_0, I_1, I_2, \ldots \rangle$ be a temporal database instance and $\ell \in \mathbb{N}_0$. The notion $\models$ is defined as usual:

- for $q \in \mathcal{C}$, we have $\mathcal{I}, \ell \models q$ if $I_\ell \models q$;
- $\mathcal{I}, \ell \models \varphi_1 \land \varphi_2$ if $\mathcal{I}, \ell \models \varphi_1$ and $\mathcal{I}, \ell \models \varphi_2$;
- $\mathcal{I}, \ell \models \Diamond\varphi$ if for some $m \in \mathbb{N}_0$ such that $m \geq \ell$, we have $\mathcal{I}, m \models \varphi$;
- $\mathcal{I}, \ell \models \Box\varphi$ if for all $m \in \mathbb{N}_0$ such that $m \geq \ell$, we have $\mathcal{I}, m \models \varphi$.

Finally,

- $\mathcal{I} \models \varphi$ if $\mathcal{I}, 0 \models \varphi$.

Two queries φ_1 and φ_2 in $\mathsf{TL}(\mathcal{C})$ are *equivalent*, denoted $\varphi_1 \equiv \varphi_2$, if for every temporal database instance $\mathcal{I}$, we have $\mathcal{I} \models \varphi_1$ if and only if $\mathcal{I} \models \varphi_2$.

The following result will be used later on.

Lemma 1. *Every temporal query in $\mathsf{TL}(\mathcal{C})$ can be formulated equivalently as a conjunction $\bigwedge_{i=1}^{n} \varphi_i$ where each φ_i is of one of the following forms:*

- *$\varphi_i \equiv q$ with $q \in \mathcal{C}$;*
- *$\varphi_i \equiv \Diamond\psi$ with $\psi \in \mathsf{TL}(\mathcal{C})$;*
- *$\varphi_i \equiv \Box q$ with $q \in \mathcal{C}$; or*
- *$\varphi_i \equiv \Box\Diamond\psi$ with $\psi \in \mathsf{TL}(\mathcal{C})$.*

Proof: Exhaustively apply the following equivalences:

$$\begin{aligned} \Box\left(\varphi_1 \land \varphi_2\right) &\equiv \Box\varphi_1 \land \Box\varphi_2 \\ \Box\left(\Box\psi\right) &\equiv \Box\psi \end{aligned}$$

The query so obtained will obviously be in one of the desired forms. ■

B. Temporal Consistent Query Answering

We now add a modal operator $\star$ to our language, giving a new language $\mathsf{TL}^{\star}(\mathcal{C})$. This modal operator $\star$ captures the paradigm of consistent query answering: the intended meaning of $\star\varphi$ is that φ is true in all repairs.

- If φ belongs to $\mathsf{TL}(\mathcal{C})$, then $\star\varphi$ belong to $\mathsf{TL}^{\star}(\mathcal{C})$;
- if φ_1 and φ_2 belong to $\mathsf{TL}^{\star}(\mathcal{C})$, then $\varphi_1 \land \varphi_2$ belongs to $\mathsf{TL}^{\star}(\mathcal{C})$; and
- if φ belongs to $\mathsf{TL}^{\star}(\mathcal{C})$, then $\Diamond\varphi$ and $\Box\varphi$ belong to $\mathsf{TL}^{\star}(\mathcal{C})$.

According to this syntax, every query in $\mathsf{TL}^{\star}(\mathcal{C})$ contains at least one occurrence of the modal operator $\star$, and this operator cannot be nested.

The semantics of $\land$, $\Diamond$ and $\Box$ is defined in the usual way (not repeated here); the semantics of $\star$ is defined next. Intuitively, $\star\varphi$ says that φ holds in every repair. That is, $\star$ acts as a universal quantification over all repairs (in the same way as $\Box$ acts as a universal quantification over all time instants). The semantics is defined relative to a repair function ρ (introduced in Section II).

- For every $\ell \in \mathbb{N}_0$, we have $\mathcal{I}, \ell \models_\rho \star\varphi$ if for every repair $\mathcal{R}$ in $\rho(\mathcal{I})$, we have $\mathcal{R}, \ell \models \varphi$; and
- $\mathcal{I} \models_\rho \star\varphi$ if $\mathcal{I}, 0 \models_\rho \star\varphi$.

Since every repair is its own repair, it should be clear why nesting of $\star$ is not useful (and hence has been disallowed in our language).

For a fixed repair function ρ (represented in some finite way), the problem $\mathsf{TEMP\_CERTAINTY}(\varphi)$ takes as input a temporal database instance $\mathcal{I}$ (represented as a concrete temporal database instance, introduced in Section V), and asks whether $\mathcal{I} \models \star\varphi$. Clearly, the above problem remains underspecified as long as we have not specified finite representations for ρ. However, The focus of this paper is not on the computational complexity of $\mathsf{TEMP\_CERTAINTY}(\varphi)$ for particular choices of ρ, but rather on general properties that hold for any ρ. We start with two easy lemmas, stating that $\star\Box\varphi$ and $\Box\star\varphi$ are equivalent, and that $\star$ distributes over $\land$.

Lemma 2. *Let φ be a temporal query in $\mathsf{TL}(\mathcal{C})$. Let ρ be a repair function. Let $\mathcal{I}$ be a temporal database instance and $\ell \in \mathbb{N}_0$. Then, $\mathcal{I}, \ell \models_\rho \star\Box\varphi$ if and only if $\mathcal{I}, \ell \models_\rho \Box\star\varphi$.*

Lemma 3. *Let φ_1, φ_2 be temporal queries in $\mathsf{TL}(\mathcal{C})$. Let ρ be a repair function. Let $\mathcal{I}$ be a temporal database instance and $\ell \in \mathbb{N}_0$. Then, $\mathcal{I}, \ell \models_\rho \star\left(\varphi_1 \land \varphi_2\right)$ if and only if $\mathcal{I}, \ell \models_\rho \star\varphi_1$ and $\mathcal{I}, \ell \models_\rho \star\varphi_2$.*

Finally, we have the following.

Lemma 4. *Let φ be a temporal query in $\mathsf{TL}(\mathcal{C})$. Let ρ be a repair function. Let $\mathcal{I}$ be a temporal database instance and $\ell \in \mathbb{N}_0$. If $\mathcal{I}, \ell \models_\rho \Diamond\star\varphi$, then $\mathcal{I}, \ell \models_\rho \star\Diamond\varphi$.*

IV. Push-Down Theorem

The converse of Lemma 4 does not generally hold: there exists a temporal database instance that satisfies $\star\Diamond\varphi$, but falsifies $\Diamond\star\varphi$. Intuitively, in every repair, φ becomes eventually true, but there is no time at which φ is simultaneously true in all repairs.

We will now introduce a sufficient condition on the repair function ρ under which $\star\Diamond\varphi$ and $\Diamond\star\varphi$ are equivalent. We say that the repair function ρ is *closed under crossover* if for every temporal database instance $\mathcal{I} = \langle I_0, I_1, \ldots\rangle$, if $\mathcal{R} = \langle R_0, R_1, \ldots\rangle$ and $\mathcal{S} = \langle S_0, S_1, \ldots\rangle$ belong to $\rho(\mathcal{I})$, then for every $\ell \in \mathbb{N}_0$, $\langle R_0, R_1, \ldots, R_\ell, S_{\ell+1}, S_{\ell+2}, \ldots\rangle$ belongs to $\rho(\mathcal{I})$.

For atemporal integrity constraints, repair functions that are closed under crossover emerge when one repairs a temporal database instance in a "time by time" manner, where for every time instant ℓ, the snapshot I_ℓ is repaired independently of previous or later times. In Section V, we will discuss some alternate repair assumptions.

Closeness under crossover would make little sense for temporal integrity constraints that span multiple time instants, like *"The city of birth of an employee cannot change."* If $\mathcal{R}$ and $\mathcal{S}$ would store different cities of birth for the same employee, their crossover would be inconsistent. Recall, however, that this paper focuses on atemporal integrity constraints (on temporal databases).

Notice that if $\rho(\mathcal{I})$ contains both $\langle R_1, R_1, R_1, \ldots\rangle$ and $\langle R_2, R_2, R_2, \ldots\rangle$, then ρ cannot be closed under crossover. Indeed, to be closed under crossover, $\rho(\mathcal{I})$ would have to contain $\langle R_1, R_2, R_1, R_2, R_1, R_2, \cdots\rangle$ as well, but the latter sequence is not a legal repair because it violates the *Finite Change Condition*.

Lemma 5 (Synchronization)**.** *Let φ be a temporal query in $\mathsf{TL}(\mathcal{C})$. Let ρ be a repair function that is closed under crossover. Let $\mathcal{I}$ be a temporal database instance and $\ell \in \mathbb{N}_0$. If $\mathcal{I}, \ell \models_\rho \star\Diamond\varphi$, then $\mathcal{I}, \ell \models_\rho \Diamond\star\varphi$.*

Proof: Assume $\mathcal{I}, \ell \models_\rho \star\Diamond\varphi$. That is, for every repair $\mathcal{R} \in \rho(\mathcal{I})$, we have $\mathcal{R}, \ell \models \Diamond\varphi$.

We need to show that for some $m \in \mathbb{N}_0$ such that $m \geq \ell$, we have $\mathcal{I}, m \models_\rho \star\varphi$.

Since the *Finite Change Condition* applies on repairs, for every repair $\mathcal{R}$ of $\mathcal{I}$ (i.e., for every $\mathcal{R} \in \rho(\mathcal{I})$), either

1) there exists $n \in \mathbb{N}_0$ such that for all $k \geq n$, we have $\mathcal{R}, k \models \varphi$; or
2) there exists $n \in \mathbb{N}_0$ such that for all $k \geq n$, we have $\mathcal{R}, k \models \neg\varphi$.

If a repair $\mathcal{R}$ satisfies the first condition, then we say that its *φ-number* is $+\infty$; otherwise its φ-number is the greatest integer s such that $\mathcal{R}, s \models \varphi$ (such integer exists since $\mathcal{R}, \ell \models \Diamond\varphi$). If the φ-number of every repair is equal to $+\infty$, then the desired result holds vacuously. Assume next that the φ-number of some repair is distinct from $+\infty$. Let s be the smallest number that is the φ-number of some repair, and let $\mathcal{S}$ be a repair whose φ-number is equal to s. From $\mathcal{S}, \ell \models \Diamond\varphi$, it follows $s \geq \ell$. We show in the next paragraph that for every repair $\mathcal{R}$ of $\mathcal{I}$, we have $\mathcal{R}, s \models \varphi$, which concludes the proof.

Assume towards a contradiction that there exists a repair $\mathcal{R}$ of $\mathcal{I}$ such that $\mathcal{R}, s \not\models \varphi$. Since the φ-number of $\mathcal{R}$ is at least s, we can assume a smallest integer $r \in \mathbb{N}_0$ such that $r > s$ and $\mathcal{R}, r \models \varphi$. The situation can be depicted as follows:

		s							r	
$\mathcal{S}$	$\cdots$	φ	$\neg\varphi$	$\neg\varphi$		$\cdots$		$\neg\varphi$	$\neg\varphi$	$\cdots$
$\mathcal{R}$	$\cdots$	$\neg\varphi$	$\neg\varphi$	$\neg\varphi$		$\cdots$		$\neg\varphi$	φ	$\cdots$

Let $\mathcal{S} = \langle S_0, S_1, \ldots\rangle$ and $\mathcal{R} = \langle R_0, R_1, \ldots\rangle$. Since ρ is closed under crossover, the sequence $\mathcal{T}$ defined next belongs to $\rho(\mathcal{I})$:

$$\mathcal{T} := \langle R_0, R_1, \ldots, R_{r-1}, S_r, S_{r+1}, S_{r+2}, \cdots\rangle.$$

We show that for all $j \geq s$, we have $\mathcal{T}, j \not\models \varphi$. First, for all $j \geq r$, we can infer $\mathcal{T}, j \not\models \varphi$ from $\mathcal{S}, j \not\models \varphi$, because all our queries are future formulas.

Next, let $j \in \{s, s+1, \ldots, r-1\}$. By Lemma 1, we can assume without loss of generality that $\varphi \equiv \bigwedge_{i=1}^{n} \varphi_i$ where each φ_i is of the form q, $\Diamond\psi$, $\Box q$, or $\Box\Diamond\psi$ with $q \in \mathcal{C}$.

Since $\mathcal{R}, j \not\models \varphi$, we can assume $i \in \{1, \ldots, n\}$ such that $\mathcal{R}, j \not\models \varphi_i$. Then φ_i must be of the form q or $\Box q$. Indeed, if $\varphi_i \equiv \Diamond\psi$ or $\varphi_i \equiv \Box\Diamond\psi$, then from $\mathcal{R}, r \models \varphi_i$ with $r \geq j$, it is correct to conclude $\mathcal{R}, j \models \varphi_i$, a contradiction. So two cases can occur:

1) Case $\varphi_i \equiv q$ with $q \in \mathcal{C}$. Since $\mathcal{R}, j \not\models \varphi_i$, it follows $\mathcal{T}, j \not\models \varphi_i$.
2) Case $\varphi_i \equiv \Box q$ with $q \in \mathcal{C}$. Since $\mathcal{R}, j \not\models \varphi_i$ and $\mathcal{R}, r \models \varphi_i$ with $r \geq j$, it must be the case that for some $k \in \{j, j+1, \ldots, r-1\}$, we have $\mathcal{R}, k \not\models q$. It follows $\mathcal{T}, j \not\models \varphi_i$.

It follows $\mathcal{T}, j \not\models \varphi$ in all cases.

Since $\mathcal{T}$ is a repair of $\mathcal{R}$, we have $\mathcal{T}, \ell \models \Diamond\varphi$. It follows that the φ-number of $\mathcal{T}$ is strictly smaller than s, a contradiction. ■

The main result of this section can now be stated.

Theorem 1 (Push-Down)**.** *Let φ be a temporal query in $\mathsf{TL}(\mathcal{C})$. Let ρ be a repair function that is closed under crossover. Let φ' be the formula obtained from φ by replacing each atom q (i.e., $q \in \mathcal{C}$) with $\star q$. Then, for every temporal database instance $\mathcal{I}$, we have $\mathcal{I} \models_\rho \star\varphi$ if and only if $\mathcal{I} \models_\rho \varphi'$.*

Proof: Immediate from Lemmas 2, 3, and 5. Intuitively, the query $\star\varphi$ can be formulated equivalently by "pushing down" the $\star$ operators until all $\star$ operators are in front of queries in $\mathcal{C}$. ■

Example 2. Continuation of Example 1. If we denote the two queries of $\mathcal{C}$ by q_1 and q_2 respectively, then the temporal query of Example 1 is $\Diamond\,(q_1 \land \Diamond\Box q_2)$. From Theorem 1, it follows that the latter query is true in every repair of some temporal database instance $\mathcal{I}$ if and only if $\mathcal{I}$ satisfies $\Diamond\,((\star q_1) \land \Diamond\Box\star q_2)$, provided that the repair function ρ is closed under crossover. $\Box$

We now discuss the practical interest of Theorem 1. In the past, CQA has mainly focused on atemporal Boolean queries $q \in \mathcal{C}$ (for some query class $\mathcal{C}$) and atemporal integrity constraints: Given an atemporal database I, what is the complexity of determining whether q evaluates to true on all repairs of I? The latter question is tantamount to determining $\mathcal{I}, \ell \models_\rho \star q$ under the assumption that temporal repairs are obtained by repairing snapshots individually. So pushing temporal operators outside the scope of $\star$ allows us to reuse tractability results for CQA obtained in the atemporal case.

Unfortunately, Theorem 1 is no longer valid if we consider temporal operators $\bigcirc$ or **until**, as illustrated next. Assume a binary relation name R whose first position is a key, i.e., $\Sigma = \{\forall x \forall y \forall z\,(R(x,y) \land R(x,z) \rightarrow y = z)\}$. Consider $\mathcal{I} = \langle I_0, I_1, I_2, \ldots\rangle$ where

$$\begin{aligned} I_0 &= \{R(a,1)\} \\ I_1 &= \{R(a,1), R(a,2)\} \\ I_2 &= \{R(a,2)\} \\ I_j &= I_2 \text{ for all } j > 2 \end{aligned}$$

Obviously, $\mathcal{I}$ has two repairs, because one fact has to be deleted from I_1. Significantly, the set containing these two repairs is closed under crossover. Since both repairs satisfy $R(a,1)$ **until** $R(a,2)$, it is correct to write $\mathcal{I} \models \star\,(R(a,1)\ \textbf{until}\ R(a,2))$. However, $\mathcal{I} \not\models (\star R(a,1))\ \textbf{until}\ \star R(a,2)$, because $\mathcal{I}, 1 \not\models \star R(a,1)$ and $\mathcal{I}, 1 \not\models \star R(a,2)$.

Likewise, we have that $\mathcal{I} \models \star\,(\Diamond\,(R(a,1) \land \bigcirc R(a,2)))$, but $\mathcal{I} \not\models \Diamond\,(\star R(a,1)) \land \bigcirc\star R(a,2)$.

V. Repairing in Practice

In this section we first introduce concrete temporal database instances. We then argue that it may be unnatural to repair snapshots individually, and provide a new repair notion that seems more natural.

A. Repairing Snapshots Individually

In the following, we assume that all integrity constraints are *denial constraints*, which are first-order sentences of the form:

$$\neg\exists\vec{y}\,(R_1(\vec{x}_1) \land \cdots \land R_n(\vec{x}_n) \land \varphi)\,,$$

where $n \geq 1$ and φ is a conjunction of built-in predicates such that every variable that occurs in φ occurs in some $\vec{x}_i$. The empty conjunction is understood to be true. Denial constraints may be more customarily expressed in the form:

$$\forall\vec{y}\,(R_1(\vec{x}_1) \land \cdots \land R_n(\vec{x}_n) \rightarrow \neg\varphi)\,.$$

For example, the following denial constraints express that no employee can have two distinct domiciles or work for two distinct companies:

$$\begin{gathered} \forall x \forall y \forall z \forall u \forall v\,(\mathsf{E}(x,y,u) \land \mathsf{E}(x,z,v) \rightarrow y = z) \\ \forall x \forall y \forall z \forall u \forall v\,(\mathsf{E}(x,y,u) \land \mathsf{E}(x,z,v) \rightarrow u = v) \end{gathered}$$

We write $\mathsf{key}(\mathsf{Ename})$ for the set that contains these two denial constraints.

The following definitions are relative to a set Σ of denial constraints. A *repair* of an atemporal database instance I is a maximal (with respect to set inclusion) subset of I that satisfies every denial constraint of Σ. This repair construct is thus by deletion only [2]. Since the empty atemporal database instance satisfies every denial constraint, every atemporal database instance has at least one repair.

The foregoing notion of repair extends to temporal database instances in a straightforward way. A repair of a temporal database instance $\mathcal{I} = \langle I_0, I_1, \ldots\rangle$ is any temporal database instance $\mathcal{R} = \langle R_0, R_1, \ldots\rangle$ such that for each $\ell \in \mathbb{N}_0$, we have that R_ℓ is a repair of I_ℓ. However, as already exemplified in Section I, the aforementioned temporal extension may result in repairs that are implausible.

B. Finite Representation of Temporal Database Instances

This section introduces concrete temporal database instances in which facts are timestamped with time intervals.

An *interval* is a pair $[i,j]$ where $i \in \mathbb{N}_0$ and $j \in \mathbb{N}_0 \cup \{+\infty\}$ such that $i \leq j$. Two intervals $[i_1, j_1]$ and $[i_2, j_2]$ are *disjoint* if there exists no $\ell \in \mathbb{N}_0 \cup \{+\infty\}$ such that both $i_1 \leq \ell \leq j_1$ and $i_2 \leq \ell \leq j_2$.

Let R be a relation name of arity n. A *temporal fact* is an expression $R(\vec{a}, [i,j])$ where $\vec{a}$ is a sequence of constants (of length n) and $[i,j]$ is an interval. A *concrete temporal database instance* is a finite set $\mathbf{db}$ of temporal facts such that for all $R(\vec{a}_1, [i_1, j_1])$, $R(\vec{a}_2, [i_2, j_2])$ in $\mathbf{db}$, if $\vec{a}_1 = \vec{a}_2$, then the intervals $[i_1, j_1]$ and $[i_2, j_2]$ are disjoint.

If $\mathbf{db}$ is a concrete temporal database instance, then $[\![\mathbf{db}]\!]$ denotes the sequence $\langle I_0, I_1, I_2, \ldots\rangle$ such that for all $\ell \in \mathbb{N}_0$,

$$I_\ell = \{R(\vec{a}) \mid \exists i \exists j : R(\vec{a}, [i,j]) \in \mathbf{db} \text{ and } i \leq \ell \leq j\}.$$

Corollary 1. *If* $\mathbf{db}$ *is concrete temporal database instance, then* $[\![\mathbf{db}]\!]$ *is a temporal database instance.*

Proof: Since $\mathbf{db}$ is a finite set, there must exist $m \in \mathbb{N}_0$ such that $I_m = I_{m+1} = I_{m+2} = \cdots$. ■

A concrete temporal database instance $\mathbf{db}$ is said to be *coalesced* if there exists no temporal database $\mathbf{db}'$ such that $|\mathbf{db}'| < |\mathbf{db}|$ and $[\![\mathbf{db}']\!] = [\![\mathbf{db}]\!]$. Intuitively, the coalesced representation minimizes the number of temporal facts. Figure 2 (left) shows a concrete temporal database instance that is coalesced.

E	Ename	LivesIn	WorksFor	Time
	Ed	Paris	Google	$[0, +\infty]$
	Ed	London	Google	$[5, +\infty]$
	An	London	Google	$[9, +\infty]$

$\mathcal{R}'$	Ename	LivesIn	WorksFor	Time
	Ed	Paris	Google	$[0, 13]$
	Ed	London	Google	$[14, 17]$
	Ed	Paris	Google	$[18, +\infty]$
	An	London	Google	$[9, +\infty]$

Figure 2. Concrete temporal database instance $\mathbf{db}_0$ (left) and a finite representation of a repair of $[\![\mathbf{db}_0]\!]$ (right).

Consistency carries over from temporal database instances to concrete temporal database instances in the natural way: a concrete temporal database instance $\mathbf{db}$ is consistent (with respect to a set Σ of integrity constraints) if $[\![\mathbf{db}]\!]$ is consistent.

C. Repairing at the Concrete Level

The repair notion of Section V-A is theoretically clean and elegant, but may lead to undesirable repairs, as already illustrated in Section I. A more detailed example is given next.

Example 3. Let $\mathbf{db}_0$ be the concrete temporal database instance of Fig. 2, and let $\mathcal{I}_0 = [\![\mathbf{db}_0]\!]$. Assume constraint $\mathsf{key}(\mathsf{Ename})$.

For any non-negative integer n, there exists a repair of $\mathcal{I}_0$ recording that Ed made n moves between Paris and London. One such repair is shown in Fig. 2 (right). One may criticize this repair for being ill-founded, because the original database provides no evidence that something happened between times 13 and 14, or between 17 and 18.

Note incidentally that by the *Finite Change Condition*, there is no repair where Ed made infinitely many moves between Paris and London. $\Box$

We now formalize the idea that some repairs are more plausible than others. Let $\mathbf{db}$ be a concrete temporal database instance. A *conflict set* of $\mathbf{db}$ with respect to Σ is a minimal (with respect to set inclusion) subset $\mathbf{cs}$ of $\mathbf{db}$ that is inconsistent with respect to Σ. We say that $\mathbf{db}$ is *factorized* with respect to Σ if for every conflict set $\mathbf{cs}$ of $\mathbf{db}$, all temporal facts of $\mathbf{cs}$ have identical time intervals.

For example, the concrete temporal database instance of Fig. 2 is not factorized, because the two temporal facts about Ed form a conflict set but have unequal time intervals. The temporal database instance of Fig. 3 is factorized.

It can be easily seen that for every temporal database instance $\mathcal{I}$, there exists a concrete temporal database instance $\mathbf{db}$ that is factorized with respect to Σ such that $[\![\mathbf{db}]\!] = \mathcal{I}$. Moreover, the following holds.

Theorem 2. *Let Σ be a set of denial constraints, and let $\mathcal{I}$ be a temporal database instance. There exists a unique minimal (with respect to cardinality) concrete temporal database instance $\mathbf{db}$ such that $[\![\mathbf{db}]\!] = \mathcal{I}$ and $\mathbf{db}$ is factorized with respect to Σ.*

Proof: Let $\mathbf{db}_1$ and $\mathbf{db}_2$ be two minimal (with respect to cardinality) concrete temporal databases, both factorized with respect to Σ, such that $[\![\mathbf{db}_1]\!] = [\![\mathbf{db}_2]\!] = \mathcal{I}$. We show $\mathbf{db}_1 \subseteq \mathbf{db}_2$ (the opposite inclusion is proved in the same way). Assume $\mathbf{db}_1$ contains $R(\vec{a}, [i, j])$ (possibly $j = +\infty$). Since $[\![\mathbf{db}_1]\!] = [\![\mathbf{db}_2]\!]$, there exists a (unique) temporal fact $R(\vec{a}, [g, h])$ in $\mathbf{db}_2$ such that $g \leq i \leq h$. We show $i = g$ (the equality $j = h$ is proved in the same way). Assume towards a contradiction $g < i$. Then, $g \leq i - 1 < i \leq h$.

From $[\![\mathbf{db}_1]\!] = [\![\mathbf{db}_2]\!]$, it follows that for some $k \in \mathbb{N}_0$, $\mathbf{db}_1$ contains $R(\vec{a}, [k, i-1])$.

Let $\sim$ be the subset of $\mathbf{db}_1 \times \mathbf{db}_1$ that is the reflexive transitive closure of:

$F \sim G$ if F and G belong to a same conflict set of $\mathbf{db}_1$ with respect to Σ.

Let A be the subset of $\mathbf{db}_1$ that contains each fact $S(\vec{b}, [k, i-1]) \in \mathbf{db}_1$ such that $S(\vec{b}, [k, i-1]) \sim R(\vec{a}, [k, i-1])$. Let B be the subset of $\mathbf{db}_1$ that contains each fact $S(\vec{b}, [i, j]) \in \mathbf{db}_1$ such that $S(\vec{b}, [i, j]) \sim R(\vec{a}, [i, j])$. Notably, A contains $R(\vec{a}, [k, i-1])$ and B contains $R(\vec{a}, [i, j])$. Since $R(\vec{a}, [k, i-1])$ and $R(\vec{a}, [i, j])$ are not coalesced in $\mathbf{db}_1$ and since $\mathbf{db}_1$ is minimal, one of the following must occur:

1) there exists $S(\vec{b}, [k, i-1]) \in A$ such that $S(\vec{b}, [i, j]) \notin B$; or
2) there exists $S(\vec{b}, [i, j]) \in B$ such that $S(\vec{b}, [k, i-1]) \notin A$.

Now it is not hard to see that this contradicts with $R(\vec{a}, [g, h]) \in \mathbf{db}_2$ where $g \leq i - 1 < i \leq h$. $\blacksquare$

If $\mathcal{I}$ is a temporal database instance, then we denote by $\mathsf{fac}_\Sigma(\mathcal{I})$ the minimal (with respect to cardinality) concrete temporal database instance $\mathbf{db}$ such that $[\![\mathbf{db}]\!] = \mathcal{I}$ and $\mathbf{db}$ is factorized with respect to Σ.

The following lemma has an easy proof.

Lemma 6. *Let Σ be a set of denial constraints, and let $\mathcal{I}$ be a temporal database instance. If $\mathbf{r}$ is a maximal (with respect to set inclusion) consistent subset of $\mathsf{fac}_\Sigma(\mathcal{I})$, then $[\![\mathbf{r}]\!]$ is a repair of $\mathcal{I}$.*

If $\mathbf{r}$ is a maximal consistent subset of $\mathsf{fac}_\Sigma(\mathcal{I})$, then $[\![\mathbf{r}]\!]$ will be called a *persistent repair* of $\mathcal{I}$ with respect to Σ. Intuitively, in a persistent repair, values of attributes persist if no new conflict emerges. Thus, Lemma 6 tells us that (finite representations of) persistent repairs are maximal (with respect to cardinality) consistent subsets of the minimal factorized concrete temporal database instance.

E

Ename	LivesIn	WorksFor	Time
Ed	Paris	Google	$[0,4]$
Ed	Paris	Google	$[5,+\infty]$
Ed	London	Google	$[5,+\infty]$
An	London	Google	$[9,+\infty]$

Figure 3. Concrete temporal database instance $\mathbf{db}_1$ such that $[\![\mathbf{db}_1]\!] = [\![\mathbf{db}_0]\!]$. $\mathbf{db}_1$ is a minimal concrete temporal database that is factorized with respect to key(Ename).

$\mathcal{R}_1$

Ename	LivesIn	WorksFor	Time
Ed	Paris	Google	$[0,+\infty]$
An	London	Google	$[9,+\infty]$

$\mathcal{R}_2$

Ename	LivesIn	WorksFor	Time
Ed	Paris	Google	$[0,4]$
Ed	London	Google	$[5,+\infty]$
An	London	Google	$[9,+\infty]$

Figure 4. Finite representation of the two persistent repairs of $[\![\mathbf{db}_0]\!]$.

Figure 4 shows the two persistent repairs for our running example: in $\mathcal{R}_1$, Ed stayed forever in Paris; in $\mathcal{R}_2$, Ed moved once from Paris to London.

D. CQA with Respect to Persistent Repairs

Let Σ be a set of denial constraints. It is natural to define the repair function ρ such that for every temporal database instance $\mathcal{I}$, we let $\rho(\mathcal{I})$ be equal to the set of persistent repairs of $\mathcal{I}$ with respect to Σ. In this way, we have arrived at a realistic setting for studying the complexity of $\mathsf{TEMP\_CERTAINTY}(\varphi)$ for queries $\varphi \in \mathsf{TL}(\mathcal{C})$. While this study is left for future work, we point out here that, unfortunately, Theorem 1 fails under such repair function, as illustrated next.

Let q_0 be the query

$$q_0 \quad : \quad \exists s\,(\mathsf{S}(\text{'Ed'}, s) \wedge \mathsf{S}(\text{'An'}, s))\,,$$

asking whether Ed and An earn the same salary. The temporal database instance of Fig. 5 (left) has two persistent repairs $\mathcal{R}_1$ and $\mathcal{R}_2$, whose finite representation is shown in Fig. 5 (right). Both persistent repairs satisfy $\Diamond q_0$. However, there exists no time instant ℓ such that q_0 is true at ℓ in both $\mathcal{R}_1$ and $\mathcal{R}_2$. Note also that ρ is not closed under crossover.

VI. Related Work

The paradigm of consistent query answering (CQA) under integrity constraints has gained an increasing research interest ever since its introduction [1]. A monograph on the topic is [3]. A significant line of research is to compute consistent query answers by query rewriting: Given a query q in some "source" language, find a query q' in some fixed "target" language such that for every database I, the consistent answer to q on I (i.e., the intersection of the answers to q on all repairs) is equal to the (standard) answer to q' on I. In several studies (e.g., [4], [5]), the source language is restricted to conjunctive queries, and the target language is first-order logic. Theorem 1 in the current paper falls in this line of research: for queries $\varphi \in \mathsf{TL}(\mathcal{C})$, the theorem provides a rewriting for $\star\varphi$, provided that rewritings of $\star q$ are known for $q \in \mathcal{C}$. Some studies have focused on CQA in the spatio-temporal domain [6], [7], albeit in settings that are quite different from the current paper.

The foundations of temporal database systems are given in [8]. Important is the distinction between (abstract) temporal databases and their concrete representations. It is common to define semantics of integrity constraints and queries on the abstract level. In this respect, it is worth noticing that our notion of persistent repair relies on the concrete representation. Temporal integrity constraints have been an active research topic for several decades; see [9] and the references therein.

Persistent repairs are reminiscent of temporal granularity [10], [11] in the following way. Attribute values are often constrained to not change within some time granule (e.g., salaries do not change within a calendar month); in the same vein, persistent repairs avoid attribute values to change at arbitrary time instants.

VII. Conclusion and Future Work

This article reflected on ways of extending consistent query answering (CQA) to temporal databases, starting with the most obvious case where all integrity constraints are atemporal. At first we assumed that repairing at any one time instant does not depend on past or future times. We showed that this assumption, formalized by the *Closeness Under Crossover* condition, results in some nice rewriting properties for a particular class of temporal queries. We then introduced a novel repair construct which captures the intuition that repairs must not add spurious changes at arbitrarily picked time instants. Unfortunately, this seemingly more realistic repair notion does not satisfy all rewriting properties that apply if *Closeness Under Crossover* holds.

This work can be extended in multiple ways. We showed that the Push-Down Theorem (Theorem 1) does not hold if $\bigcirc$ or **until** are allowed in temporal queries. It would be interesting to explore alternate rewriting techniques in the presence of these temporal operators. For example, it seems that $\star\Diamond\,(q_1 \wedge \bigcirc q_2)$ is equivalent to $\Diamond\,(\star q_1 \wedge \star\,(q_1 \vee q_2)\ \textbf{until}\ \star q_2)$. Notice that the latter query uses disjunction, which is currently not in $\mathsf{TL}(\mathcal{C})$. In general, given a source language with a choice of temporal operators ($\Diamond$, $\Box$, $\bigcirc$, **until**) and Boolean connectives ($\wedge$, $\vee$, $\neg$), determine a minimal target language such that for every query φ in the source language, $\star\varphi$ can be expressed in the target language (extended with $\star q$ for every $q \in \mathcal{C}$). These questions merit to be studied for generic repair functions ρ, but also for the specific repair functions that map each temporal database instance to its set of persistent repairs.

S

Ename	Sal	Time
Ed	10K	$[1, 61]$
Ed	12K	$[1, 61]$
An	10K	$[1, 30]$
An	12K	$[31, 61]$

$\mathcal{R}_1$

Ename	Sal	Time
Ed	10K	$[1, 61]$
An	10K	$[1, 30]$
An	12K	$[31, 61]$

$\mathcal{R}_2$

Ename	Sal	Time
Ed	12K	$[1, 61]$
An	10K	$[1, 30]$
An	12K	$[31, 61]$

Figure 5. Concrete temporal database instance (left) and its two persistent repairs (right).

Our notion of persistent repair was defined for atemporal integrity constraints; it remains an open question how to extend this notion to constraints that are truly temporal, i.e., that refer to more than one time instant.

Acknowledgment

Jan Chomicki is supported in part by NSF grant IIS-1524469.

References

[1] M. Arenas, L. E. Bertossi, and J. Chomicki, "Consistent query answers in inconsistent databases," in *Proceedings of the Eighteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, May 31 - June 2, 1999, Philadelphia, Pennsylvania, USA*, V. Vianu and C. H. Papadimitriou, Eds. ACM Press, 1999, pp. 68–79.

[2] J. Chomicki and J. Marcinkowski, "Minimal-change integrity maintenance using tuple deletions," *Inf. Comput.*, vol. 197, no. 1-2, pp. 90–121, 2005.

[3] L. E. Bertossi, *Database Repairing and Consistent Query Answering*, ser. Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2011.

[4] P. Koutris and J. Wijsen, "The data complexity of consistent query answering for self-join-free conjunctive queries under primary key constraints," in *Proceedings of the 34th ACM Symposium on Principles of Database Systems, PODS 2015, Melbourne, Victoria, Australia, May 31 - June 4, 2015*, T. Milo and D. Calvanese, Eds. ACM, 2015, pp. 17–29.

[5] M. Bienvenu and R. Rosati, "Tractable approximations of consistent query answering for robust ontology-based data access," in *IJCAI 2013, Proceedings of the 23rd International Joint Conference on Artificial Intelligence, Beijing, China, August 3-9, 2013*, F. Rossi, Ed. IJCAI/AAAI, 2013.

[6] M. A. Rodríguez, L. E. Bertossi, and M. C. Marileo, "Consistent query answering under spatial semantic constraints," *Inf. Syst.*, vol. 38, no. 2, pp. 244–263, 2013.

[7] F. Parisi and J. Grant, "Repairs and consistent answers for inconsistent probabilistic spatio-temporal databases," in *Scalable Uncertainty Management - 8th International Conference, SUM 2014, Oxford, UK, September 15-17, 2014. Proceedings*, ser. Lecture Notes in Computer Science, U. Straccia and A. Calì, Eds., vol. 8720. Springer, 2014, pp. 265–279.

[8] J. Chomicki and D. Toman, "Temporal databases," in *Handbook of Temporal Reasoning in Artificial Intelligence*, ser. Foundations of Artificial Intelligence, M. Fisher, D. M. Gabbay, and L. Vila, Eds. Elsevier, 2005, vol. 1, pp. 429–467.

[9] J. Wijsen, "Temporal integrity constraints," in *Encyclopedia of Database Systems*, L. Liu and M. T. Özsu, Eds. Springer US, 2009, pp. 2976–2982.

[10] I. A. Goralwalla, Y. Leontiev, M. T. Özsu, D. Szafron, and C. Combi, "Temporal granularity: Completing the puzzle," *J. Intell. Inf. Syst.*, vol. 16, no. 1, pp. 41–63, 2001.

[11] C. Bettini, X. S. Wang, and S. Jajodia, "Temporal granularity," in *Encyclopedia of Database Systems*, L. Liu and M. T. Özsu, Eds. Springer US, 2009, pp. 2968–2973.

[12] L. Liu and M. T. Özsu, Eds., *Encyclopedia of Database Systems*. Springer US, 2009.