Determining the Currency of Data

Wenfei Fan
University of Edinburgh &
Harbin Institute of Technology
wenfei@inf.ed.ac.uk

Floris Geerts
School of Informatics
University of Edinburgh
fgeerts@inf.ed.ac.uk

Jef Wijsen
Institut d'Informatique
Université de Mons
jef.wijsen@umons.ac.be

Abstract

Data in real-life databases become obsolete rapidly. One often finds that multiple values of the same entity reside in a database. While all of these values were once correct, most of them may have become stale and inaccurate. Worse still, the values often do not carry reliable timestamps. With this comes the need for studying data currency, to identify the current value of an entity in a database and to answer queries with the current values, in the absence of timestamps.

This paper investigates the currency of data. (1) We propose a model that specifies partial currency orders in terms of simple constraints. The model also allows us to express what values are copied from other data sources, bearing currency orders in those sources, in terms of copy functions defined on correlated attributes. (2) We study fundamental problems for data currency, to determine whether a specification is consistent, whether a value is more current than another, and whether a query answer is certain no matter how partial currency orders are completed. (3) Moreover, we identify several problems associated with copy functions, to decide whether a copy function imports sufficient current data to answer a query, whether such a function copies redundant data, whether a copy function can be extended to import necessary current data for a query while respecting the constraints, and whether it suffices to copy data of a bounded size. (4) We establish upper and lower bounds of these problems, all matching, for combined complexity and data complexity, and for a variety of query languages. We also identify special cases that warrant lower complexity.

Categories and Subject Descriptors: H.2.3 [**Information Systems**]: Database Management – *Languages*; F.4.1 [**Mathematical Logic and Formal Languages**]: Mathematical Logic — *Computational Logic*

General Terms: Languages, Theory, Design.

1. Introduction

The quality of data in a real-life database quickly degenerates over time. Indeed, it is estimated that " 2% of records in a customer file become obsolete in one month" [15]. That is, in a database of 500 000 customer records, 10 000 records may go stale per month, 120 000 records per year, and within two years about 50% of all the records may be obsolete. In light of this, we often find that multiple values of the same entity reside in a database, which were *once correct*, *i.e.,* they were true values of the entity at some time. However, most of them have become *obsolete* and *inaccurate*. As an example from daily life, when one moves to a new address, her bank may retain her old address, and worse still, her credit card bills may still be sent to her old address for quite some time (see, *e.g.,* [22] for more examples). Stale data is one of the central problems to data quality. It is known that dirty data costs US businesses 600 billion USD each year [15], and stale data accounts for a large part of the losses.

	FN	LN	address	salary	status
s_1:	Mary	Smith	2 Small St	50k	single
s_2:	Mary	Dupont	10 Elm Ave	50k	married
s_3:	Mary	Dupont	6 Main St	80k	married
s_4:	Bob	Luth	8 Cowan St	80k	married
s_5:	Robert	Luth	8 Drum St	55k	married

(a) Relation Emp

	dname	mgrFN	mrgLN	mgrAddr	budget
t_1:	R&D	Mary	Smith	2 Small St	6500k
t_2:	R&D	Mary	Smith	2 Small St	7000k
t_3:	R&D	Mary	Dupont	6 Main St	6000k
t_4:	R&D	Ed	Luth	8 Cowan St	6000k

(b) Relation Dept

Figure 1: A company database

This highlights the need for studying the *currency of data*, which aims to identify the current values of entities in a database, and to answer queries with the current values.

The question of data currency would be trivial if all data values carried valid timestamps. In practice, however, one often finds that timestamps are unavailable or imprecise [34]. Add to this the complication that data values are often copied or imported from other sources [2, 12, 13], which may not support a uniform scheme of timestamps. These make it challenging to identify the current values.

Not all is lost. One can often deduce currency orders from the semantics of the data. Moreover, data copied from other sources inherit currency orders from those sources. Taken together, these often provide sufficient current values of the data to answer certain queries, as illustrated below.

Example 1.1: Consider two relations of a company shown in Fig. 1. Each Emp tuple is an employee record with name, address, salary and marital status. A Dept tuple specifies the name, manager and budget of a department. Records in these relations may be stale, and do not carry timestamps. By entity identification techniques (see, *e.g.,* [16]), we know

PODS'11, June 13–15, 2011, Athens, Greece.

that tuples s_1, s_2 and s_3 refer to the same employee Mary, but s_4 and s_5 represent different people distinct from Mary. Consider the following queries posed on these relations.

(1) Query Q_1 is to find Mary's current salary. No timestamps are available for us to tell which of 50k or 80k is more current. However, we may know that the salary of each employee in the company does *not* decrease, as commonly found in real world. This yields currency orders $s_1 \prec_{\mathsf{salary}} s_3$ and $s_2 \prec_{\mathsf{salary}} s_3$, *i.e.,* $s_3[\mathsf{salary}]$ is more current than both $s_1[\mathsf{salary}]$ and $s_2[\mathsf{salary}]$. Hence the answer to Q_1 is 80k.

(2) Query Q_2 is to find Mary's current last name. We can no longer answer Q_2 as above. Nonetheless, we may know the following: (a) marital status can only change from single to married and from married to divorced; but not from married to single; and (b) Emp tuples with the most current marital status also contain the most current last name. Therefore, $s_1 \prec_{\mathsf{LN}} s_2$ and $s_1 \prec_{\mathsf{LN}} s_3$, and the answer to Q_2 is Dupont.

(3) Query Q_3 is to find Mary's current address. We may know that Emp tuples with the most current status or salary contain the most current address. Putting this and (1) above together, we know that the answer to Q_3 is "6 Main St".

(4) Finally, query Q_4 is to find the current budget of department R&D. Again no timestamps are available for us to evaluate the query. However, we may know the following: (a) Dept tuples t_1 and t_2 have copied their mgrAddr values from $s_1[\mathsf{address}]$ in Emp; similarly, t_3 has copied from s_3, and t_4 from s_4; and (b) in Dept, tuples with the most current address also have the most current budget. Taken together, these tell us that $t_1 \prec_{\mathsf{budget}} t_3$ and $t_2 \prec_{\mathsf{budget}} t_3$. Observe that we do not know which budget in t_3 or t_4 is more current. Nevertheless, in either case the most current budget is 6000k, and hence it is the answer to Q_4. □

These suggest that we give a full treatment of data currency, and answer the following questions. How should we specify currency orders on data values in the absence of timestamps but in the presence of copy relationships? When currency orders are only partly available, can we decide whether an attribute value is more up-to-date than another? How can we answer a query with only current data in a database? To answer a query, do we need to import current data from another source, and what to copy? The ability to answer these questions may provide guidance for practitioners to decide, *e.g.,* whether the answer to a query is corrupted by stale data, or what copy functions are needed.

A model for data currency. To answer these questions, we approach data currency based on the following.

(1) For each *attribute* A of a relation D, we assume an (implicit) currency order $\prec_A$ on its tuples such that for tuples t_1 and t_2 in D that represent the same real-world entity, $t_1 \prec_A t_2$ indicates that t_2 is more up-to-date than t_1 in the A attribute value. Here $\prec_A$ is not a total order since in practice, currency information is only partially available. Note that for distinct attributes A and B, we may have $t_1 \prec_A t_2$ and $t_2 \prec_B t_1$, *i.e.,* there may be no single tuple that is most up-to-date in all attribute values.

(2) We express additional currency relationships as denial constraints [3, 7], which are simple universally quantified FO sentences that have been used to improve the consistency of data. We show that the same class of constraints also suffices to express currency semantics commonly found in practice. For instance, all the currency relations we have seen in Example 1.1 can be expressed as denial constraints.

(3) We define a copy relationship from relation D_j to D_k in terms of a partial mapping, referred to as a *copy function*. It specifies what attribute values in D_j have been copied from D_k along with their currency orders in D_k. It also assures that correlated attributes are copied together. As observed in [2, 12, 13], copy functions are common in real world, and can be automatically discovered.

Putting these together, we consider $\mathbf{D} = (D_1, \ldots, D_n)$, a collection of relations such that (a) each D_j has currency orders partially defined on its tuples for each attribute, indicating *available* currency information; (b) each D_j satisfies a set Σ_j of denial constraints, which expresses currency orders derived from the semantics of the data; and (c) for each pair D_j, D_k of relations, there are possibly copy functions defined on them, which import values from one to another.

We study *consistent completions* D_j^c of D_j, which extend $\prec_A$ in D_j to a total order on all tuples pertaining to the same entity, such that D_j^c satisfies Σ_j and constraints imposed by the copy functions. One can construct from D_j^c the *current tuple* for each entity *w.r.t.* $\prec_A$, which contains the entity's most current A value for each attribute A. This yields the *current instance* of D_j^c consisting of only the current tuples of the entities in D_j, from which currency orders are removed. We evaluate a query Q on current instances of relations in $\mathbf{D}$, without worrying about currency orders. We study *certain current answers* to Q in $\mathbf{D}$, *i.e.,* tuples that are the answers to Q in all consistent completions of $\mathbf{D}$.

Reasoning about data currency. We study fundamental problems for data currency. (a) The *consistency problem* is to determine, given denial constraints Σ_j imposed on each D_j and copy functions between these relations, whether there exist consistent completions of every D_j, *i.e.,* whether the specification makes sense. (b) The *certain ordering problem* is to decide whether a currency order is contained in all consistent completions. (c) The *deterministic current instance problem* is to determine whether the current instance of each relation remains unchanged for all consistent completions. The ability to answer these questions allows us to determine whether an attribute value is certainly more current than another, and to identify the current value of an entity. (d) The *certain current query answering problem* is to decide whether a tuple t is a certain current answer to a query Q, *i.e.,* it is certainly computed using current data.

Currency preserving copy functions. It is natural to ask what values should be copied from one data source to another in order to answer a query. To characterize this intuition we introduce a notion of currency preservation. Consider data sources $\mathbf{D} = (D_1, \ldots, D_p)$ and $\mathbf{D}' = (D'_1, \ldots, D'_q)$, each consisting of a collection of relations with denial constraints imposed on them. Consider copy functions $\overline{\rho}$ from relations in $\mathbf{D}'$ to those in $\mathbf{D}$. For a query Q posed on $\mathbf{D}$, we say that $\overline{\rho}$ is *currency preserving* if no matter how we extend $\overline{\rho}$ by copying from $\mathbf{D}'$ more values of those entities in $\mathbf{D}$, the certain current answers to Q in $\mathbf{D}$ remain unchanged. In other words, $\overline{\rho}$ has already imported data values needed for computing certain current answers to Q.

We identify several problems associated with currency-preserving copy functions. (a) The *currency preservation problem* is to determine, given Q, $\overline{\rho}$, $\mathbf{D}$, $\mathbf{D}'$ and their denial

constraints, whether $\overline{\rho}$ is currency preserving for Q. Intuitively, we want to know whether we need to extend $\overline{\rho}$ in order to answer Q. (b) The *minimal copying problem* is to decide whether $\overline{\rho}$ is minimal among all currency-preserving copy functions for Q, *i.e.,* $\overline{\rho}$ copies the least amount of data. This helps us inspect whether $\overline{\rho}$ copies unnecessary data. (c) The *existence problem* is to determine whether $\overline{\rho}$ can be extended to be currency preserving for Q. (d) Moreover, the *bounded copying* problem is to decide whether there exists such an extension that imports additional data of a bounded size. Intuitively, we want to find currency-preserving copy functions that import as few data values as possible.

Complexity results. We provide *combined complexity* and *data complexity* of all the problems stated above. For the combined complexity of the problems that involve queries, we investigate the impact of various query languages, including conjunctive queries (CQ), unions of conjunctive queries (UCQ), positive existential FO ($\exists$FO$^+$) and FO. We establish upper and lower bounds of these problems, *all matching*, ranging over $O(1)$, NP, coNP, Δ_2^p, Π_2^p, Σ_2^p, Δ_3^p, Π_3^p, Σ_3^p, Σ_4^p and PSPACE. We find that most of the problems are intractable. In light of this, we also identify special practical cases with lower complexity, some in PTIME. We also study the impact of denial constraints. For example, in the absence of denial constraints, the certain current query answering problem is in PTIME for SP queries (CQ queries without "join"), but it becomes intractable when denial constraints are present, even when the constraints are fixed.

This work is a first step towards a systematic study of data currency in the absence of reliable timestamps but in the presence of copy relationships. The results help practitioners specify data currency, analyze query answers and design copy functions. We also provide a complete picture of complexity bounds for important problems associated with data currency and copy functions, which are proved by using a variety of reductions and by providing (PTIME) algorithms.

Related work. There has been a host of work on temporal databases (see, *e.g.,* [8, 30] for surveys). Temporal databases provide support for valid time, transaction time, or both. They assume the availability of timestamps, and refer to "now" by means of current-time variables [9, 14]. Dynamic and temporal integrity constraints allow to restrict the set of legal database evolutions. Our currency model differs from temporal data models in several respects. We do not assume explicit timestamps. Nevertheless, if such timestamps are present, they can be related to currency by means of denial constraints. Unlike temporal databases that timestamp entire tuples, our model allows that different values within the same tuple have distinct currencies. That is, the same tuple can contain an up-to-date value for one attribute, and an outdated value for another attribute.

Since currency orders are different from temporal orders used in temporal databases, our currency (denial) constraints differ from traditional temporal constraints. Currency constraints can sometimes be derived from temporal constraints. For example, when salaries are constrained to be non-decreasing, we can express that the highest salary is the most current one. Also, our copy functions can require certain attributes to be copied together when these attributes cannot change independently, as for example expressed by the dynamic functional dependencies in [33].

Closer to this work are [31, 24, 25, 20] on querying indefinite data. In [31], the evaluation of CQ queries is studied on data that is linearly ordered but only provides a partial order. The problem studied there is similar to (yet different from) certain current query answering. An extension of conditional tables [19, 21] is proposed in [24] to incorporate indefinite temporal information, and in that setting, the complexity bounds for FO query evaluation are provided in [25]. Recently the non-emptiness problem for datalog on linear orders is investigated in [20]. However, none of these considers copying data from external sources, or the analyses of certain ordering and currency-preserving copy functions. In addition, we answer queries using current instances of relations, which are normal relations without (currency) ordering. This semantics is quite different from its counterparts in previous work. We also consider denial constraints and copy functions, which are not expressible in CQ or datalog studied in [31, 20]. In contrast to our work, [24, 25] assume explicit timestamps, while we use denial constraints to specify data currency. To encode denial constraints in extended conditional tables of [24, 25], an exponential blowup is inevitable. Because of these reasons, the results of [31, 24, 25, 20] cannot carry over to our setting, and vice versa.

There has also been a large body of work on the temporal constraint satisfaction problem (TCSP), which is to find a valuation of temporal variables that satisfies a set of temporal constraints (see, *e.g.,* [4, 29]). It differs from our consistency problem in that it considers neither completions of currency orders that satisfy denial constraints, nor copy relationships. Hence the results for TCSP are not directly applicable to our consistency problem, and vice versa.

Copy relationships between data sources have recently been studied in [2, 12, 13]. The previous work has focused on automatic discovery of copying dependencies and functions. Copy relationships are also related to data provenance, which studies propagation of annotations in data transformations and updates (see [5, 6] for recent surveys on data provenance). However, to the best of our knowledge, no previous work has studied currency-preserving copy functions and their associated problems.

Denial constraints have proved useful in detecting data inconsistencies and data repairing (see, *e.g.,* [3, 7]). We adopt the same class of constraints to specify the currency of data, so that data currency and consistency could be treated in a uniform logical framework. Denial constraints can also be automatically discovered, along the same lines as data dependency profiling (see, *e.g.,* [17]).

The study of data currency is also related to research on incomplete information (see [32] for a survey), when missing data concerns data currency. In contrast to that line of work, we investigate how to decide whether a value is more current than another, and study the properties of copy functions. We use denial constraints to specify data currency, which are, as remarked earlier, more succinct than, *e.g.,* C-tables and V-tables for representing incomplete information [19, 21]. In addition, we evaluate queries using current instances, a departure from the study of incomplete information.

Certain query answers have been studied in data integration and exchange. In data integration, for a query Q posed on a global database D_G, it is to find the certain answers to Q over all data sources that are consistent with D_G *w.r.t.* view definitions (see *e.g.,* [27]). In data exchange, it is to find the certain answers to a query over all target

databases generated from data sources via schema mapping (see [23]). In contrast, we consider certain answers to a query over all completions of currency orders, which satisfy denial constraints and constraints from copy functions. Certain current query answering is also different from consistent query answering (see, *e.g.,* [3, 7]), which is to find certain answers to a query over all *repairs* of a database and does not distinguish between stale and current data in the repairs. Finally, whereas it may be possible to model our setting as a data exchange scenario with built-in constraints [11], our complexity results do not follow gratuitously and a careful analysis of the chase is required in this setting.

Organization. Section 2 presents the data currency model. Section 3 states its related problems. Section 4 establishes the complexity bounds of those problems. Section 5 introduces the notion of currency preservation and its fundamental problems, followed by their complexity analysis in Section 6. Section 7 summarizes the main results of the paper.

2. Data Currency

We introduce a model for specifying data currency. A specification consists of (a) partial currency orders, (b) denial constraints, and (c) copy functions. We first present these notions, and then study consistent completions of currency orders. Finally, we show how queries are answered on current instances that are derived from these completions.

Data with partial currency orders. A relation schema is specified as $R=(\mathsf{EID}, A_1, \ldots, A_n)$, where EID denotes entity id that identifies tuples pertaining to the same entity, as introduced by Codd [10]. EID values can be obtained using entity identification techniques (*a.k.a.* record linkage, matching and data deduplication; see, *e.g.,* [16]). A finite instance D of R is referred to as a *normal instance of* R.

A *temporal instance* D_t *of* R is given as $(D, \prec_{A_1}, \ldots, \prec_{A_n})$, where each $\prec_{A_i}$ is a strict partial order on D such that for tuples t_1 and t_2 in D, $t_1 \prec_{A_i} t_2$ implies $t_1[\mathsf{EID}]=t_2[\mathsf{EID}]$. We call $\prec_{A_i}$ the *currency order for attribute* A_i. Recall that a strict partial order is irreflexive and transitive, and therefore asymmetric. Intuitively, if $t_1 \prec_{A_i} t_2$, then t_1 and t_2 refer to *the same entity*, and t_2 contains a *more current* A_i-value for that entity than t_1, *i.e.,* t_2 is more current than t_1 *in attribute* A_i. A currency order $\prec_{A_i}$ is empty when no currency information is known for attribute A_i.

A *completion* of D_t is a temporal instance $D_t^c=(D, \prec^c_{A_1}, \ldots, \prec^c_{A_n})$ of R, such that for each $i\in[1,n]$, (1) $\prec_{A_i} \subseteq \prec^c_{A_i}$, and (2) for all $t_1, t_2 \in D$, t_1 and t_2 are comparable under $\prec^c_{A_i}$ iff $t_1[\mathsf{EID}]=t_2[\mathsf{EID}]$. The latter condition implies that $\prec^c_{A_i}$ induces a total order on tuples that refer to the same entity, while tuples representing distinct entities are not comparable under $\prec^c_{A_i}$. We call $\prec^c_{A_i}$ a *completed currency order*.

Denial constraints. We use denial constraints [3, 7] to specify additional currency information derived from the semantics of data, which enriches $\prec_{A_i}$. A *denial constraint* φ for R is a universally quantified FO sentence of the form:

$$\forall t_1, \ldots, t_k : R \big(\bigwedge_{j\in[1,k]} (t_1[\mathsf{EID}]=t_j[\mathsf{EID}] \wedge \psi) \rightarrow t_u \prec_{A_i} t_v \big),$$

where $u, v \in [1,k]$, each t_j is a tuple variable denoting a tuple of R, and ψ is a conjunction of predicates of the form (1) $t_j \prec_{A_l} t_h$, *i.e.,* t_h is more current than t_j in attribute A_l; (2) $t_j[A_l]=t_h[A_l]$ (resp. $t_j[A_l]\neq t_h[A_l]$), *i.e.,* $t_j[A_l]$ and $t_h[A_l]$ are identical (resp. distinct) values; (3) $t_j[A_l]=c$ (resp. $t_j[A_l]\neq c$), where c is a constant; and (4) possibly other built-in predicates defined on particular domains.

The constraint is interpreted over completions D_t^c of temporal instances of R. We say that D_t^c *satisfies* φ, denoted by $D_t^c \models \varphi$, if for all tuples $t_1, \ldots, t_k$ in D that have the same EID value, if these tuples satisfy the predicates in ψ following the standard semantics of FO, then $t_u \prec_{A_i} t_v$. The use of EID in φ enforces that φ is imposed on tuples that refer to *the same entity*. We say that D_t^c satisfies a set Σ of denial constraints, denoted by $D_t^c \models \Sigma$, if $D_t^c \models \varphi$ for all $\varphi \in \Sigma$.

Example 2.1: Recall relations Emp and Dept given in Fig. 1. Denial constraints on these relations include:

φ_1: $\forall s,t : \mathsf{Emp}\big((s[\mathsf{EID}]=t[\mathsf{EID}] \wedge s[\mathsf{salary}] > t[\mathsf{salary}]) \rightarrow t \prec_{\mathsf{salary}} s\big)$
φ_2: $\forall s,t : \mathsf{Emp}\big((s[\mathsf{EID}]=t[\mathsf{EID}] \wedge s[\mathsf{status}]=\text{“married”} \wedge t[\mathsf{status}]=\text{“single”}) \rightarrow t \prec_{\mathsf{LN}} s\big)$
φ_3: $\forall s,t : \mathsf{Emp}\big((s[\mathsf{EID}]=t[\mathsf{EID}] \wedge t \prec_{\mathsf{salary}} s) \rightarrow t \prec_{\mathsf{address}} s\big)$
φ_4: $\forall s,t : \mathsf{Dept}\big((s[\mathsf{EID}]=t[\mathsf{EID}] \wedge t \prec_{\mathsf{mgrAddr}} s) \rightarrow t \prec_{\mathsf{budget}} s\big)$

Here φ_1 states that when Emp tuples s and t refer to the same employee, if $s[\mathsf{salary}] > t[\mathsf{salary}]$, then s is more current than t in attribute salary. Note that '$<$' denotes the built-in predicate "less-than" in the numeric domain of salary, whereas $\prec_{\mathsf{salary}}$ is the currency order for salary. Constraint φ_2 asserts that if $s[\mathsf{status}]$ is married and $t[\mathsf{status}]$ is single, then s is more current than t in LN. Constraint φ_3 states that if s is more current than t in salary, then s is also more current than t in address; similarly for φ_4. □

Copy functions. Consider two temporal instances $D_{(t,1)}=(D_1, \prec_{A_1}, \ldots, \prec_{A_p})$ and $D_{(t,2)}=(D_2, \prec_{B_1}, \ldots, \prec_{B_q})$ of (possibly distinct) relation schemas R_1 and R_2, respectively. A *copy function* ρ of signature $R_1[\vec{A}] \Leftarrow R_2[\vec{B}]$ is a partial mapping from D_1 to D_2, where $\vec{A}=(A_1, \ldots, A_l)$ and $\vec{B}=(B_1, \ldots, B_l)$, denoting attributes in R_1 and R_2, respectively. Here ρ is required to satisfy *the copying condition*: for each tuple t in D_1, if $\rho(t)=s$, then $t[A_i]=s[B_i]$ for all $i\in[1,l]$.

Intuitively, for tuples $t\in D_1$ and $s\in D_2$, $\rho(t)=s$ indicates that the values of the $\vec{A}$ attributes of t have been imported from the $\vec{B}$ attributes of tuple s in D_2. Here $\vec{A}$ specifies a list of *correlated attributes* that should be copied together.

The copy function ρ is called $\prec$*-compatible* (*w.r.t.* the currency orders found in $D_{(t,1)}$ and $D_{(t,2)}$) if for all $t_1, t_2 \in D_1$, for each $i\in[1,l]$, if $\rho(t_1)=s_1$, $\rho(t_2)=s_2$, $t_1[\mathsf{EID}]=t_2[\mathsf{EID}]$ and $s_1[\mathsf{EID}]=s_2[\mathsf{EID}]$, then $s_1 \prec_{B_i} s_2$ implies $t_1 \prec_{A_i} t_2$.

Intuitively, $\prec$*-compatibility* requires that copy functions preserve currency orders. In other words, when attribute values are imported from D_2 to D_1 the currency orders on corresponding tuples defined in $D_{(t,2)}$ are inherited by $D_{(t,1)}$.

Example 2.2: Consider relations Emp and Dept shown in Fig. 1. A copy function ρ of signature $\mathsf{Dept}[\mathsf{mgrAddr}] \Leftarrow \mathsf{Emp}[\mathsf{address}]$, depicted in Fig. 1 by arrows, is given as follows: $\rho(t_1)=s_1$, $\rho(t_2)=s_1$, $\rho(t_3)=s_3$ and $\rho(t_4)=s_4$. That is, the mgrAddr values of t_1 and t_2 have both been imported from $s_1[\mathsf{address}]$, while $t_3[\mathsf{mgrAddr}]$ and $t_4[\mathsf{mgrAddr}]$ are copied from $s_3[\mathsf{address}]$ and $s_4[\mathsf{address}]$, respectively. The function satisfies the copying condition, since $t_1[\mathsf{mgrAddr}]=t_2[\mathsf{mgrAddr}]=s_1[\mathsf{address}]$, $t_3[\mathsf{mgrAddr}]=s_3[\mathsf{address}]$, and $t_4[\mathsf{mgrAddr}]=s_4[\mathsf{address}]$.

Suppose that $\prec_A$ is empty for each attribute A in Emp or Dept. Then copy function ρ is $\prec$-compatible *w.r.t.* these temporal instances of Emp and Dept. In contrast, as-

sume that partial currency orders $s_1 \prec_{\mathsf{address}} s_3$ on Emp and $t_3 \prec_{\mathsf{mgrAddr}} t_1$ are given. Then ρ is not $\prec$-compatible. Indeed, since s_1, s_3 pertain to the same person Mary, and t_1, t_3 to the same department R&D, the relation $s_1 \prec_{\mathsf{address}} s_3$ should carry over into $t_1 \prec_{\mathsf{mgrAddr}} t_3$, as $\rho(t_1) = s_1$ and $\rho(t_3) = s_3$. Clearly, $t_3 \prec_{\mathsf{mgrAddr}} t_1$ and $t_1 \prec_{\mathsf{mgrAddr}} t_3$ are contradictory. □

Consistent completions of temporal orders. A *specification* $\mathbf{S}$ of data currency consists of (1) a collection of temporal instances $D_{(t,i)}$ of schema R_i for $i \in [1,s]$, (2) a set Σ_i of denial constraints imposed on each $D_{(t,i)}$, and (3) a (possibly empty) copy function $\rho_{(i,j)}$ that imports data from $D_{(t,i)}$ to $D_{(t,j)}$ for $i,j \in [1,s]$. It specifies data values and entities (by normal instances embedded in $D_{(t,i)}$), partial currency orders known for each relation (by $D_{(t,i)}$), additional currency information derived from the semantics of the data (Σ_i), and data that has been copied from one source to another ($\rho_{(i,j)}$). These $D_{(t,i)}$'s may denote different data sources, *i.e.,* they may not necessarily be in the same database.

A *consistent completion* $\mathbf{D}^c$ of $\mathbf{S}$ consists of temporal instances $D^c_{(t,i)}$ of R_i such that for all $i,j \in [1,s]$,

1. $D^c_{(t,i)}$ is a completion of $D_{(t,i)}$,
2. $D^c_{(t,i)} \models \Sigma_i$, and
3. $\rho_{(i,j)}$ is compatible *w.r.t.* the completed currency orders found in $D^c_{(t,i)}$ and $D^c_{(t,j)}$.

We use $\mathsf{Mod}(\mathbf{S})$ to denote the set of all consistent completions of $\mathbf{S}$. We say that $\mathbf{S}$ is *consistent* if $\mathsf{Mod}(\mathbf{S}) \neq \emptyset$, *i.e.,* there exists at least one consistent completion of $\mathbf{S}$.

Intuitively, if $D_{(t,i)} = (D_i, \prec_{A_1}, \ldots, \prec_{A_n})$ is part of a specification and $D^c_{(t,i)} = (D_i, \prec^c_{A_1}, \ldots, \prec^c_{A_n})$ is part of a consistent completion of that specification, then each $\prec^c_{A_j}$ extends $\prec_{A_j}$ to a completed currency order, and the completed orders satisfy the denial constraints Σ_i and the constraints imposed by copy functions. Observe that the copying condition and $\prec$-compatibility impose constraints on consistent completions. This is particularly evident when a data source imports data from multiple sources, and when two data sources copy from each other, directly or indirectly. In addition, these constraints interact with denial constraints.

Example 2.3: Consider a specification $\mathbf{S}_0$ consisting of Emp and Dept of Fig. 1, the denial constraints φ_1–φ_4 given in Example 2.1, and the copy ρ defined in Example 2.2. Assume that no currency orders are known for Emp and Dept initially. A consistent completion $\mathbf{D}^c_0$ of $\mathbf{S}_0$ defines (1) $s_1 \prec_A s_2 \prec_A s_3$ when A ranges over FN, LN, address, salary and status for Emp tuples, and (2) $t_1 \prec_B t_2 \prec_B t_4 \prec_B t_3$ when B ranges over mgrFN, mgrLN, mgrAddr and budget for Dept tuples (here we assume that dname is the EID attribute of Dept). One can verify that $\mathbf{D}^c_0$ satisfies the denial constraints and the constraints imposed by ρ, and hence, $\mathbf{D}^c_0 \in \mathsf{Mod}(\mathbf{S}_0)$. Note that no currency order is defined between any of s_1, s_2, s_3 and any of s_4, s_5, since they represent different entities.

Suppose that Dept also copies from a source D_1 consisting of a single tuple s'_1, which is the same as s_1 except that $s'_1[\mathsf{address}] =$ "5 Elm Ave". It uses a copy function ρ_1 that imports $s'_1[\mathsf{address}]$ to $t_1[\mathsf{mrgAddr}]$. Then there exists no consistent completion in this setting since t_1 may not import distinct values $s'_1[\mathsf{address}]$ and $s_1[\mathsf{address}]$ for $t_1[\mathsf{mrgAddr}]$. In other words, the constraints imposed by the copying conditions of ρ and ρ_1 cannot be satisfied at the same time.

D	a normal instance of a relation schema R
D_t	a temporal instance of R with partial currency orders
D^c_t	a completion of partial currency orders in D_t
$\mathbf{S}$	a specification of data currency
$\mathbf{D}^c$	a consistent completion of a specification $\mathbf{S}$
$\mathsf{LST}(\mathbf{D}^c)$	the current instance of $\mathbf{D}^c$
$\bar{\rho}$	a collection of copy functions in $\mathbf{S}$
$\bar{\rho}^e$	an extension of copy functions $\bar{\rho}$
$\mathbf{S}^e$	an extension of specification $\mathbf{S}$ by $\bar{\rho}^e$
$\mathbf{D}^e$	an extension of temporal instances by $\bar{\rho}^e$

Table 1: A summary of notations

As another example, suppose that there is a copy function ρ_2 that imports budget attribute values of t_1 and t_3 from the budget attributes of s''_1 and s''_3 in another source D_2, respectively, where $s''_1 = t_1$ and $s''_3 = t_3$, but in D_2, $s''_3 \prec_{\mathsf{budget}} s''_1$. Then there is no consistent completion in this setting either. Indeed, all completed currency orders of $\prec_{\mathsf{budget}}$ in Dept have to satisfy denial constraints φ_1, φ_3 and φ_4, which enforce $t_1 \prec_{\mathsf{budget}} t_3$, but ρ_2 is not $\prec$-compatible with this currency order. This shows the interaction between denial constraints and currency constraints of copy functions. □

Current instances. In a temporal instance $D_t = (D, \prec_{A_1}, \ldots, \prec_{A_n})$ of R, let $E = \{t[\mathsf{EID}] \mid t \in D\}$, and for each entity $e \in E$, let $I_e = \{t \in D \mid t[\mathsf{EID}] = e\}$. That is, E contains all EID values in D, and I_e is the set of tuples pertaining to the entity whose EID is e.

In a completion D^c_t of D_t, for each attribute A of R, the *current A value* for entity $e \in E$ is the value $t[A]$, where t is the greatest (*i.e.,* most current) tuple in the totally ordered set $(I_e, \prec^c_A)$. The *current tuple* for entity $e \in E$, denoted by $\mathsf{LST}(e, D^c_t)$, is the tuple t_e such that for each attribute A of R, $t_e[A]$ is the current A value for entity e.

We use $\mathsf{LST}(D^c_t)$ to denote $\{\mathsf{LST}(e, D^c_t) \mid e \in E\}$, referred to as the *current instance* of D^c_t. Observe that $\mathsf{LST}(D^c_t)$ is a *normal instance* of R, carrying no currency orders. For any $\mathbf{D}^c \in \mathsf{Mod}(\mathbf{S})$, we define $\mathsf{LST}(\mathbf{D}^c) = \{\mathsf{LST}(D^c_{(t,i)}) \mid D^c_{(t,i)} \in \mathbf{D}^c\}$, the set of all current instances.

Example 2.4: Recall the completion $\mathbf{D}^c_0$ of $\mathbf{S}_0$ from Example 2.3. Then $\mathsf{LST}(\mathbf{D}^c_0) = \{\mathsf{LST}(\mathsf{Emp}), \mathsf{LST}(\mathsf{Dept})\}$, where $\mathsf{LST}(\mathsf{Emp}) = \{s_3, s_4, s_5\}$, and $\mathsf{LST}(\mathsf{Dept}) = \{t_3\}$. Note that $\mathsf{LST}(\mathsf{Emp})$ and $\mathsf{LST}(\mathsf{Dept})$ are normal instances.

As another example, suppose that s_4 and s_5 refer to the same person. Consider an extension of the currency orders given in $\mathbf{D}^c_0$ by adding $s_4 \prec_A s_5$ and $s_5 \prec_B s_4$, where A ranges over FN, LN, address and status while B is salary. Then the current tuple of this person is (Robert, Luth, 8 Drum St, 80k, married), in which the first four attributes are taken from s_5 while its salary attribute is taken from s_4. □

Evaluating queries with current values. Consider a query Q posed on normal instances of $(R_1, \ldots, R_l)$, which does not refer to currency orders, where R_i is in specification $\mathbf{S}$ for $i \in [1,l]$. We say that a tuple t is a *certain current answer* to Q *w.r.t.* $\mathbf{S}$ if t is in

$$\bigcap_{\mathbf{D}^c \in \mathsf{Mod}(\mathbf{S})} Q\big(\mathsf{LST}(\mathbf{D}^c)\big).$$

That is, t is warranted to be an answer computed from the current values no matter how the partial currency orders in $\mathbf{S}$ are completed, as long as the denial constraints and constraints imposed by the copy functions of $\mathbf{S}$ are satisfied.

Example 2.5: Recall queries Q_1, Q_2, Q_3 and Q_4 from Example 1.1, and specification $\mathbf{S}_0$ from Example 2.3. One can

verify that answers to the queries given in Example 1.1 are certain current answers *w.r.t.* $\mathbf{S}_0$, *i.e.,* the answers remain unchanged in $\mathsf{LST}(\mathbf{D}^c)$ for all $\mathbf{D}^c \in \mathsf{Mod}(\mathbf{S}_0)$. □

We summarize notations in Table 2, including those given in this section and notations to be introduced in Section 5.

3. Decision Problems for Data Currency

We study four problems associated with data currency.

The consistency of specifications. The first problem is to decide whether a given specification $\mathbf{S}$ makes sense, *i.e.,* whether there exists any consistent completion of $\mathbf{S}$. As shown in Example 2.3, there exist specifications $\mathbf{S}$ such that $\mathsf{Mod}(\mathbf{S})$ is empty, because of the interaction between denial constraints and copy functions, among other things.

CPS:	The *consistency problem for specifications.*
INPUT:	A specification $\mathbf{S}$ of data currency.
QUESTION:	Is $\mathsf{Mod}(\mathbf{S})$ nonempty?

Certain currency orders. The next question studies whether a given currency order is contained in all consistent completions of a specification. Given two temporal instances $D_{(t,1)} = (D, \prec_{A_1}, \ldots, \prec_{A_n})$ and $D_{(t,2)} = (D, \prec'_{A_1}, \ldots, \prec'_{A_n})$ of the same schema R, we say that $D_{(t,1)}$ is *contained in* $D_{(t,2)}$, denoted by $D_{(t,1)} \subseteq D_{(t,2)}$, if $\prec_{A_j} \subseteq \prec'_{A_j}$ for all $j \in [1,n]$.

Consider a specification $\mathbf{S}$ in which there is a temporal instance $D_t = (D, \prec_{A_1}, \ldots, \prec_{A_n})$ of schema R. A *currency order* for D_t is a temporal instance $O_t = (D, \prec'_{A_1}, \ldots, \prec'_{A_n})$ of R. Observe that O_t does not necessarily contain D_t.

COP:	The *certain ordering problem.*
INPUT:	A specification $\mathbf{S}$ in which D_t is a temporal instance, and a currency order O_t for D_t.
QUESTION:	Is for all $\mathbf{D}^c \in \mathsf{Mod}(\mathbf{S})$, $O_t \subseteq D_t^c$? Here D_t^c is the completion of D_t in $\mathbf{D}^c$.

Example 3.1: Consider specification $\mathbf{S}_0$ of Example 2.3. We want to know whether $s_1 \prec_{\mathsf{salary}} s_3$ is assured by every completion $\mathbf{D}^c \in \mathsf{Mod}(\mathbf{S}_0)$. To this end we construct a currency order $O_t = (\mathsf{Emp}, \prec_{\mathsf{FN}}, \prec_{\mathsf{LN}}, \prec_{\mathsf{address}}, \prec_{\mathsf{salary}}, \prec_{\mathsf{status}})$, in which $s_1 \prec_{\mathsf{salary}} s_3$ is in $\prec_{\mathsf{salary}}$, but the partial orders for all other attributes are empty. One can verify that O_t is indeed a certain currency order, as assured by denial constraint φ_1.

Similarly, one can define a currency order O'_t to check whether $t_3 \prec_{\mathsf{mgrFN}} t_4$ is entailed by all $\mathbf{D}^c \in \mathsf{Mod}(\mathbf{S}_0)$. One can readily verify that it is not the case. Indeed, there exists $\mathbf{D}_1^c \in \mathsf{Mod}(\mathbf{S}_0)$, such that $t_4 \prec_{\mathsf{mgrFN}} t_3$ is given in $\mathbf{D}_1^c$. □

Certain current instances. Given a specification $\mathbf{S}$ of data currency, one naturally wants to know whether every consistent completion of $\mathbf{S}$ yields the same current instances. We say that a specification $\mathbf{S}$ of data currency is *deterministic for current instances* if for all consistent completions $\mathbf{D}_1^c, \mathbf{D}_2^c \in \mathsf{Mod}(\mathbf{S})$, $\mathsf{LST}(\mathbf{D}_1^c) = \mathsf{LST}(\mathbf{D}_2^c)$. This definition naturally carries over to a particular relation schema R: specification $\mathbf{S}$ is said to be *deterministic for current R instances* if for all consistent completions $\mathbf{D}_1^c, \mathbf{D}_2^c \in \mathsf{Mod}(\mathbf{S})$, the instance of R in $\mathsf{LST}(\mathbf{D}_1^c)$ is equal to the instance of R in $\mathsf{LST}(\mathbf{D}_2^c)$.

DCIP:	The *deterministic current instance problem*
INPUT:	A specification $\mathbf{S}$.
QUESTION:	Is $\mathbf{S}$ deterministic for current instances?

Example 3.2: The specification $\mathbf{S}_0$ of Example 2.3 is deterministic for current Emp instances. Indeed, for all $\mathbf{D}^c \in \mathsf{Mod}(\mathbf{S}_0)$, if D_{Emp}^c is the completion of the Emp instance in $\mathbf{D}^c$, then $\mathsf{LST}(D_{\mathsf{Emp}}^c) = \{s_3, s_4, s_5\}$. □

Query answering. Given a query Q, we want to know whether a tuple t is in $Q\big(\mathsf{LST}(\mathbf{D}^c)\big)$ for all $\mathbf{D}^c \in \mathsf{Mod}(\mathbf{S})$.

CCQA($\mathcal{L}_Q$):	The *certain current query answering* problem.
INPUT:	A specification $\mathbf{S}$, a tuple t and a query $Q \in \mathcal{L}_Q$.
QUESTION:	Is t a certain current answer to Q *w.r.t.* $\mathbf{S}$?

We study CCQA($\mathcal{L}_Q$) when $\mathcal{L}_Q$ ranges over the following query languages (see, *e.g.,* [1] for the details):

- CQ, the class of conjunctive queries built up from relation atoms and equality ($=$), by closing under conjunction $\wedge$ and existential quantification $\exists$;
- UCQ, unions of conjunctive queries of the form $Q_1 \cup \cdots \cup Q_k$, where for each $i \in [1,k]$, Q_i is in CQ;
- $\exists$FO$^+$, first-order logic (FO) queries built from atomic formulas, by closing under $\wedge$, *disjunction* $\vee$ and $\exists$; and
- FO queries built from atomic formulas using $\wedge$, $\vee$, *negation* $\neg$, $\exists$ and universal quantification $\forall$.

While different query languages have no impact on the data complexity of CCQA($\mathcal{L}_Q$), as will be seen soon, they do make a difference when the combined complexity is concerned.

4. Reasoning about the Currency of Data

In this section we focus on CPS, COP, DCIP and CCQA. We establish the data complexity and combined complexity of these problems. For the data complexity, we fix denial constraints and queries (for CCQA), and study the complexity in terms of varying size of data sources and copy functions. For the combined complexity we also allow denial constraints and queries to vary (see, *e.g.,* [1] for a detailed discussion of data and combined complexity).

The consistency of specifications. We start with CPS, which is to decide, given a specification $\mathbf{S}$ consisting of partial currency orders, denial constraints and copy functions, whether there exists any consistent completion in $\mathsf{Mod}(\mathbf{S})$.

The result below tells us the following. (1) The problem is nontrivial: it is Σ_2^p-complete. It remains intractable when denial constraints are fixed (data complexity). (2) Denial constraints are a major factor that makes the problem hard. Indeed, the complexity bounds are not affected even when no copy functions are defined in $\mathbf{S}$.

Theorem 4.1: For CPS, (1) the combined complexity is Σ_2^p-complete, and (2) the data complexity is NP-complete. The upper bounds and lower bounds remain unchanged even in the absence of copy functions. □

Proof sketch: (1) *Lower bounds.* For the combined complexity, we show that CPS is Σ_2^p-hard by reduction from the $\exists^*\forall^* 3$SAT problem, which is Σ_2^p-complete (cf. [28]). Given a sentence $\phi = \exists X \forall Y \psi(X,Y)$, we construct a specification $\mathbf{S}$ consisting of a single temporal instance D_t of a binary relation schema and a set Γ of denial constraints, such that ϕ is true iff $\mathsf{Mod}(\mathbf{S}) \neq \emptyset$. We use D_t to encode truth assignments μ_X for X, and Γ to assure that μ_X satisfies $\forall Y \psi(X,Y)$ if there exists a consistent completion of D_t. Here $\forall Y \psi(X,Y)$ is encoded by leveraging the property $\forall Y(\bigwedge_{i\in[1,r]} C_i(X,Y)) = \bigwedge_{i\in[1,r]} \forall Y C_i(X,Y)$, for $\psi(X,Y) = \bigwedge_{i\in[1,r]} C_i(X,Y)$.

For the data complexity, we show that CPS is NP-hard by reduction from the Betweenness problem, which is NP-complete (cf. [18]). Given two sets E and $F = \{(e_i, e_j, e_k) \mid e_i, e_j, e_k \in E\}$, the Betweenness problem is to decide whether there is a bijection $\pi : E \to \{1, \ldots, |E|\}$ such that for each $(e_i, e_j, e_k) \in E$, either $\pi(e_i) < \pi(e_j) < \pi(e_k)$ or $\pi(e_k) < \pi(e_j) < \pi(e_i)$. Given E and F, we define a specification $\mathbf{S}$ with a temporal instance D_t of a 4-ary schema, and a set of *fixed* denial constraints. We show that there exists a solution to the Betweenness problem iff $\mathsf{Mod}(\mathbf{S})$ is nonempty.

(2) *Upper bounds.* We provide an algorithm that, given a specification $\mathbf{S}$, guesses a completion $\mathbf{D}^c$ of total orders in $\mathbf{S}$, and then checks whether $\mathbf{D}^c \in \mathsf{Mod}(\mathbf{S})$. The checking involves (a) denial constraints and (b) the copying condition and $\prec$-compatibility of copy functions in $\mathbf{S}$. Step (b) is in PTIME. Step (a) is in PTIME if the denial constraints are fixed, and it uses an NP-oracle otherwise. Hence CPS is in NP for data complexity and in Σ_2^p for combined complexity.

In the proofs for the lower bounds, no copy functions are defined, and the relation schemas are fixed. □

The certainty of currency orders. We next study COP and DCIP. The certain currency ordering problem COP is to determine, given a specification $\mathbf{S}$ and a currency order O_t, whether each $t \prec_A s$ in O_t is entailed by the partial currency orders, denial constraints and copy functions in $\mathbf{S}$. The deterministic current instance problem DCIP is to decide, given $\mathbf{S}$, whether the current instance of each temporal instance of $\mathbf{S}$ is unchanged for all consistent completions of $\mathbf{S}$. These problems are, unfortunately, also beyond reach in practice.

Corollary 4.2: For both COP and DCIP, (1) the combined complexity is Π_2^p-complete, and (2) the data complexity is coNP-complete. The complexity bounds remain unchanged when no copy functions are present. □

Proof sketch: (1) *Lower bounds.* For both COP and DCIP, the lower bounds are verified by reduction from the complement of CPS, for data complexity and combined complexity.

(2) *Upper bounds.* A non-deterministic algorithm is developed for each of COP and DCIP, which is in coNP (data complexity) and Π_2^p (combined complexity). □

Query answering. The certain currency query answering problem $\mathsf{CCQA}(\mathcal{L}_Q)$ is to determine, given a tuple t, a specification $\mathbf{S}$ and a query $Q \in \mathcal{L}_Q$, whether $t \in Q\big(\mathsf{LST}(\mathbf{D}^c)\big)$ for all $\mathbf{D}^c \in \mathsf{Mod}(\mathbf{S})$. The result below provides the data complexity of the problem, as well as its combined complexity when $\mathcal{L}_Q$ ranges over CQ, UCQ, $\exists$FO$^+$and FO. It tells us the following. (1) Disjunctions in UCQ and $\exists$FO$^+$do not incur extra complexity to CCQA. Indeed, CCQA has the same complexity for CQ as for UCQ and $\exists$FO$^+$. (2) In contrast, the presence of negation in FO complicates the analysis. (3) Copy functions have no impact on the complexity bounds.

Theorem 4.3: The combined complexity of $\mathsf{CCQA}(\mathcal{L}_Q)$ is

- Π_2^p-complete when $\mathcal{L}_Q$ is CQ, UCQ or $\exists$FO$^+$, and
- PSPACE-complete when $\mathcal{L}_Q$ is FO.

The data complexity is coNP-complete when $\mathcal{L}_Q \in \{$CQ, UCQ,$\exists$FO$^+$, FO$\}$. These complexity bounds are unchanged in the absence of copy functions. □

Proof sketch: *Lower bounds.* For the combined complexity, we show the following: (a) CCQA is already Π_2^p-hard for CQ, and (b) it is PSPACE-hard for FO.

(a) For CQ, we verify it by reduction from the $\forall^*\exists^*$3SAT problem, which is Π_2^p-complete (cf. [28]). Given a sentence $\phi = \forall X \exists Y \psi(X, Y)$, we define a CQ query Q, a fixed tuple t, and a specification $\mathbf{S}$ consisting of five temporal instances of fixed schemas. We use these temporal instances to encode (i) disjunction and negation, which are not expressible in CQ, (ii) truth assignments μ_X for X, with an instance D_X, and (iii) relations for inspecting whether t is an answer to Q. Query Q encodes $\exists Y \psi(X, Y)$ *w.r.t.* μ_X, such that ϕ is true iff t is an answer to Q for each consistent completion of D_X, *i.e.,* when μ_X ranges over all truth assignments for X.

(b) For FO, we show that CCQA is PSPACE-hard by reduction from Q3SAT, which is PSPACE-complete (cf. [28]). Given an instance ϕ of Q3SAT, we define an FO query Q, a fixed tuple t and a specification $\mathbf{S}$ with a single temporal instance. Query Q encodes ϕ, and the relation encodes Boolean values for which there is a single completion $\mathbf{D}_0^c$ in $\mathsf{Mod}(\mathbf{S})$. We show that ϕ is true iff t is in $Q(\mathbf{D}_0^c)$.

For the data complexity, we show that CCQA is coNP-hard even for CQ, by reduction from the complement of 3SAT. Given a propositional formula ψ, we define a *fixed* CQ query Q, a fixed tuple t and a specification $\mathbf{S}$ consisting of two temporal instances D_ψ and $D_{\neg\psi}$ of fixed relation schemas. We use D_ψ to encode (i) truth assignments μ_X for variables X in ψ, and (ii) literals in ψ. We encode the negations of clauses in ψ using $D_{\neg\psi}$, for which there is a unique consistent completion. For each consistent completion of D_ψ, *i.e.,* each μ_X for X, query Q returns t iff ψ is not satisfied by μ_X.

In the lower bound proofs, neither denial constraints nor copy functions are defined, and all the schemas are fixed.

Upper bounds. We develop an algorithm that, given a query Q, a tuple t and a specification $\mathbf{S}$, returns "no" if there exists $\mathbf{D}^c \in \mathsf{Mod}(\mathbf{S})$ such that $t \notin Q\big(\mathsf{LST}(\mathbf{D}^c)\big)$. The algorithm first guesses $\mathbf{D}^c$, and then checks whether (a) $\mathbf{D}^c \in \mathsf{Mod}(\mathbf{S})$ and (b) $t \notin Q\big(\mathsf{LST}(\mathbf{D}^c)\big)$. Step (b) is in coNP when Q is in $\exists$FO$^+$, and is in PSPACE if Q is in FO. When Q is fixed, step (b) is in PTIME no matter whether Q is in CQ or FO. Putting these together with Theorem 4.1 (for step (a)), we conclude that the data complexity of CCQA is in coNP, and its combined complexity is in Π_2^p for $\exists$FO$^+$and in PSPACE for FO. □

Special cases. The results above tell us that it is nontrivial to reason about data currency. In light of this, we look into special cases of these problems with lower complexity. As shown by Theorem 4.1 and Corollary 4.2, denial constraints make the analyses of CPS, COP and DCIP intricate. Indeed, these problems are intractable even when denial constraints are fixed. Hence we consider specifications with no denial constraints, but containing partial currency orders and copy functions. The result below shows that the absence of denial constraints indeed simplifies the analyses.

Theorem 4.4: In the absence of denial constraints, CPS, COP and DCIP are in PTIME. □

Proof sketch: For CPS, we develop an algorithm that, given a specification $\mathbf{S}$ with no denial constraints defined, checks whether $\mathsf{Mod}(\mathbf{S}) \neq \emptyset$. Let $\bar{\rho}$ denote the collection of copy functions in $\mathbf{S}$. One can verify that in the absence of denial constraints, $\mathbf{S}$ is consistent iff there exists no violation of the copying condition or $\prec$-compatibility of $\bar{\rho}$ in any temporal instance of $\mathbf{S}$. Hence it suffices to detect violations in the instances of $\mathbf{S}$, rather than in their completions.

As shown in Example 2.2, it is not straightforward to check $\prec$-compatibility, especially when tuples are imported indirectly from other sources. Nonetheless, we show that this can be done in $O(|\mathbf{S}|^2)$ time, where $|\mathbf{S}|$ is the size of $\mathbf{S}$.

For COP, we show that given a specification $\mathbf{S}$ without denial constraints, to decide whether a currency order is contained in all completions of $\mathbf{S}$, it suffices to check temporal instances of $\mathbf{S}$. This can be decided by a variation of the algorithm for CPS, also in PTIME; similarly for DCIP. □

In contrast, the absence of denial constraints does not make our lives easier when it comes to CCQA. Indeed, in the proof of Theorem 4.3, the lower bounds of CCQA are verified using neither denial constraints nor copy functions.

Corollary 4.5: In the absence of denial constraints, CCQA($\mathcal{L}_Q$) remains coNP-hard (data complexity) and

- Π_2^p-hard (combined complexity) even for CQ, and
- PSPACE-hard (combined complexity) for FO. □

Theorem 4.3 tells us that the complexity of CCQA for CQ is rather robust: adding disjunctions does not increase the complexity. We next investigate the impact of removing Cartesian product from CQ on the complexity of CCQA. We consider SP queries, which are CQ queries of the form

$$Q(\vec{x}) = \exists e\, \vec{y}\ \big(R(e, \vec{x}, \vec{y}) \wedge \psi\big),$$

where ψ is a conjunction of equality atoms and $\vec{x}$ and $\vec{y}$ are disjoint sequences of variables in which no variable appears twice. SP queries support projection and selection only. For instance, $Q_1 - Q_4$ of Example 1.1 are SP queries. SP queries in which ψ is a tautology are referred to as *identity queries*.

We show that for SP queries, denial constraints make a difference. Without denial constraints, CCQA is in PTIME for SP queries. In contrast, when denial constraints are imposed, CCQA is no easier for identity queries than for $\exists$FO$^+$.

Corollary 4.6: For SP queries, CCQA(SP) is

- in PTIME in the absence of denial constraints, and
- coNP-complete (data complexity) and Π_2^p-complete (combined complexity) in the presence of denial constraints, even for identity queries. □

Proof sketch: (1) In the absence of denial constraints, one can verify that for any specification $\mathbf{S}$ and each SP query Q, $\bigcap_{\mathbf{D}^c \in \mathsf{Mod}(\mathbf{S})} Q\big(\mathsf{LST}(\mathbf{D}^c)\big)$ can be obtained from (1) evaluating Q on a representation $\mathsf{repr}(\mathbf{S})$ of $\big\{\mathsf{LST}(\mathbf{D}^c) \,|\, \mathbf{D}^c \in \mathsf{Mod}(\mathbf{S})\big\}$, where $\mathsf{repr}(\mathbf{S})$ compactly represents all possible latest values by special symbols; and (2) by removing all tuples in $Q\big(\mathsf{repr}(\mathbf{S})\big)$ that contain such special symbols. Capitalizing on this property, we develop an algorithm that takes as input an SP query Q, a tuple t and a specification $\mathbf{S}$ without denial constraints. It checks whether t is a certain current answer to Q *w.r.t.* $\mathbf{S}$ as follows: (a) check whether $\mathbf{S}$ is consistent, and return "no" if not; (b) compute the representation $\mathsf{repr}(\mathbf{S})$; (c) check whether t is in the "stripped" version of $Q\big(\mathsf{repr}(\mathbf{S})\big)$; and (d) return "yes" if so, and "no" otherwise. Step (c) is PTIME for SP queries. By Theorem 4.4 and by leveraging the certain order computed by the algorithm for CPS, steps (a) and (b) are also in PTIME. Therefore, CCQA is in PTIME in this setting, for both combined complexity (when queries are not fixed) and data complexity.

(2) In the presence of denial constraints, we show that CCQA is coNP-hard (data complexity) and Π_2^p-hard (combined complexity) by reduction from the complement of CPS, for which the complexity is established in Theorem 4.1. Given a specification $\mathbf{S}$, we construct a specification $\mathbf{S}'$, a tuple t and an identity query Q, such that $\mathsf{Mod}(\mathbf{S})$ is *empty* iff t is in $Q\big(\mathsf{LST}(\mathbf{D}^c)\big)$ for each $\mathbf{D}^c \in \mathsf{Mod}(\mathbf{S}')$. The upper bounds of CCQA is this setting follow from Theorem 4.3. □

5. Currency Preservation in Data Copying

As we have seen earlier, copy functions tell us what data values in a relation have been imported from other data sources. Naturally we want to leverage the imported values to improve query answers. This gives rise to the following questions: do the copy functions import sufficient current data values for answering a query Q? If not, how do we extend the copy functions such that Q can be answered with more up-to-date data values? To answer these questions we introduce a notion of currency-preserving copy functions.

We consider a specification $\mathbf{S}$ of data currency consisting of two collections of temporal instances (data sources) $\mathbf{D} = (D_1, \ldots, D_p)$ and $\mathbf{D}' = (D'_1, \ldots, D'_q)$, with (1) a set Σ_i (resp. Σ'_j) of denial constraints on D_i for each $i \in [1,p]$ (resp. D'_j for $j \in [1,q]$), and (2) a collection $\overline{\rho}$ of copy functions $\rho_{(j,i)}$ that imports tuples from D'_j to D_i, for $i \in [1,p]$ and $j \in [1,q]$, *i.e.,* all the functions of $\overline{\rho}$ import data from $\mathbf{D}'$ to $\mathbf{D}$.

Extensions. To formalize currency preservation, we first present the following notions. Assume that $D_i = (D, \prec_{A_1}, \ldots, \prec_{A_n})$ and $D'_j = (D', \prec_{B_1}, \ldots, \prec_{B_m})$ are temporal instances of relation schemas $R_i = (\mathsf{EID}, A_1, \ldots, A_n)$ and $R'_j = (\mathsf{EID}, B_1, \ldots, B_m)$, respectively. Assume that $n \leq m$.

An *extension* of D_i is a temporal instance $D_i^e = (D^e, \prec^e_{A_1}, \ldots, \prec^e_{A_n})$ of R_i such that (1) $D \subseteq D^e$, (2) $\prec_{A_h} \subseteq \prec^e_{A_h}$ for all $h \in [1,n]$, and (3) $\pi_{\mathsf{EID}}(D^e) = \pi_{\mathsf{EID}}(D)$. Intuitively, D_i^e extends D_i by adding new tuples for those entities that are already in D_i. It does not introduce new entities.

Consider two copy functions: $\rho_{(j,i)}$ imports tuples from D'_j to D_i, and $\rho^e_{(j,i)}$ from D'_j to D_i^e, both of signature $R_i[\vec{A}] \Leftarrow R'_j[\vec{B}]$, where $\vec{A} = (A_1, \ldots, A_n)$ and $\vec{B}$ is a sequence of n attributes in R'_j. We say that $\rho^e_{(j,i)}$ *extends* $\rho_{(j,i)}$ if

1. D_i^e is an extension of D_i;
2. for each tuple t in D_i, if $\rho_{(j,i)}(t)$ is defined, then so is $\rho^e_{(j,i)}(t)$ and moreover, $\rho^e_{(j,i)}(t) = \rho_{(j,i)}(t)$;
3. for each tuple t in $D_i^e \setminus D_i$, there exists a tuple s in D'_j such that $\rho^e_{(j,i)}(t) = s$.

We refer to D_i^e as the *extension of* D_i *by* $\rho^e_{(j,i)}$.

Observe that D_i^e is not allowed to expand arbitrarily. (a) Each new tuple t in D_i^e is copied from a tuple s in D'_j. (b) No new entity is introduced. Note that only copy functions that cover all attributes but EID of R_i can be extended. This assures that all the attributes of a new tuple are in place.

An *extension* $\overline{\rho}^e$ of $\overline{\rho}$ is a collection of copy functions $\rho^e_{(j,i)}$ such that $\overline{\rho}^e \neq \overline{\rho}$ and moreover, for all $i \in [1,p]$ and $j \in [1,q]$, either $\rho^e_{(j,i)}$ is an extension of $\rho_{(j,i)}$, or $\rho^e_{(j,i)} = \rho_{(j,i)}$.

We denote the set of all extensions of $\overline{\rho}$ as $\mathsf{Ext}(\overline{\rho})$.

For each $\overline{\rho}^e$ in $\mathsf{Ext}(\overline{\rho})$, we denote as $\mathbf{S}^e$ the *extension of* $\mathbf{S}$ *by* $\overline{\rho}^e$, which consists of the same $\mathbf{D}'$ and denial constraints as in $\mathbf{S}$, but has copy functions $\overline{\rho}^e$ and $\mathbf{D}^e = (D_1^e, \ldots, D_p^e)$, where D_i^e is an extension of D_i by $\rho^e_{(j,i)}$ for all $j \in [1,q]$.

Currency preservation. We are now ready to define currency preservation. Consider a collection $\overline{\rho}$ of copy functions

	FN	LN	address	salary	status	phone
s_1':	Mary	Dupont	6 Main St	60k	married	6671975
s_2':	Mary	Dupont	6 Main St	80k	married	6671975
s_3':	Mary	Smith	2 Small St	80k	divorced	2962845

Figure 2: Relation Mgr

in a specification $\mathbf{S}$. We say that $\overline{\rho}$ is *currency preserving* for a query Q *w.r.t.* $\mathbf{S}$ if (a) $\mathsf{Mod}(\mathbf{S}) \neq \emptyset$, and moreover, (b) for all $\overline{\rho}^e \in \mathsf{Ext}(\overline{\rho})$ such that $\mathsf{Mod}(\mathbf{S}^e) \neq \emptyset$, we have that

$$\bigcap_{\mathbf{D}^c \in \mathsf{Mod}(\mathbf{S})} Q\big(\mathsf{LST}(\mathbf{D}^c)\big) = \bigcap_{\mathbf{D}_e^c \in \mathsf{Mod}(\mathbf{S}^e)} Q\big(\mathsf{LST}(\mathbf{D}_e^c)\big).$$

Intuitively, $\overline{\rho}$ is currency preserving if (1) $\overline{\rho}$ is meaningful; and (2) for each extension $\overline{\rho}^e$ of $\overline{\rho}$ that makes sense, the certain current answers to Q are not improved by $\overline{\rho}^e$, *i.e.,* no matter what additional tuples are imported for those entities in $\mathbf{D}$, the certain current answers to Q remain unchanged.

Example 5.1: As shown in Fig. 2, relation Mgr collects manager records. Consider a specification $\mathbf{S}_1$ consisting of the following: (a) temporal instances Mgr of Fig. 2 and Emp of Fig. 1, in which partial currency orders are $\emptyset$ for all attributes; (b) denial constraints φ_1–φ_3 of Example 2.1 and

φ_5: $\forall s,t$: Mgr $\big((s[\mathsf{EID}] = t[\mathsf{EID}] \wedge s[\mathsf{status}] = \text{“divorced”} \wedge t[\mathsf{status}] = \text{“married”}) \rightarrow t \prec_{\mathsf{LN}} s\big)$,

i.e., if s[status] is divorced and t[status] is married, then s is more current than t in LN; and (c) a copy function ρ with signature $\mathsf{Emp}[\vec{A}] \Leftarrow \mathsf{Mgr}[\vec{A}]$, where $\vec{A}$ is (FN, LN, address, salary, status), such that $\rho(s_3) = s_2'$, *i.e.,* s_3 of Emp is copied from s_2' of Mgr. Obviously $\mathbf{S}_1$ is consistent.

Recall query Q_2 of Example 1.1, which is to find Mary's current last name. For Q_2, ρ is not currency preserving. Indeed, there is an extension ρ_1 of ρ by copying s_3' to Emp. In all consistent completions of the extension Emp_1 of Emp by ρ_1, the answer to Q_2 is Smith. However, the answer to Q_2 in all consistent completions of Emp is Dupont (see Examples 1.1 and 2.5). In contrast, ρ_1 is currency preserving for Q_2. Indeed, copying more tuples from Mgr (*i.e.,* tuple s_1') to Emp does not change the answer to Q_2 in Emp_1. □

Deciding currency preservation. There are several decision problems associated with currency-preserving copy functions, which we shall investigate in the rest of the paper. The first problem is to decide whether given copy functions have imported necessary current data for answering a query.

CPP($\mathcal{L}_Q$):	The *currency preservation problem.*
INPUT:	A query Q in $\mathcal{L}_Q$, and a specification $\mathbf{S}$ of data currency with copy functions $\overline{\rho}$.
QUESTION:	Is $\overline{\rho}$ currency preserving for Q?

Minimal copying. Moreover, we want to know whether a currency preserving $\overline{\rho}$ does not copy unnecessary or redundant data. To this end, we use $|\overline{\rho}|$ to denote the sum of the sizes of data values copied by $\overline{\rho}$. We say that $\overline{\rho}$ is *minimal* for Q if there exists no collection $\overline{\rho}'$ of copy functions such that (1) $\overline{\rho} \in \mathsf{Ext}(\overline{\rho}')$, (2) $|\overline{\rho}'| < |\overline{\rho}|$, and (3) $\overline{\rho}'$ is currency preserving for Q. That is, there exists no currency-preserving $\overline{\rho}'$ that imports less data from $\mathbf{D}'$ to $\mathbf{D}$ than $\overline{\rho}$.

MCP($\mathcal{L}_Q$):	The *minimal copying problem.*
INPUT:	$\mathbf{S}$ and Q as in CPP, with a collection $\overline{\rho}$ of currency-preserving copy functions in $\mathbf{S}$.
QUESTION:	Is $\overline{\rho}$ minimal for Q?

Extending copy functions. Consider a consistent specification $\mathbf{S}$ in which $\overline{\rho}$ is not currency preserving for a query Q. The next problem is to decide whether $\overline{\rho}$ in $\mathbf{S}$ can be extended to be currency preserving for Q.

ECP($\mathcal{L}_Q$):	The *existence problem.*
INPUT:	A query Q in $\mathcal{L}_Q$, and a consistent specification $\mathbf{S}$ with non-currency-preserving $\overline{\rho}$
QUESTION:	Does there exist $\overline{\rho}^e$ in $\mathsf{Ext}(\overline{\rho})$ that is currency preserving for Q?

Bounded extension. We also want to know whether it suffices to extend $\overline{\rho}$ by copying additional data of a bounded size, and make it currency preserving.

BCP($\mathcal{L}_Q$):	The *bounded copying problem.*
INPUT:	$\mathbf{S}$, $\overline{\rho}$ and Q as in ECP, and a positive number k.
QUESTION:	Does there exist $\overline{\rho}^e \in \mathsf{Ext}(\overline{\rho})$ such that $\overline{\rho}^e$ is currency preserving for Q and $\lvert\overline{\rho}^e\rvert \leq k + \lvert\overline{\rho}\rvert$?

6. Deciding Currency Preservation

We next study the decision problems in connection with currency-preserving copy functions, namely, CPP($\mathcal{L}_Q$), MCP($\mathcal{L}_Q$), ECP($\mathcal{L}_Q$) and BCP($\mathcal{L}_Q$) when $\mathcal{L}_Q$ is CQ, UCQ, $\exists$FO$^+$ or FO. We provide their combined complexity and data complexity, and identify special cases with lower complexity.

Checking currency preservation. We first investigate CPP($\mathcal{L}_Q$), the problem of deciding whether a collection of copy functions in a given specification is currency preserving for a query Q. We show that CPP is nontrivial. Indeed, its combined complexity is already Π_3^p-hard when Q is in CQ, and it is PSPACE-complete when Q is in FO.

One might be tempted to think that fixing denial constraints would make our lives easier. Indeed, in practice denial constraints are often predefined and fixed, and only data, copy functions and query vary. Moreover, as shown in Theorem 4.1 for the consistency problem, fixing denial constraints indeed helps there. Unfortunately, it does not simplify the analysis of the combined complexity when it comes to CPP. Even when both query and denial constraints are fixed, the problem is Π_2^p-complete (data complexity).

Theorem 6.1: For CPP($\mathcal{L}_Q$), the combined complexity is

- Π_3^p-complete when $\mathcal{L}_Q$ is CQ, UCQ or $\exists$FO$^+$, and
- PSPACE-complete when $\mathcal{L}_Q$ is FO.
- Its data complexity is Π_2^p-complete when $\mathcal{L}_Q \in \{$CQ, UCQ, $\exists$FO$^+$,FO$\}$.

The combined complexity bounds remain unchanged when denial constraints and copy functions are fixed. □

Proof sketch: (1) *Lower bounds.* For the combined complexity, it suffices to show that CPP is already Π_3^p-hard for CQ, while for FO, it is PSPACE-hard. For the data complexity, we show that CPP is Π_2^p-hard with CQ queries.

The Π_3^p lower bound is verified by reduction from the $\forall^*\exists^*\forall^*$3SAT problem, which is Π_3^p-complete (cf. [28]). Given a sentence $\phi = \forall X \exists Y \forall Z \psi(X, Y, Z)$, we construct a query Q in CQ, and a specification $\mathbf{S}$ with (i) two data sources $\mathbf{D}$ and $\mathbf{D}'$, (ii) a single copy function ρ and (iii) a set of denial constraints. Source $\mathbf{D}$ consists of four relations encoding Boolean values, disjunction, conjunction and negation, as well as a relation D_b for testing certain current query answers. Source $\mathbf{D}'$ has a single relation D_b' from which some tuples are copied to D_b by ρ. Query Q encodes ϕ by leverag-

ing the relations in $\mathbf{D}$ for Boolean operations, and the property of $\forall Z\psi(X,Y,Z)$ explored in the proof of Theorem 4.1. We show that ϕ is true iff ρ is currency preserving for Q.

When it comes to FO, we show that CPP is PSPACE-hard by a straightforward reduction from Q3SAT.

In these proofs, both denial constraints and copy functions are fixed, independent of $\forall^*\exists^*\forall^*$3SAT or Q3SAT instances.

For the data complexity, we verify the Π_2^p lower bound for CQ by reduction from the $\forall^*\exists^*$3SAT problem. Given a sentence $\phi = \forall X \exists Y \psi(X,Y)$, we define a query Q in CQ and a specification $\mathbf{S}$ such that ϕ is true iff ρ is currency preserving for Q *w.r.t.* $\mathbf{S}$. Here Q is fixed, *i.e.,* it is independent of ϕ.

(2) *Upper bounds.* We give an algorithm that takes a query Q and a specification $\mathbf{S}$ as input, and checks whether the copy functions $\bar{\rho}$ in $\mathbf{S}$ are currency preserving for Q. It invokes oracles to check whether $\mathbf{S}$ is consistent and whether some guessed tuples are certain current answers to Q in $\mathbf{D}^c \in \mathsf{Mod}(\mathbf{S})$ but are not answers to Q in some $\mathbf{D}_e^c \in \mathsf{Mod}(\mathbf{S}^e)$, where $\mathbf{S}^e$ is an extension of $\mathbf{S}$ by extending copy functions. The oracles are in Π_2^p or Σ_2^p when Q is in $\exists$FO$^+$, and in PSPACE when Q is in FO. When Q is fixed, the oracles are in NP or coNP. From these the upper bounds follow. □

Minimal copying. We next study $\mathsf{MCP}(\mathcal{L}_Q)$, which is to decide whether currency-preserving copy functions import any data that is unnecessary or redundant for a query Q. This problem is even harder than CPP: for CQ queries, its combined complexity is Δ_4^p-complete ($P^{\Sigma_3^p}$) even if denial constraints are fixed, and its data complexity is Δ_3^p-complete. For FO queries, it is still PSPACE-complete.

Theorem 6.2: For $\mathsf{MCP}(\mathcal{L}_Q)$, the combined complexity is

- Δ_4^p-complete when $\mathcal{L}_Q$ is CQ, UCQ or $\exists$FO$^+$, and
- PSPACE-complete when $\mathcal{L}_Q$ is FO.
- Its data complexity is Δ_3^p-complete when $\mathcal{L}_Q \in \{$CQ, UCQ, $\exists$FO$^+$,FO$\}$.

The combined complexity bounds remain unchanged when denial constraints and copy functions are fixed. □

Proof sketch: (1) *Lower bounds.* For FO, the proofs are similar to their counterparts of CPS. For CQ queries, we show that MCP is Δ_4^p-hard (combined complexity) by reduction from the MSA($\exists^*\forall^*\exists^*$3SAT) problem, which is Δ_4^p-complete [26]. The latter problem is to determine, given a satisfiable sentence $\phi = \exists X \forall Y \exists Z\, \psi(X,Y,Z)$, whether in the lexicographically maximum truth assignment μ_X for variables in X such that $\forall Y \exists Z\, \psi(\mu_X(x_1),\ldots,\mu_X(x_n),Y,Z)$ evaluates to true, the last variable $x_n \in X$ has value 1, *i.e.,* whether $\mu_X(x_n) = 1$. For the data complexity, we verify that it is Δ_3^p-hard for CQ by reduction from MSA($\exists^*\forall^*$3SAT)-problem. These reductions need to encode the lexicographical successor function for variables in X, and are more involved than those given in the proof of Theorem 6.1.

(2) *Upper bounds.* We provide an algorithm that takes as input a query Q and a specification $\mathbf{S}$ with currency preserving copy functions $\bar{\rho}$. For each tuple t copied by $\bar{\rho}$, it checks whether removing t makes the copy functions not currency preserving, by using the algorithm for CPP in the proof of Theorem 6.1 as an oracle. Hence MCP is in Δ_4^p ($P^{\Sigma_3^p}$, combined complexity) and in Δ_3^p (data complexity). □

The feasibility of currency preservation. We now consider $\mathsf{ECP}(\mathcal{L}_Q)$. It is to decide, given a query Q and a specification $\mathbf{S}$ in which copy functions $\bar{\rho}$ are not currency preserving for Q, whether we can extend $\bar{\rho}$ to preserve currency. The good news is that the answer to this question is affirmative: we can always extend $\bar{\rho}$ and make them currency preserving for Q. Hence the decision problem ECP is in $O(1)$ time, although it may take much longer time to explicitly construct a currency preserving extension of $\bar{\rho}$.

Proposition 6.3: $\mathsf{ECP}(\mathcal{L}_Q)$ is decidable in $O(1)$ time for both the combined complexity and data complexity, when $\mathcal{L}_Q$ is CQ, UCQ, $\exists$FO$^+$or FO. □

Proof sketch: To show the existence of currency preserving extensions of ρ, we give an algorithm to construct one. Let $\mathbf{D}'$ be the data source in $\mathbf{S}$ from which tuples can be copied. For all copy function ρ in $\bar{\rho}$, if ρ can be extended, the algorithm expands ρ by copying tuples one by one from $\mathbf{D}'$, starting from the most current ones, as long as the extended instances satisfy the denial constraints in $\mathbf{S}$ and the constraints of the copy functions. It proceeds until no more tuples from $\mathbf{D}'$ can be copied while satisfying the constraints, and yields a currency-preserving extension of $\bar{\rho}$. □

Bounded extensions. In contrast to ECP, when it comes to deciding whether $\bar{\rho}$ can be made currency-preserving by copying data within a bounded size, the analysis becomes far more intricate. Indeed, the result below tells us that even for CQ, BCP is Σ_4^p-hard, and fixing denial constraints and copy functions does not help. When both queries and denial constraints are fixed, BCP is Σ_3^p-complete.

Theorem 6.4: For $\mathsf{BCP}(\mathcal{L}_Q)$, the combined complexity is

- Σ_4^p-complete when $\mathcal{L}_Q$ is CQ, UCQ or $\exists$FO$^+$, and
- PSPACE-complete when $\mathcal{L}_Q$ is FO.
- Its data complexity is Σ_3^p-complete when $\mathcal{L}_Q \in \{$CQ, UCQ, $\exists$FO$^+$,FO$\}$.

The combined complexity bounds remain unchanged when denial constraints and copy functions are fixed. □

Proof sketch: (1) *Lower bounds.* For FO, we show that BCP is PSPACE-hard by a simple reduction from Q3SAT.

For CQ, we show that BCP is Σ_4^p-hard by reduction from the $\exists^*\forall^*\exists^*\forall^*$3SAT problem, which is Σ_4^p-complete (cf. [28]). Given a sentence $\phi = \exists W \forall X \exists Y \forall Z \psi(W,X,Y,Z)$, we define a CQ query Q, a positive number k and a specification $\mathbf{S}$ with two copy functions $\bar{\rho}$. We show that ϕ is true iff there exists an extension $\bar{\rho}^e$ of $\bar{\rho}$ such that $|\bar{\rho}^e| \leq |\bar{\rho}| + k$ and $\bar{\rho}^e$ is currency preserving for Q. We use temporal instances in $\mathbf{S}$ to encode disjunction and negation, which Q leverages to express ϕ. We also define a data source in $\mathbf{S}$ whose current instance ranges over all truth assignments for W, and use the data source to inspect possible extensions of $\bar{\rho}$.

For the data complexity, we verify the Σ_3^p lower bound for CQ by reduction from the $\exists^*\forall^*\exists^*$3SAT problem. This reduction is more involved than the one developed for the combined complexity. Given an $\exists^*\forall^*\exists^*$ sentence ϕ, we define a query Q in CQ that is independent of ϕ but nonetheless, is able to encode the negations of the clauses in ϕ, by making use of temporal instances in a specification constructed.

In all these proofs, denial constraints and copy functions are independent of input sentences, *i.e.,* they are fixed.

(2) *Upper bounds.* We develop an algorithm for BCP, which first guesses an extension $\bar{\rho}^e$ of $\bar{\rho}$ that copies more data of a

bounded size, and then checks whether $\bar{\rho}^e$ is currency preserving, by invoking the algorithm for CPP. From Theorem 6.1 the upper bounds for BCP follow immediately. □

Special cases. Theorem 6.1, 6.2 and 6.4 motivate us to explore special cases that simplify the analyses. In contrast to Theorem 4.1 and Corollary 4.2, we have seen that fixing denial constraints does not make our lives easier when it comes to CPP, MCP or BCP. However, when denial constraints are absent, these problems become tractable for SP queries. This is consistent with Corollary 4.6.

Theorem 6.5: When denial constraints are absent, for SP queries both the combined complexity and the data complexity are in PTIME for CPP, MCP and BCP (when the bound k on the size of additional data copied is fixed). □

Proof sketch: (1) CPP. We develop a PTIME algorithm $\mathcal{A}$ that takes as input a specification $\mathbf{S}$ and an SP query Q defined on instances of a relation schema R. It checks whether the copy functions $\bar{\rho}$ in $\mathbf{S}$ are currency preserving for Q. The main idea behind $\mathcal{A}$ is as follows. Let E be the set of all entities in D_t, where D_t is the temporal instance of R in $\mathbf{S}$. Let C be the set of certain current answers to Q. By Corollary 4.6, C can be computed in PTIME. For each $e \in E$, $\mathcal{A}$ inspects tuples t in $\mathbf{D}'$ one by one, to find whether it would "spoil" some answer $s \in C$ produced by the current tuple of e if t were imported, *i.e.,* adding t would make $\mathsf{LST}(e, \mathbf{D}^c)$ either empty or a distinct tuple that does not produce s via Q. Here $\mathbf{D}'$ is the data source in $\mathbf{S}$ from which tuples are copied. When denial constraints are absent and Q is in SP, this can be done in PTIME. The algorithm then checks spoilers for all $e \in E$, to find whether there exists $s \in C$ spoiled by importing tuples for all entities that yield s, or whether some spoilers introduce a new certain current answer. This can be done in PTIME since the entities in E are independent of each other. The algorithm returns "yes" iff no such spoilers exist. It is in PTIME as argued above.

(2) MCP. Given $\mathcal{A}$, an algorithm for MCP is immediate: for each "reduced" $\bar{\rho}_c$ that removes one imported tuple from $\bar{\rho}$, it checks whether $\bar{\rho}_c$ is currency preserving for Q by using $\mathcal{A}$. The algorithm is obviously in PTIME.

(3) BCP. When the bound k is fixed, there are polynomially many extensions $\bar{\rho}^e$ of $\bar{\rho}$ such that $|\bar{\rho}^e| \leq |\bar{\rho}| + k$. For each such $\bar{\rho}^e$ we check whether $\bar{\rho}^e$ is currency preserving for Q by invoking $\mathcal{A}$. This can be done in PTIME. □

For $\mathsf{BCP}(\mathcal{L}_Q)$, when the bound k is predefined and fixed, the analysis also becomes simpler when $\mathcal{L}_Q$ is CQ, UCQ or $\exists$FO$^+$. For FO, however, BCP remains PSPACE-complete.

Corollary 6.6: When k is fixed, $\mathsf{BCP}(\mathcal{L}_Q)$ is

- Δ_4^p-complete (combined) for CQ, UCQ and $\exists$FO$^+$,
- Δ_3^p-complete (data) for CQ, UCQ and $\exists$FO$^+$, but is
- PSPACE-complete for FO (combined complexity). □

Proof sketch: (1) *Upper bounds.* As remarked earlier, when k is fixed there are polynomially many extensions $\bar{\rho}^e$ of $\bar{\rho}$ such that $|\bar{\rho}^e| \leq |\bar{\rho}| + k$. For each such $\bar{\rho}^e$ we check whether $\bar{\rho}^e$ is currency preserving, by invoking the algorithm for checking CPP given in the proof of Theorem 6.1. Therefore, BCP is in $P^{\Sigma_3^p} = \Delta_4^p$ for $\exists$FO$^+$(combined complexity). Moreover, when queries and denial constraints are fixed, it is in $P^{\Sigma_3^p} = \Delta_3^p$ (data complexity). For FO, the CPP checking is in PSPACE, and hence so is the upper bound for BCP.

(2) *Lower bounds.* We show that when k is fixed, BCP is already Δ_4^p-hard (combined complexity) for CQ queries, by reduction from the MSA($\exists^*\forall^*\exists^*$3SAT) problem. In addition, we verify that it is Δ_3^p-hard (data complexity) for CQ queries by reduction from the MSA($\exists^*\forall^*$3SAT) problem. For FO queries, we show that it remains PSPACE-hard (combined complexity) by reduction from Q3SAT. □

7. Conclusions

We have proposed a model to specify the currency of data in the absence of reliable timestamps but in the presence of copy relationships. We have also introduced a notion of currency preservation to assess copy functions for query answering. We have identified eight fundamental problems associated with data currency and currency preservation (CPS, COP, DCIP, CCQA($\mathcal{L}_Q$), CPP($\mathcal{L}_Q$), MCP($\mathcal{L}_Q$), ECP($\mathcal{L}_Q$) and BCP($\mathcal{L}_Q$)). We have provided a complete picture of the lower and upper bounds of these problems, all matching, for their data complexity as well as combined complexity when $\mathcal{L}_Q$ ranges over a variety of query languages. These results are not only of theoretical interest in their own right, but may also help practitioners distinguish current values from stale data, answer queries with current data, and design proper copy functions to import data from external sources.

The main complexity results are summarized in Tables 2 and 3, annotated with their corresponding theorems.

The study of data currency is still preliminary. An open issue concerns generalizations of copy functions. To simplify the presentation we assume a single copy function from one relation to another. Nonetheless we believe that all the results remain intact when multiple such functions coexist. For currency-preserving copy functions, we assume that the signatures "cover" all attributes (except EID) of the importing relation. It is nontrivial to relax this requirement, however, since otherwise unknown values need to be introduced for attributes whose value is not provided by the extended copy functions. A second issue is about practical use of the study. As shown in Tables 2 and 3, most of the problems are intractable. To this end we expect to (a) identify practical PTIME cases in various applications, (b) develop efficient heuristic algorithms with certain performance guarantees, and (c) conduct incremental analysis when data or copy functions are updated, which is expected to result in a lower complexity than its batch counterpart when the area affected by the updates is small, as commonly found in practice. A third issue concerns the interaction between data consistency and data currency. There is an intimate connection between these two central issues of data quality. Indeed, identifying the current value of an entity helps resolve data inconsistencies, and conversely, repairing data helps remove obsolete data. While these processes should logically be unified, we are not aware of any previous work on this topic. Finally, it is interesting to develop syntactic characterizations of currency-preserving copy functions.

Acknowledgments. Fan and Geerts are supported in part by an IBM scalable data analytics for a smarter planet innovation award, and the RSE-NSFC Joint Project Scheme.

8. References

[1] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison-Wesley, 1995.

	CPS	COP	DCIP
Data complexity	NP-complete (Th 4.1)	coNP-complete (Cor 4.2)	coNP-complete (Cor 4.2)
Combined complexity	Σ_2^p-complete (Th 4.1)	Π_2^p-complete (Cor 4.2)	Π_2^p-complete (Cor 4.2)
Special case	In the absence of denial constraints		
Combined and data complexity	PTIME (Th 4.4)	PTIME (Th 4.4)	PTIME (Th 4.4)

Table 2: Complexity of problems for reasoning about data currency (CPS, COP, DCIP)

Complexity	CCQA($\mathcal{L}_Q$)	CPP($\mathcal{L}_Q$)	MCP($\mathcal{L}_Q$)	ECP($\mathcal{L}_Q$)	BCP($\mathcal{L}_Q$)
Data	coNP-complete (Th 4.3)	Π_2^p-complete (Th 6.1)	Δ_2^p-complete (Th 6.2)	$O(1)$ (Prop 6.3)	Σ_3^p-complete (Th 6.4)
Combined ($\mathcal{L}_Q$)					
CQ, UCQ, $\exists$FO$^+$	Π_2^p-complete (Th 4.3)	Π_3^p-complete (Th 6.1)	Δ_3^p-complete (Th 6.2)	$O(1)$ (Prop 6.3)	Σ_4^p-complete (Th 6.4)
FO	PSPACE-complete (Th 4.3)	PSPACE-complete (Th 6.1)	PSPACE-complete (Th 6.2)	$O(1)$ (Prop 6.3)	PSPACE-complete (Th 6.4)
Special case	SP queries in the absence of denial constraints				
Combined & data	PTIME (Cor 4.6)	PTIME (Th 6.5)	PTIME (Th 6.5)	$O(1)$ (Prop 6.3)	PTIME (Th 6.5)

Table 3: Complexity of problems for query answering and for determining currency preservation

[2] L. Berti-Equille, A. D. Sarma, X. Dong, A. Marian, and D. Srivastava. Sailing the information ocean with awareness of currents: Discovery and application of source dependence. In *CIDR*, 2009.

[3] L. Bertossi. Consistent query answering in databases. *SIGMOD Rec.*, 35(2), 2006.

[4] M. Bodirsky and J. Kára. The complexity of temporal constraint satisfaction problems. *JACM*, 57(2), 2010.

[5] P. Buneman, J. Cheney, W. Tan, and S. Vansummeren. Curated databases. In *PODS*, 2008.

[6] J. Cheney, L. Chiticariu, and W. C. Tan. Provenance in databases: Why, how, and where. *Foundations and Trends in Databases*, 1(4):379–474, 2009.

[7] J. Chomicki. Consistent query answering: Five easy pieces. In *ICDT*, 2007.

[8] J. Chomicki and D. Toman. Time in database systems. In M. Fisher, D. Gabbay, and L. Vila, editors, *Handbook of Temporal Reasoning in Artificial Intelligence*. Elsevier, 2005.

[9] J. Clifford, C. E. Dyreson, T. Isakowitz, C. S. Jensen, and R. T. Snodgrass. On the semantics of "now" in databases. *TODS*, 22(2):171–214, 1997.

[10] E. F. Codd. Extending the database relational model to capture more meaning. *TODS*, 4(4):397–434, 1979.

[11] A. Deutsch, A. Nash, and J. B. Remmel. The chase revisited. In *PODS*, 2008.

[12] X. Dong, L. Berti-Equille, Y. Hu, and D. Srivastava. Global detection of complex copying relationships between sources. In *VLDB*, 2010.

[13] X. Dong, L. Berti-Equille, and D. Srivastava. Truth discovery and copying detection in a dynamic world. In *VLDB*, 2009.

[14] C. E. Dyreson, C. S. Jensen, and R. T. Snodgrass. Now in temporal databases. In L. Liu and M. T. Özsu, editors, *Encyclopedia of Database Systems*. Springer, 2009.

[15] W. W. Eckerson. Data quality and the bottom line: Achieving business success through a commitment to high quality data. Data Warehousing Institute, 2002.

[16] A. K. Elmagarmid, P. G. Ipeirotis, and V. S. Verykios. Duplicate record detection: A survey. *TKDE*, 19(1):1–16, 2007.

[17] W. Fan, F. Geerts, J. Li, and M. Xiong. Discovering conditional functional dependencies. *TKDE*, 23(4):683–698, 2011.

[18] M. Garey and D. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, 1979.

[19] G. Grahne. *The Problem of Incomplete Information in Relational Databases*. Springer, 1991.

[20] M. Grohe and G. Schwandtner. The complexity of datalog on linear orders. *Logical Methods in Computer Science*, 5(1), 2009.

[21] T. Imieliński and W. Lipski, Jr. Incomplete information in relational databases. *JACM*, 31(4), 1984.

[22] Knowledge Integrity. Two sides to data decay. DM Review, 2003.

[23] P. G. Kolaitis. Schema mappings, data exchange, and metadata management. In *PODS*, 2005.

[24] M. Koubarakis. Database models for infinite and indefinite temporal information. *Inf. Syst.*, 19(2):141–173, 1994.

[25] M. Koubarakis. The complexity of query evaluation in indefinite temporal constraint databases. *TCS*, 171(1-2):25–60, 1997.

[26] M. W. Krentel. Generalizations of Opt P to the polynomial hierarchy. *TCS*, 97(2):183–198, 1992.

[27] M. Lenzerini. Data integration: A theoretical perspective. In *PODS*, 2002.

[28] C. H. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.

[29] E. Schwalb and L. Vila. Temporal constraints: A survey. *Constraints*, 3(2-3):129–149, 1998.

[30] R. T. Snodgrass. *Developing Time-Oriented Database Applications in SQL*. Morgan Kaufmann, 1999.

[31] R. van der Meyden. The complexity of querying indefinite data about linearly ordered domains. *JCSS*, 54(1), 1997.

[32] R. van der Meyden. Logical approaches to incomplete information: A survey. In J. Chomicki and G. Saake, editors, *Logics for Databases and Information Systems*. Kluwer, 1998.

[33] V. Vianu. Dynamic functional dependencies and database aging. *J. ACM*, 34(1):28–59, 1987.

[34] H. Zhang, Y. Diao, and N. Immerman. Recognizing patterns in streams with imprecise timestamps. In *VLDB*, 2010.