CDMTCS
Research
Report
Series

Data-completeness Tailored Database Design with Embedded Functional Dependencies

Ziheng Wei
The University of Auckland

Sebastian Link
The University of Auckland

CDMTCS-537
May 2019

Centre for Discrete Mathematics and
Theoretical Computer Science

Data-completeness Tailored Database Design with Embedded Functional Dependencies

ZIHENG WEI
The University of Auckland, New Zealand
z.wei@auckland.ac.nz

SEBASTIAN LINK
The University of Auckland, New Zealand
s.link@auckland.ac.nz

May 23, 2019

Abstract

We establish a robust schema design framework for data with missing values. The framework is based on the new notion of an embedded functional dependency, which is independent of the interpretation of missing values, able to express completeness and integrity requirements on application data, and capable of capturing many redundant data value occurrences. We establish axiomatic, algorithmic, and logical foundations for reasoning about embedded functional dependencies. These foundations allow us to establish generalizations of Boyce-Codd and Third normal forms that do not permit any redundancy in any future application data, or minimize their redundancy across dependency-preserving decompositions, respectively. We show how to transform any given schema into application schemata that meet given completeness and integrity requirements and the conditions of the generalized normal forms. Data over those application schemata are therefore fit for purpose by design. Extensive experiments with benchmark schemata and data illustrate our framework, and the effectiveness and efficiency of our algorithms, but also provide quantified insight into database schema design trade-offs.

Keywords: Boyce-Codd Normal Form; Database design; Decomposition; Functional dependency; Key; Missing data; Normal Form; Redundancy; Synthesis; Third Normal Form

1 Introduction

SQL continues to be the de-facto industry standard and choice for data management. This holds true even after several decades of use and even in the light of new application data such as complex-value data, Web data, big data, or uncertain data. Nevertheless, even the simplest extensions to the underlying relational model cause significant issues in database practice. A prime example is the handling of incomplete data, which has attracted continuous interest from academics and practitioners over decades. While many advances can be celebrated, it is still unclear what a right notion of a query answer of incomplete data constitutes. In this research we are interested in the design of database schemata in the presence of incomplete data. SQL handles incomplete data by the use of a null marker, denoted by $\perp$, which indicates a missing data value occurrence on an attribute. The null marker is distinguished from actual data values, and treated differently. Different null marker occurrences, however, are treated uniformly in SQL to avoid processing overheads. Occurrences of null markers are on the rise for modern applications such as data integration. Indeed, in order to accommodate the integration of data from different schemata, frequent uses of null markers are necessary to conform to the structure of the underlying schema that integrates the data.

1.1 Desiderata

The primary aim of database design is to find a schema that facilitates the processing of the future application workload as well as possible. The well-known database design framework for relational databases is centered around the notion of data redundancy [5, 24, 26]. In practice, the majority of redundant data value occurrences originate from functional dependencies (FDs) [7]. These dependencies can express important integrity requirements of the underlying application domain. Informally, an FD $X \to Y$ expresses that every pair of tuples with matching values on all the attributes in X also has matching values on all the attributes in Y. Relational database design provides a framework to compute schemata in which the integrity requirements can be enforced efficiently during updates. The relational framework does not accommodate real-world requirements on the completeness of data, as no missing data is permitted to occur at all.

We will now summarize which properties an FD should have to advance database design in practice. There has been a plethora of research on finding suitable extensions for the notion of an FD to accommodate missing data. This has led to useful notions such as weak and strong FDs [21, 22], no information FDs [3, 15], or possible and certain FDs [18]. However, all of these notions assume that the same interpretation applies to all occurrences of the null marker in the given data set, such as “value does not exist”, “value exists but is unknown”, or “no information”. This sensitivity to a fixed null marker interpretation is difficult to justify in practice. Furthermore, it is not clear why we would want null marker occurrences to have an impact on the validity of an FD at all. Instead, it is more sensible to make the semantics of FDs dependant on actual data value occurrences only. This would mean the FD $X \to Y$ is satisfied whenever for every pair of records that has no null marker occurrences in any columns in the set union XY of X and Y, matching values on all attributes in X imply matching values on all the attributes in Y. Indeed, this notion is robust under different interpretations of null marker occurrences, and hence beyond guesswork.

Hence, *firstly* we desire a notion of an FD that is ignorant of missing data, in contrast to all previous notions as they are sensitive to it. *Secondly*, for a notion of an FD to be

$p(arent)$	$b(enefit)$	$c(hild)$
Homer	**610**	Bart
Homer	**610**	Lisa
Homer	915	$\perp$

Table 1: Sample r

$p(arent)$	$b(enefit)$
Homer	610

$p(arent)$	$c(hild)$
Homer	Bart
Homer	Lisa

Table 2: Lossless redundancy-eliminating decomposition of r from Table 1 for applications that require complete data values on c, p, and b

practically useful for schema design it should capture many redundant data value occurrences and facilitate lossless decompositions to eliminate them. *Thirdly*, applications for real-world data have not only integrity requirements but also completeness requirements. In contrast to previous work, we require our notion of an FD to accommodate completeness requirements in addition to the integrity requirements. *Fourthly*, we expect that the resulting schema design framework coincides with the well-known relational framework for the fragment of the data that meet the completeness requirements of the applications. In fact, the point of the framework is to tailor relational schema designs to the completeness and integrity requirements of applications.

1.2 Motivating example

We illustrate our desiderata on the simple example of Table 1. The schema collects information about the benefit that parents receive for all their children together. As the benefit changes, updates must be processed efficiently.

Robustness. It is easy to observe that the FD $p \to b$ does not hold, as the second and third tuple have matching (non-null) values on p but different values on b. Since no missing data is involved, this is true for any interpretation of null markers. Instead, for the FD $cp \to b$ the situation is quite different. If the sole occurrence of $\perp$ in r is interpreted as "value does not exist", then the FD should be true. If its interpretation is "value exists but unknown", then there are possible worlds of r (originating from the replacement of $\perp$ by actual values) that satisfy the FD and others that do not. Hence, under this interpretation, the FD is possible but not certain to hold in r. If we interpret $\perp$ as "no information", then the FD holds because there are no tuples with matching (non-null) values on p and c. Hence, if we do not know which interpretation applies to a given null marker occurrence, or if different null marker occurrences have different interpretations, then the semantics of an FD is not robust.

Data redundancy and their elimination by lossless decompositions. Under our robust semantics, the FD $cp \to b$ is satisfied. However, since there are no tuples with matching (non-null) values on c and p, this FD does not capture any redundant data value occurrences. In fact, we cannot express that the FD $p \to b$ actually holds on the subset of all tuples that have no missing values on attribute c. This novel observation leads us to the new notion of an *embedded functional dependency* (eFD). This is a statement $E : X \to Y$ where E is a set of attributes and $X, Y \subseteq E$. Indeed, E defines the subset of tuples $r^E \subseteq r$ that have no missing data on any of the attributes in E. We require $X, Y \subseteq E$ to make our notion of an FD robust, as explained before. For convenience, we sometimes simply write $E - XY : X \to Y$, understanding implicitly that all attributes in XY belong to E. We call an eFD $E : X \to Y$ *pure* whenever $E - XY$ is non-empty. In our example, we obtain the eFD $c : p \to b$, which clarifies the roles of the attributes c and p: p functionally determine b whenever c (and p and b) have no missing values. Now, our example shows that the eFD $c : p \to b$ captures redundant data value occurrences. Each of the two values is redundant in the classical sense [29] that every change to one of those values to a different value will result in a violation of the eFD $c : p \to b$. Our experiments will show that pure eFDs cause a significant number of redundant data value occurrences that cannot be captured by previously studied FDs, that is, eFDs of the special case $\emptyset : X \to Y$. In fact, we will show that pure eFDs are frequent among those eFDs that cause the most redundant data value occurrences on real-world benchmark data. The ability to capture many redundant data values enables eFDs to facilitate new lossless decompositions that can eliminate those redundancies.

Data completeness. The need to accommodate data quality requirements are another driver for our notion of an eFD. An eFD $E : X \to Y$ enables users to declare completeness as well as integrity requirements, and to carefully distinguish between these two data quality dimensions. In fact, this notion now allows us to separate dependence from completeness concerns: the attribute subsets X and Y form the actual FD $X \to Y$ which must hold on the subset r^E of all tuples that have no missing data on attributes in X, Y, and $E - XY$. In fact, as r^E satisfies the FD $X \to Y$ over relation schema R, r^E is the lossless join over its two projections on XY and $X(R - Y)$. Hence, we can eliminate all redundant data values on Y caused by the eFD $E : X \to Y$ without a loss of information for the fragment r^E of our application data that meets the requirement of having complete data values on all columns in E. This illustrates that eFDs drive data-completeness tailored database design.

In summary, the example shows that eFDs are different from previous FDs in three aspects: they are robust under different interpretations of missing data and accommodate completeness requirements, they capture redundant data values occurrences that could not be captured before, and they facilitate lossless decompositions that eliminate redundant values for the data that is meeting the completeness requirements. Table 2 shows a lossless decomposition for the fragment of our example relation r that meets the requirements for tuples to be complete on c, p, and b, and eliminates the redundant data value occurrences from r. This is not achievable for previous notions of FDs. Hence, changes to the benefit b for a parent just require one update on the decomposed relation. As the data evolves, more tuples may meet the completeness requirements. For example, if the occurrence of $\perp$ in r is updated to *Maggie*, then the resulting relation violates our eFD $c : p \to b$. In response we may update both value occurrences of 610 on b to 915 to reflect the new information that Homer has three children. On the decomposed schema, this would be represented by an insertion of *(Homer, Maggie)* and a single update of the value 610 on *benefit* to 915.

1.3 Contributions

We establish the first fully-fledged framework that brings forward schema designs tailored to the data completeness requirements of applications.

- We propose the new notion of an *embedded functional dependency* (eFD). We demonstrate that eFDs provide an intuitive and useful notion for database schema design in practice.
- We develop a full design theory including axiomatic and algorithmic characterizations of the implication problem for eFDs. Just like reasoning about FDs is indispensable for relational normalization, our design theory for eFDs is essential to our design framework.
- We establish a schema design framework that accommodates completeness and integrity requirements of applications, based on suitable extensions of a) the notion of redundant data values, b) Boyce-Codd normal form, and c) Third normal form.
- We show how to embed data completeness requirements into the relational normalization framework without additional overheads, including BCNF decomposition and 3NF synthesis algorithms. For data that does not meet the completeness requirements, previous approaches that are sensitive to the interpretation of null markers can be applied.
- We conduct comprehensive experiments on real-world benchmark schemata and data. In particular, we provide insight on how many redundant data values occur in the data, rank the relevance of our eFDs by how many data redundancies they cause, show how often schemata satisfy a normal form condition, how much redundancy 3NF syntheses permit, how many dependencies BCNF decompositions preserve, and how large decomposed schemata become. We consider the times of computations, and suggest by examples how data stewards can use our ranking of eFDs.

1.4 Organization

We explain our contributions over related work in Section 2, fix notation and propose eFDs in Section 3. Schema design foundations are developed in Section 4, normal forms are proposed in Section 5, and normalization is discussed in Section 6. An analysis of our comprehensive experiments is presented in Section 7. We conclude and outline future work in Section 8. More details, including proofs, can be found in the appendix and here[1].

2 Related Work

We review previous work to emphasize our contributions.

---

[1] `https://bit.ly/2AoOis6`

Firstly, our core objective is tailoring classical schema design to data-completeness requirements of applications. Hence, the achievements of classical schema design are fundamental to our work. These achievements are handsomely summarized in surveys and textbooks [6,24]. As our article starts research on data-quality driven database design, we focus on the most common class of integrity constraints and source of data redundancy, namely functional dependencies. More general constraints such as join or inclusion dependencies are left for future work [9,12,23,28]. Hence, we are interested in Boyce-Codd Normal Form (BCNF) [8,16] and Third Normal Form (3NF) [7], with their well-known tradeoffs [6,11,20]: Any relation schema enjoys lossless BCNF decompositions that eliminate all data redundancy caused by FDs, but may not be dependency-preserving. On the other hand, every relation schema enjoys lossless 3NF syntheses that are guaranteed to be dependency-preserving, but may not eliminate all data redundancy. Our work subsumes all of these results as the special case where an application requires that no data values are missing, that is, when $E = R$. Important for these achievements is Vincent's classical notion of data redundancy [29], which we generalize to a notion of data redundancy under data completeness requirements. This makes it possible to clearly state and demonstrate the achievements of our normal forms. Recently, [25] described how to combine classical FD discovery with classical BCNF-decomposition to drive schema normalization from data. They did not consider data quality criteria, and handled nulls like any other value.

Secondly, schema design for data with missing values has been a long-standing open problem [14,18,22]. Almost exclusively, the main focus of the research has been on suitable extensions of FDs to incorporate null markers. In that area there is a plethora of research, mostly focusing on foundational aspects such as reasoning. Among those extensions, there are some approaches where null-free sub-schemata have been considered for reasoning about FDs [3,15]. That work is different from our approach, and focused on reasoning rather than schema design. In particular, the approach is more restricted because null-free sub-schemata do not permit any null marker occurrences in the columns of the sub-schema. Instead, we do permit null marker occurrences in any columns, but let the application requirements decide whether such records should be considered for design decisions. An interesting FD extension are weak and strong FDs that hold in some or all possible worlds of data sets with missing values [21,22]. As with other extensions, the semantics of the extended FDs depends strongly on the interpretation of the null marker occurrences. This makes it difficult to address modern applications, such as data integration, where different null marker occurrences may require different interpretations. A second limitation is that the complexity of reasoning becomes often prohibitively expensive to guarantee efficient schema designs [22]. Recently, the concept of possible and certain FDs was introduced and shown to provide suitable extensions of schema design for data with missing values, at least in terms of BCNF [18]. The work contains a review and comparison of FD extensions to data with missing values. We refer the interested reader to this survey. In summary, these approaches focus on the interpretation of null markers, aiming at their inclusion in decisions about schema design. These approaches can be criticized in different respects. Firstly, it is doubtful whether missing information should have an impact on schema design decisions. Secondly, modern applications such as data integration accommodate missing data values that require different interpretations, which makes it difficult to justify these approaches. Finally, application requirements

tuple_id	*f(_name)*	*l(_name)*	*c(ity)*	*z(ip)*	*p(hone)*	*d(ate_register)*
t_1	sam	anderson	green level	27217	$\perp$	05/11/1940
t_2	ida	cheek	**burlington**	27217	226 4848	05/11/1940
t_3	effie	massey	**burlington**	27217	336 226 8544	05/11/1940
t_4	peggy	floyd	jackson	27845	252 536 2668	06/15/1936
t_5	essie	warren	lasker	27845	252 539 2866	05/10/1940
t_6	rodney	glockner	wilmington	28405	910 509 3864	01/01/1930
t_7	sallie	blanchard	rose hill	28458	910 289 3320	01/01/1930
t_8	joseph	cox	new bern	28562	$\perp$	03/06/1935
t_9	joseph	cox	new bern	28562	252 288 4763	03/06/1935
t_{10}	james	smith	chinquapin	28521	910 285 3493	01/01/1936
t_{11}	james	smith	burlington	27215	584 4202	05/06/1940
t_{12}	dorothy	faucette	mebane	27302	919 563 1285	4/05/1940
t_{13}	dorothy	allred	mebane	27302	563 1426	05/06/1940
t_{14}	eloise	croom	kinston	28504	252 569 4436	05/02/1940
t_{15}	$\perp$	croom	kinston	28504	252 569 9516	05/04/1940

Table 3: Sample snippet r from *ncvoter* benchmark data set

have not been considered in these approaches, even though design decisions should be based on them. In contrast, our approach bases decisions about the design only on information that is available. That is, we design schemata beyond guesswork by considering complete data fragments. At the same time, application requirements become the primary focus point of our approach. In fact, the requirements can be declared as part of the FDs.

Thirdly, embedded unique constraints (eUCs) and embedded cardinality constraints (eCCs) have been investigated in previous work [30, 31]. Those articles investigated primarily their implication problem. In particular, the embedded version of a unique or cardinality constraint with embedding E holds on a data set r whenever the uniqueness or cardinality constraint holds on the scope r^E of the given data set r. Our results on the implication problem for the combined class of eUCs and eFDs subsumes the results of [31] on the individual class of eUCs. Neither eFDs nor data-completeness tailored database design have been considered before.

Finally, an important extension of classical FDs are conditional FDs (cFDs) [13], which encode data quality rules that target the cleaning of data without missing values but not schema design for data with missing values. Specifically, eFDs encode data completeness requirements and are a major source of E-data redundancy. This makes them important for schema design. Note that the implication problem of general cFDs is coNP-complete to decide [13], and already their consistency problem is NP-complete [13]. In contrast, classical FDs are always consistent and implication is linear-time decidable. As we show, this is also achieved by eFDs, which are therefore suited for schema design purposes from a computational point of view as well.

In summary, our work marks the first approach to tailor classical database schema designs to data-completeness requirements of applications.

3 Embedded Constraints

We introduce the data model and constraints.

We begin with basic terminology. A *relation schema* is a finite non-empty set R of *attributes*. Each attribute A of a relation schema R is associated with a domain $dom(A)$ which represents the possible values that can occur in column A. In order to encompass incomplete information, the domain of each attribute contains the null marker, denoted by $\perp$. In line with SQL, and to cater for all different types of missing values, the interpretation of $\perp$ is to mean "no information" [32]. We stress that the null marker is not a domain value. In fact, it is a purely syntactic convenience that we include the null marker in the domain of each attribute.

For attribute sets X and Y we may write XY for their set union $X \cup Y$. If $X = \{A_1, \dots, A_m\}$, then we may write $A_1 \cdots A_m$ for X. In particular, we may write A to represent the singleton $\{A\}$. A *tuple* (or *record*) over R is a function $t : R \to \bigcup_{A \in R} dom(A)$ with $t(A) \in dom(A)$ for all $A \in R$. For $X \subseteq R$ let $t(X)$ denote the restriction of the tuple t over R to X. We say that a tuple t is X*-total* (X*-complete*) if $t(A) \neq \perp$ for all $A \in X$. A tuple t over R is said to be a *total tuple* if it is R-total. A *relation* r over R is a finite set of tuples over R. A relation r over R is a *total relation* if every tuple $t \in r$ is total. The sub-set r^X of X-total tuples in r is called the *scope* of r with respect to X, or simply the scope of r when X is fixed. We say that two tuples t, t' over R have matching values on an attribute $A \in R$ whenever $t(A) = t'(A)$.

A *key* over R is a subset $X \subseteq R$. A total relation r over R satisfies the key X over R whenever there are not any two distinct tuples in r with matching values on all the attributes in X. A *functional dependency* (FD) over R is an expression $X \to Y$ with $X, Y \subseteq R$. A total relation r over R satisfies the FD $X \to Y$ over R whenever every pair of records in r with matching values on all the attributes in X has also matching values on all the attributes in Y. The semantics of keys and FDs can be extended to relations with missing values by adopting uniformly (that is, for all null marker occurrences) either the $\perp = \perp$ or $\perp \neq \perp$ semantics. Under either of these semantics, $\perp$ is considered to be different from any actual domain value. Other semantics have been defined, and we refer the interested reader to [18] for an overview of those. In a nutshell, different semantics lead to different notions of constraints each of which is useful in different contexts. However, in classical but even more in modern applications such as data integration, different null marker occurrences in a relation may require different interpretations. This makes it difficult, if not impossible, to justify any uniform interpretation of $\perp$.

In this article we take a different approach. Firstly, we let the application decide which data completeness requirements tuples must meet to be fit for use by the application. That is, we embed the data completeness requirements in the declaration of constraints. Secondly, the semantics of our constraints is based exclusively on the complete information embedded in the underlying relation. In other words, we follow the principled approach that missing values must not impact the decision whether a constraint is satisfied by the given relation or not. This decision is entirely determined by the actual data values that are available.

Similar ideas motivated us [31] to introduce embedded unique constraints (eUCs). Given a relation schema R, an *embedded unique* (eUC) is an expression of the form

$E : U$ where $U \subseteq E \subseteq R$ holds. A relation r satisfies the eUC $E : U$ iff the scope $r^E = \{t \in r \mid \forall A \in E(t(A) \neq \perp)\}$ of r with respect to E satisfies the key U. If $E = U$, the eUC $U : U$ is satisfied by relation r iff the key U is satisfied by r using the $\perp \neq \perp$ semantics iff the SQL unique constraint on U is satisfied by r. Of course, if E contains some attribute that is not in U, then the semantics of eUCs cannot be captured by any other notion of a key. The decision to require $U \subseteq E$ ensures that the semantics of the eUC only depends on the complete fragments embedded in the given relation. This motivates the following definition.

Definition 1 *Given a relation schema R, an* embedded functional dependency *(eFD) is an expression of the form $E : X \to Y$ where $XY \subseteq E \subseteq R$ holds. A relation r satisfies the eFD $E : X \to Y$ if and only if the scope r^E of r with respect to E satisfies the functional dependency $X \to Y$.*

Given $E : X$ or $E : X \to Y$, we sometimes simply write $E - X : X$ or $E - (XY) : X \to Y$, respectively, to emphasize which additional attributes apart from those in XY are required to have no missing values. The choice of E is based on several factors, such as completeness requirements and the target of redundant data values.

Consider our running example from Table 3 where the underlying schema R consists of attributes c, d , f, l, p, and z. The eFD $p : dz \to c$ is satisfied by r because the FD $dz \to c$ holds on the scope $r^{cdpz} = r - \{t_1, t_8\}$. In fact, all people who provided some phone number and registered on the same day under the same zip code also used the same city alias. However, the eFD $\emptyset : dz \to c$ does not hold on r because the FD $dz \to c$ does not hold on the scope $r^{cdz} = r$. In fact, there are people who registered on the same day under the same zip code, but used different alias for the city. Similarly, the eUC $p : cfl$ is satisfied by r because the compound key cfl is satisfied on the scope $r^{cflp} = r - \{t_1, t_8\}$. In fact, there are no two people who both provided a phone number and are registered in the same city with the same first and last name. However, the eUC $\emptyset : cfl$ is violated by r because there are two different registrations in the same city with the same first and last name.

Every total relation over R satisfies the FD $X \to R$ iff it satisfies the key X. This relationship occurs in our framework as well: A relation over R satisfies the eFD $R : X \to R$ iff it satisfies the eUC $R : X$. Indeed, if a relation satisfies $E : X$, then it also satisfies $E : X \to E$, but not necessarily vice versa. In fact, the relation in Table 3 satisfies the eFD $dpl : dp \to dpl$, but it violates the eUC $dpl : dp$. This observation is important, as it does not suffice for our targeted schema design framework to consider eFDs in isolation from eUCs. In particular, eFDs drive data redundancy, while eUCs inhibit data redundancy. Hence, the combined class of eFDs and eUCs needs to be studied. This is different from the special case of total relations where any key X over R (an eUC of type $R : X$) can be expressed by the FD $X \to R$ (an eFD of type $R : X \to R$).

For our example we regard the eFD $p : dz \to c$ as a meaningful constraint of our application domain. That is, for people that provide some phone number and register on the same day under the same zip code we will always use the same city alias. Hence, different city alias may only be associated with the same zip code for people on different registration dates or who prefer not to provide a phone number. In this case, there are

$\overline{R : R}$ (trivial ekey)	$\dfrac{E : U}{EE' : UU'}$ (eUC extension)	
$\overline{E : XY \to X}$ (trivial eFD)	$\dfrac{E : X \to Y}{E : X \to XY}$ (eFD extension)	$\dfrac{E : X \to Y \quad E' : Y \to Z}{EE' : X \to Z}$ (eFD transitivity)
$\dfrac{E : X}{E : X \to E}$ (eUC to eFD)	$\dfrac{E : XY \quad E : X \to Y}{E : X}$ (eUC pullback)	

Table 4: Axiomatizations for eUCs and eFDs

relations that exhibit data redundancy. Indeed, each of the two bold *city* occurrences of 'burlington' in tuples t_2 and t_3 in Table 3 is redundant. However, such redundant values are intrinsically linked to the requirement that the tuples must be complete on *phone*, since the eFD does not apply otherwise. This link between data redundancy and data completeness requirements is encoded explicitly in the eFDs. In what follows, we will develop a full-fledged normalization framework that tailors classical schema design to data completeness requirements.

4 Foundations

This section establishes axiomatic and algorithmic characterizations of the implication problem for eUCs and eFDs. The linear-time decidability we establish is important for the schema design framework we will develop subsequently.

Let $\Sigma \cup \{\varphi\}$ denote a set of eUCs and eFDs over relation schema R. We say that Σ *implies* φ, denoted $\Sigma \models \varphi$, iff every relation over R that satisfies all $\sigma \in \Sigma$ also satisfies φ. The *implication problem* for a class $\mathcal{C}$ of constraints is to decide, for arbitrary R and $\Sigma \cup \{\varphi\}$ in $\mathcal{C}$, whether Σ implies φ. Strictly speaking, the implication problem we have just defined is the finite implication problem because we restrict relations to be finite. Permitting also infinite relations would lead us to the unrestricted implication problem. For our class of constraints, however, it is easy to see that finite and unrestricted implication problems coincide. We will therefore speak of *the* implication problem.

4.1 Axiomatic Characterization

Firstly, we would like to obtain an axiomatization for eUCs and eFDs which extends the well-known Armstrong axioms [2]. An axiomatization does not only help us understand the interaction of the constraints, but also enables us to prove that our syntactic normal form proposal captures precisely the semantic normal form proposal in which no redundant data value can ever occur in any future database instance. This is an important use case of axiomatizations. The definitions of inference from a system $\mathfrak{S}$ ($\vdash_{\mathfrak{S}}$), as well as the definitions of *sound* and *complete* sets of inference rules are standard [24, 26].

$X^+_{E,\Sigma}$	$E - X^+_{E,\Sigma}$	$R - E$
0⋯0	0⋯0	0⋯0
0⋯0	1⋯1	$\perp \cdots \perp$

d	*z*	*c*	*p*	*f*	*l*
5/11/1940	27217	burlington	226 4848	ida	cheek
5/11/1940	27217	green level	$\perp$	$\perp$	$\perp$

Table 5: 2-tuples for Theorem 1 proof and sample

Table 4 shows three axiomatizations. The top box is one for eUCs alone [31], the middle box is one for eFDs alone, and all boxes form the axiomatization $\mathfrak{E}$ for eUCs and eFDs together. The following rules

$$\underset{\text{(eFD decompose)}}{\frac{E : X \to YZ}{E : X \to Y}} \qquad \underset{\text{(eFD union)}}{\frac{E : X \to Y \quad E : X \to Z}{E : X \to YZ}} \qquad \underset{\text{(eFD add-on)}}{\frac{E : X \to Y}{EE' : X \to Y}}$$

follow from $\mathfrak{E}$.

Theorem 1 *$\mathfrak{E}$ is a sound and complete axiomatization for the implication of eUCs and eFDs.*

The proof of Theorem 1 is based on the *closure* of an attribute set X with respect to the data completeness requirement E and the set Σ of eUCs and eFDs: $X^+_{E,\Sigma} = \{A \in E \mid \Sigma \vdash_{\mathfrak{E}} E : X \to A\}$. Indeed, the completeness proof uses the two-tuple relation from Table 8 to show that Σ does not imply φ whenever φ cannot be inferred from Σ using $\mathfrak{E}$. Hence, the implication problems for eUCs and eFDs coincide, independently of whether we consider infinite relations, just finite relations, or even just relations with two tuples.

The rule

$$\frac{R : X \to R}{R : X}$$

together with the (eUC to eFD)-rule with $E = R$ express that keys are special cases of functional dependencies in the relational model of data. The rule above can be inferred as follows:

$$\frac{\overline{R : R} \quad \overline{R : X \to R}}{R : X}$$

However, this situation does not generalize to the embedded setting. That is, the rule

$$\frac{E : X \to E}{E : X}$$

is not sound as the following counterexample shows:

E(mployee)	*D(epartment)*	*M(anager)*
Sheldon	Physics	Siebert
Sheldon	Physics	$\perp$

It satisfies $ED : E \to D$ but violates $ED : E$.

Example 1 *Consider our running example where* $R = \{c, d, f, l, p, z\}$ *and* $\Sigma = \{p : cfl, p : dz \to c\}$. *Then* Σ *implies the eUC* $\varphi = c : dflpz$ *as the following inference shows:*

$$\dfrac{\dfrac{}{cdflpz : cdflpz} \qquad \dfrac{\dfrac{\dfrac{}{cdflpz : dflpz \to dz} \quad p : dz \to c}{cdflpz : dflpz \to c}}{cdflpz : dflpz \to cdflpz}}{cdflpz : dflpz} \; .$$

However, the eFD $\emptyset : dz \to c$ *is not implied by* Σ *as the two-tuple example in Table 8 constructed according to the general two-tuple relation from Table 8 shows.*

4.2 Algorithmic Characterization

Reasoning efficiently about eUCs and eFDs will help us decide if a given schema meets a normal form condition, or transform the schema into one that meets the condition. In the relational model, FD implication can be decided in linear time. We will achieve the same for eUCs and eFDs. Many other tasks, including data profiling, transaction processing, and query optimization, benefit from the ability to efficiently decide implication.

Let E denote an attribute set that represents the completeness requirements of a given application. The technical underpinning of our framework translates every set Σ of eUCs and eFDs into a set $\Sigma[E]$ of FDs. The translation makes it possible to utilized any existing algorithms for deciding FD implication to decide implication of eUCs and eFDs. The translation is given next by the following definition.

Definition 2 *For a given set* Σ *of eUCs and eFDs over relation schema* R, *and a given attribute set* $E \subseteq R$, *let* $\Sigma[E] := \{X \to R \mid \exists E' \subseteq E(E' : X \in \Sigma)\} \cup \{X \to Y \mid \exists E' \subseteq E(E' : X \to Y \in \Sigma)\}$ *denote the* (E,FD)-reduct *of* Σ.

We illustrate the notion of an *(E,FD)*-reduct on our running example.

Example 2 *Recall that* $R = \{c, d, f, l, p, z\}$ *and* $\Sigma = \{p : cfl, p : dz \to c\}$. *For* $E = cdflpz = R$, *the* (E,FD)*-reduct of* Σ *is* $\Sigma[E] = \{cfl \to dpz, dz \to c\}$.

The importance of the *(E,FD)*-reduct is embodied in the following algorithmic characterization of the implication problem for eUCs and eFDs.

Theorem 2 *Let* $\Sigma \cup \{E : X, E : X \to Y\}$ *denote a set of eUCs and eFDs over relation schema* R. *Then:*

1. $\Sigma \models E : X \to Y$ *if and only if* $\Sigma[E] \models X \to Y$
2. $\Sigma \models E : X$ *if and only if a)* $E = R = X$, *or b) there is some* $E' : X' \in \Sigma$ *such that* $E' \subseteq E$ *and* $X' \subseteq X^+_{\Sigma[E]}$.

Here, *1.* says that the eFD $E : X \to Y$ is implied by the eUC/eFD set Σ if and only if the FD $X \to Y$ is implied by the *(E,FD)*-reduct $\Sigma[E]$ of Σ. Furthermore, *2.* says that the eUC $E : X$ is implied by the eUC/eFD set Σ if and only if the eUC $E : X$ is the trivial eUC $R : R$, or there is another eUC $E' : X'$ in Σ such that $E' \subseteq E$ and the *(E,FD)*-reduct $\Sigma[E]$ implies the FD $X \to X'$.

An analysis of Theorem 2 results in the proposal of Algorithm 1 for deciding our implication problem. If $\varphi = R : R$, we answer yes (lines 3/4). Otherwise, standard algorithms compute the closure $X^+_{\Sigma[E]}$ of the attribute set X given $\Sigma[E]$ (lines 6/7). If φ is an eUC and *2.* in Theorem 2 is met, then we answer yes (lines 8/9). If φ is an eFD and *1.* in Theorem 2 is met, then we answer yes (lines 11/12). Otherwise, we answer no (lines 13/14).

Algorithm 1 Deciding Implication

```
 1: Input:  Set $\Sigma \cup \{\varphi\}$ of eUCs and eFDs over schema $R$
 2: Output: { Yes  , if $\Sigma \models \varphi$
            { No   , otherwise
 3: if $\varphi = R : R$ then
 4:     return(Yes);
 5: else
 6:     if $\varphi = E : X$ or $\varphi = E : X \to Y$ then
 7:         Compute $X^+_{\Sigma[E]}$;                      ▷ FD attribute set closure
 8:         if $\varphi = E : X \wedge \exists E' : X' \in \Sigma(E' \subseteq E \wedge X' \subseteq X^+_{\Sigma[E]})$ then
 9:             return(Yes);
10:         else
11:             if $\varphi = E : X \to Y \wedge Y \subseteq X^+_{\Sigma[E]}$ then
12:                 return(Yes);
13:             else
14:                 return(No);
```

The soundness of Algorithm 1 follows from Theorem 2, linear time decidability from that of FD implication [4], and PTIME-hardness from a reduction of HORN-SAT [10,17].

Corollary 1 *The implication problem of eUCs and eFDs is PTIME-complete. On input* $(\Sigma \cup \{\varphi\}, R)$*, Algorithm 1 decides the implication problem* $\Sigma \models \varphi$ *in time* $\mathcal{O}(|\Sigma \cup \{\varphi\}|)$.

Note that the PTIME-hardness means that deciding implication for eUCs and eFDs is at least as hard as any other decision problem for which there is some deterministic Turing machine that can decide any instance of the problem in polynomial time.

Example 3 *In our running example* $R = \{c, d, f, l, p, z\}$ *and* $\Sigma = \{p : cfl, p : dz \to c\}$, Σ *implies* $\varphi = c : dflpz$ *since* $\Sigma[cdflpz] = \{cfl \to dpz, dz \to c\}$*, there is some eUC* $E' = cflp : cfl = X' \in \Sigma$ *such that* $E' \subseteq E$*, and* $X' \subseteq (dflpz)^+_{\Sigma[cdflpz]} = cdflpz$*, which means* 2. *of Theorem 2 is met. The eFD* $\emptyset : dz \to c$ *is not implied by* Σ *as* $\Sigma[E] = \Sigma[cdz] = \emptyset$*, and* $dz \to c$ *is not implied by* $\Sigma[E]$*, which means* 1. *of Theorem 2 is not met.*

As a summary, our axiomatization $\mathfrak{E}$ will enable us to formally justify the syntactic normal form proposals we will put forward in the following section, while our algorithmic characterizations will facilitate our normalization strategy to apply well-known relational decomposition and synthesis approaches to the *(E,FD)*-reduct of an input set of embedded unique constraints and embedded functional dependencies.

5 Normal Forms

Our goal is to tailor relational schema design to data completeness requirements of applications. For that purpose, we stipulate the semantic normal form condition that no redundant data values can ever occur in any E-complete records on any relations that satisfy a given set of eUCs and eFDs. We will characterize this condition by generalizing the well-known Boyce-Codd normal form. Similarly, we are able to generalize the well-known 3NF to characterize the minimization of data redundancy in E-complete records across all dependency-preserving decompositions.

5.1 E-Redundancy Free Normal Form

Motivated by our examples, we propose notions of data redundancy that are tailored towards the requirements of records regarding their completeness. For this, we generalize the following classical proposal by Vincent [29]. Intuitively, a data value in a relation that satisfies a constraint set Σ is redundant if every update to a different value results in a relation that violates some constraint in Σ. Formally, for relation schema R, attribute A of R, tuple t over R, and set Σ of constraints over R, a *replacement* of $t(A)$ is a tuple $\bar{t}$ over R such that: i) for all $\bar{A} \in R - \{A\}$ we have $\bar{t}(\bar{A}) = t(\bar{A})$, and ii) $\bar{t}(A) \neq t(A)$. For a relation r over R that satisfies Σ and $t \in r$, the data value occurrence $t(A)$ in r is *redundant* for Σ if for every replacement $\bar{t}$ of $t(A)$, $\bar{r} := (r - \{t\}) \cup \{\bar{t}\}$ violates some constraint in Σ. A relation schema R is in *Redundancy-Free normal form* (RFNF) for a set Σ of constraints if there are no relation r over R that satisfies Σ, tuple $t \in r$, and attribute $A \in R$, such that the data value occurrence $t(A)$ is redundant for Σ [29]. In other words, we guarantee at design time that there will never be an instance over R that satisfies Σ and has some redundant data value occurrence.

Definition 3 *Let R denote a relation schema, $E \subseteq R$, Σ a set of constraints over R, $A \in E$ an attribute, r a relation over R that satisfies Σ, and t a tuple in r^E. An E-*replacement *of $t(A)$ is a replacement of $t(A)$ that is E-complete. The data value occurrence $t(A)$ is* E-redundant *for Σ if and only if for every E-replacement $\bar{t}$ of $t(A)$, $\bar{r} := (r - \{t\}) \cup \{\bar{t}\}$ violates some constraint in Σ. R is in E-Redundancy-Free normal form (E-RFNF) for Σ if and only if there are no relation r over R that satisfies Σ, tuple $t \in r^E$, and attribute $A \in E$, such that the data value occurrence $t(A)$ is E-redundant for Σ.*

We illustrate the notion of E-redundancy next.

Example 4 *Recall that $R = cdflpz$ and $\Sigma = \{p : cfl, p : dz \to c\}$. The relation in Table 3 shows that R is not in $cdpz$-RFNF for Σ: every $cdpz$-replacement for either of the bold occurrences would violate the eFD $p : dz \to c$.*

While E-RFNF is independent of the type of constraints, we will assume from now on that Σ is a set of eUCs and eFDs. As our first result we characterize the E-RFNF for Σ in terms of the RFNF for the (E, FD)-reduct $\Sigma[E]$.

Theorem 3 *For all sets Σ over R and all $E \subseteq R$, R is in E-RFNF for Σ if and only if R is in RFNF for $\Sigma[E]$.*

For our example the characterization works as follows.

Example 5 *For $R = cdflpz$, $\Sigma = \{p : cfl, p : dz \rightarrow c\}$ and $E = cdpz$, we have $\Sigma[E] = \{dz \rightarrow c\}$. That is, R is also not in RFNF for $\Sigma[E]$.*

5.2 E-BCNF

We now characterize the semantic E-RFNF by purely syntactic means. For that purpose, we generalize the BCNF condition to accommodate completeness requirements. Recall that a relation schema R is in BCNF for an FD set Σ iff for all $X \rightarrow Y \in \Sigma^+_{\mathfrak{A}}$ where $Y \nsubseteq X$, $X \rightarrow R \in \Sigma^+_{\mathfrak{A}}$. Here, $\mathfrak{A}$ denotes the well-known Armstrong's axioms [2].

Definition 4 *For relation schema R and $E \subseteq R$, R is in E-*BCNF *for a set Σ over R if and only if for every eFD $E : X \rightarrow Y \in \Sigma^+_{\mathfrak{E}}$ where $Y \nsubseteq X$, $E : X \in \Sigma^+_{\mathfrak{E}}$.*

Our running example can be further analyzed as follows.

Example 6 *For $R = cdflpz$ and $\Sigma = \{p : cfl, p : dz \rightarrow c\}$, R is not in cdpz-BCNF for Σ, since the eFD $p : dz \rightarrow c \in \Sigma \subseteq \Sigma^+_{\mathfrak{E}}$, $\{c\} \nsubseteq \{dz\}$, but $dpz : dz \notin \Sigma^+_{\mathfrak{E}}$.*

Recall that sets Σ and Θ are $\mathcal{C}$-*covers* of one another if they imply the same constraints in class $\mathcal{C}$. Being in E-BCNF for Σ is independent of the representation of Σ. That is, for any cover Θ of Σ, R is in E-BCNF for Σ iff R is in E-BCNF for Θ. The E-BCNF condition for Σ can be characterized by the BCNF condition for $\Sigma[E]$.

Theorem 4 *Relation schema R is in E-BCNF for the set Σ if and only if R is in BCNF for $\Sigma[E]$.*

Theorem 4 works as follows on our example.

Example 7 *For $R = cdflpz$, $\Sigma = \{p : cfl, p : dz \rightarrow c\}$, and $E = cdpz$, R is not in BCNF for $\Sigma[E] = \{dz \rightarrow c\}$.*

5.3 E-RFNF at Application Design Time

We can now characterize the semantic E-RFNF by the syntactic E-BCNF. Extending the relational case, schemata in E-BCNF guarantee at application design time that there will never be an instance that contains any E-redundant data value occurrence.

Theorem 5 *For all relation schemata R, all attribute subsets $E \subseteq R$, and all sets Σ over R, R is in E-RFNF for Σ if and only if R is in E-BCNF for Σ.*

We can apply the characterization to our example.

Example 8 *For $R = cdflpz$ and $\Sigma = \{p : cfl, p : dz \rightarrow c\}$, R is in cflp-RFNF for Σ since R is in BCNF for $\Sigma[E] = \{cfl \rightarrow cflp\}$.*

5.4 Efficient Testing

Due to the cover-insensitivity of the E-BCNF condition, one may wonder about the efficiency of checking whether a given schema is in E-BCNF for a given set Σ. As in the relational case it suffices to check some eFDs in Σ instead of checking all eFDs in $\Sigma^+_{\mathfrak{C}}$.

Theorem 6 *A relation schema R is in E-BCNF for Σ if and only if for every eFD $E' : X \to Y \in \Sigma$ where $E' \subseteq E$ and $Y \nsubseteq X$, $E : X \in \Sigma^+_{\mathfrak{C}}$. Hence, deciding if a schema is in E-BCNF for Σ is quadratic in Σ.*

We apply the simpler characterization to our example.

Example 9 *For $R = cdflpz$ and $\Sigma = \{p : cfl, p : dz \to c\}$, R is in $cflp$-BCNF for Σ since there is no eFD $E' : X \to Y \in \Sigma$ such that $E' \subseteq E = cflp$.*

5.5 E-3NF

We now introduce E-Third normal form (E-3NF) which ensures that all FDs can be enforced locally, without the need of joining relations to check for consistency of updates. Recall the 3NF condition [7]: R is in *3NF* for an FD set Σ if for every FD $X \to Y \in \Sigma^+_{\mathfrak{A}}$ where $Y \nsubseteq X$, $X \to R \in \Sigma^+_{\mathfrak{A}}$ or every attribute in $Y - X$ is prime. An attribute A is *prime* if it occurs in some minimal key of R for Σ. An attribute subset X of R is a *key* of R for Σ if $X \to R \in \Sigma^+_{\mathfrak{A}}$. A key X of R is *minimal* for Σ if every proper subset $Y \subset X$ is not a key of R for Σ. We extend these concepts to handle data completeness requirements. For $E \subseteq R$ and an eUC/eFD set Σ, an eUC $E : K \in \Sigma^+_{\mathfrak{C}}$ is E-minimal for Σ if and only if there is no E-key $E : K' \in \Sigma^+_{\mathfrak{C}}$ for Σ such that $K' \subset K$. An attribute A is *E-prime* for Σ if and only if $A \in K$ for some E-minimal key $E' : K \in \Sigma^+_{\mathfrak{C}}$.

Definition 5 *A relation schema R is in E-3NF for Σ if and only if for every non-trivial eFD $E : X \to Y \in \Sigma^+_{\mathfrak{C}}$ with $E' \subseteq E$, $E : X \in \Sigma^+_{\mathfrak{C}}$ or every attribute in $Y - X$ is E-prime.*

We can check this condition on our running example.

Example 10 *For $R = cdflpz$ and $\Sigma = \{p : cfl, p : dz \to c\}$, we have seen that $c : dflpz$ is an R-minimal key for Σ, the other R-minimal key being $dpz : cfl$. That is, every attribute in R is R-prime. Hence, R is in R-3NF. However, R is not in cdpz-3NF as eFD $p : dz \to c \in \Sigma$, $c \notin \{d, z\}$, $cp : dz \notin \Sigma^+_{\mathfrak{C}}$, and c is not cdpz-prime.*

Similar to E-BCNF and BCNF, we can check that R is in E-3NF for Σ by testing that R is in 3NF for $\Sigma[E]$.

Theorem 7 *For all relation schemata R, all $E \subseteq R$, and all sets Σ over R, R is in E-3NF for Σ if and only if R is in 3NF for $\Sigma[E]$.*

Example 11 *In our running example $R = cdflpz$, $\Sigma = \{p : cfl, p : dz \to c\}$ and $E = R$, $\Sigma[E] = \{cfl \to dpz, dz \to c\}$. The two minimal keys are cfl and $dflpz$. As every attribute is prime, R is in 3NF for $\Sigma[E]$.*

Finally, E-3NF can be validated by checking the relevant conditions for just the input Σ, rather than its closure $\Sigma^+_{\mathfrak{C}}$.

Theorem 8 *R is in E-3NF for a set Σ of eUCs and eFDs over R if and only if for every eFD $E' : X \to Y \in \Sigma$ where $E' \subseteq E$ and $Y \nsubseteq X$, $E : X \in \Sigma^+_{\mathfrak{C}}$ or every attribute in $Y - X$ is E-prime.*

This translates to our running example as follows.

Example 12 *For $R = cdflpz$, $\Sigma = \{p : cfl, p : dz \to c\}$, $E = R$ and $p : dz \to c \in \Sigma$, $c \in R$ is E-prime. By Theorem 8 this suffices to establish that R is in E-3NF for Σ.*

5.6 Hardness of Normal Form Criteria

As relational normalization occurs as the special case where $E = R$, checking normal form criteria is hard in general.

Theorem 9 *Deciding whether a sub-schema of a given schema is in E-BCNF for a given set Σ is* coNP*-complete. Deciding whether a given schema is in E-3NF for a given set Σ is* NP*-complete.*

6 Tailoring Normalization

We now establish algorithms to design relational database schemata that are tailored to the completeness requirements of applications. For that purpose, we normalize a given schema R for the given set Σ of eUCs and eFDs. The completeness requirements are consolidated in an attribute subset $E \subseteq R$, expressing that the application only handles E-complete records. The choice of E determines the set $\Sigma[E]$ of traditional FDs that are used to normalize R. For each E we pursue i) lossless BCNF decompositions that are redundancy-free but potentially not dependency-preserving, and ii) lossless 3NF syntheses that are dependency-preserving but potentially not redundancy-free.

6.1 E-BCNF Decomposition

We recall terminology from relational databases. A *decomposition* of relation schema R is a set $\mathcal{D} = \{R_1, \dots, R_n\}$ of relation schemata such that $R_1 \cup \cdots \cup R_n = R$. For $R_j \subseteq R$ and FD set Σ over R, $\Sigma_{R_j} = \{X \to Y \mid X \to Y \in \Sigma^+_{\mathfrak{A}} \text{ and } X, Y \subseteq R_j\}$ denotes the *projection* of Σ onto R_j. A decomposition $\mathcal{D}$ of R with FD set Σ is *lossless* if every relation r over R that satisfies Σ is the join of its projections on the elements of $\mathcal{D}$, that is, $r = \bowtie_{R_j \in \mathcal{D}} r[R_j]$. Here, $r[R_j] = \{t(R_j) \mid t \in r\}$. A *BCNF* decomposition of R with FD set Σ is a decomposition $\mathcal{D}$ of R where every $R_j \in \mathcal{D}$ is in BCNF for Σ_{R_j}. Theorem 4 motivates a definition of an E-lossless BCNF decomposition.

Definition 6 *An* E-lossless BCNF decomposition *of a schema R for a set Σ of eUCs and eFDs over R is a lossless BCNF decomposition of R for $\Sigma[E]$.*

Instrumental to Definition 6 is the following decomposition theorem. It covers the classical decomposition theorem [27] as the special case where $E = R$. Data completeness-tailored normalization does not loose any records by following a hybrid decomposition approach. Given E, a relation is decomposed horizontally into its application-relevant part r^E of E-complete records, and its application-irrelevant part $r - r^E$ of records with missing data on E. Classical vertical decomposition can then be applied to r^E.

Theorem 10 *Let $E : X \to Y$ be an eFD that satisfies the relation r over relation schema R. Then the set of E-complete records of r is the lossless join of its projections on XY and $X(R - Y)$, that is, $r^E = r^E[XY] \bowtie r^E[X(R - Y)]$. Also, r is the disjoint union of the set of E-complete records of r, and the set of records of r with missing data on some column in E, that is, $r = r^E \uplus (r - r^E)$.*

Hence, an E-lossless BCNF decomposition for a set Σ of eUCs and eFDs can simply be obtained by a classical lossless BCNF decomposition for the (E, FD)-reduct $\Sigma[E]$ of Σ.

PROBLEM:	E-BCNF Decomposition
INPUT:	Relation Schema R Set Σ of eUCs and eFDs over R Attribute subset $E \subseteq R$
OUTPUT:	E-lossless BCNF decomposition of R for Σ
METHOD:	Perform a lossless BCNF decomposition of R for $\Sigma[E]$

We illustrate the decomposition on our running example.

Example 13 *In our running example $R = cdflpz$, $\Sigma = \{p : cfl, p : dz \to c\}$, and $E = R$, R is not in E-BCNF for Σ. In fact, R is not in BCNF for $\Sigma[E] = \{cfl \to dpz, dz \to c\}$. A BCNF decomposition yields $R_1 = cdz$ with $\Sigma_1 = \{dz \to c\}$ and $R_2 = dflpz$ with $\Sigma_2 = \emptyset$. For the relation r from Table 3, the projection of r^E on the decomposed schema is as follows, except for the last row in both tables because tuple t_{15} has a null marker occurrences on column* f_name.

c(ity)	z(ip)	d(ate_register)
burlington	*27217*	*05/11/1940*
jackson	*27845*	*06/15/1936*
lasker	*27845*	*05/10/1940*
wilmington	*28405*	*01/01/1930*
rose hill	*28458*	*01/01/1930*
new bern	*28562*	*03/06/1935*
chinquapin	*28521*	*01/01/1936*
burlington	*27215*	*05/06/1940*
mebane	*27302*	*4/05/1940*
mebane	*27302*	*05/06/1940*
kinston	*28504*	*05/02/1940*
kinston	*28504*	*05/04/1940*

f(_name)	l(_name)	z(ip)	p(hone)	d(ate_register)
ida	*cheek*	*27217*	*226 4848*	*05/11/1940*
effie	*massey*	*27217*	*336 226 8544*	*05/11/1940*
peggy	*floyd*	*27845*	*252 536 2668*	*06/15/1936*
essie	*warren*	*27845*	*252 539 2866*	*05/10/1940*
rodney	*glockner*	*28405*	*910 509 3864*	*01/01/1930*
sallie	*blanchard*	*28458*	*910 289 3320*	*01/01/1930*
joseph	*cox*	*28562*	*252 288 4763*	*03/06/1935*
james	*smith*	*28521*	*910 285 3493*	*01/01/1936*
james	*smith*	*27215*	*584 4202*	*05/06/1940*
dorothy	*faucette*	*27302*	*919 563 1285*	*4/05/1940*
dorothy	*allred*	*27302*	*563 1426*	*05/06/1940*
eloise	*croom*	*28504*	*252 569 4436*	*05/02/1940*
$\perp$	*croom*	*28504*	*252 569 9516*	*05/04/1940*

All E-redundant data value occurrences from r have been eliminated. However, the eUC $p:cfl$ was not preserved.

Recall that a decomposition $\mathcal{D}$ of schema R with FD set Σ is *dependency-preserving* whenever $\Sigma^+_{\mathfrak{A}} = (\cup_{R_j\in\mathcal{D}}\Sigma[R_j])^+_{\mathfrak{A}}$.

Definition 7 *An* E-dependency-preserving *decomposition of a schema R for the eUC/eFD set Σ is a dependency-preserving decomposition of R for $\Sigma[E]$.*

6.2 E-3NF Synthesis

3NF synthesis guarantees dependency-preservation, but may exhibit data value redundancy caused by FDs. It was shown recently that 3NF exhibits minimal levels of data redundancy when achieving dependency-preservation [1, 19]. Hence, we will equip our new framework with an appropriate 3NF synthesis strategy. Recall that a 3NF decomposition of a relation schema R for an FD set Σ is a decomposition $\mathcal{D}$ of R where every $R_j \in \mathcal{D}$ is in 3NF for Σ_{R_j}. Theorem 7 motivates the following definition.

Definition 8 *An* E-dependency-preserving, E-lossless 3NF decomposition *of a schema R for the set Σ of eUCs and eFDs is a dependency-preserving, lossless 3NF decomposition of R for $\Sigma[E]$.*

Following Theorem 10, an E-dependency-preserving, E-lossless 3NF synthesis for a set Σ of eUCs and eFDs can simply be obtained by a classical dependency-preserving lossless 3NF synthesis for the (E, FD)-reduct $\Sigma[E]$ of Σ.

PROBLEM:	E-3NF Synthesis
INPUT:	Relation schema R Set Σ of eUCs and eFDs over R Attribute subset $E \subseteq R$
OUTPUT:	E-dependency-preserving, E-lossless 3NF decomposition of R wrt Σ
METHOD:	Perform a dependency-preserving, lossless 3NF synthesis of R for $\Sigma[E]$

We illustrate the synthesis on our running example.

Example 14 *In our running example* $R = cdflpz$, $\Sigma = \{p : cfl, p : dz \to c\}$, *$R$ is indeed in R-3NF for* Σ. *For* $E = cdpz$, *however, R is not in E-3NF for* Σ. *In fact, R is not in 3NF for* $\Sigma[E] = \{dz \to c\}$. *A 3NF synthesis yields* $R_1 = cdz$ *with* $\Sigma_1 = \{dz \to c\}$ *and to ensure E-losslessness we add the E-minimal key* $R_2 = dflpz$ *with* $\Sigma_2 = \emptyset$. *For the relation r from Table 3, the projection of* r^E *onto the decomposed schema is the same as in Example 13 but inclusive of the last row in both tables because of the looser data-completeness requirements.*

Summary. We have tailored the entire relational schema design framework to data-completeness requirements of applications. We allow data stewards to declare these requirements as an extension to the familiar concept of a functional dependency. The results show that extensions of the familiar BCNF (3NF) normal form achieve an elimination (minimization across dependency-preserving decompositions) of data values that may occur redundantly in records that meet the completeness requirements. As an optimal result for database practice, schemata can be automatically transformed into these normal forms by applying relational normalization algorithms to a set of relational FDs that emerge from the set of extended FDs and the data-completeness requirements. The next section illustrates what our framework achieves when applied to real-world schemata and data.

7 Experiments

We report on experiments with our framework using real-world benchmark schemata and data, available for download here[2]. We provide insight on how many E-redundant data values occur in the data, rank the relevance of our eFDs by how many data redundancies they cause, show how often schemata satisfy a normal form condition, how much redundancy E-3NF permits, how many dependencies E-BCNF preserves, and how large decomposed schemata become. We consider the times of computations, and suggest how data stewards can use our ranking of eFDs, using the example of our two applications from the introduction. All our experiments are run on an Intel Xeon 3.6 GHz, 256GB RAM, Windows 10 Dell workstation.

7.1 E-redundancy and eFD Ranking

Table 6 lists for each incomplete benchmark data set the number *#complete* of data occurrences that are complete, the number *#red* of those that are redundant, the percentage *%red* of redundant data value occurrences in the data set, and the *time* in seconds to compute all the redundant occurrences given the data set and given the canonical cover of the eUCs and eFDs that hold on the data set. The canonical covers can be computed by some algorithms that will be discussed in a separate article. The sheer number and percentage of redundant occurrences clearly motivates our research, and the time taken

---

[2] `https://bit.ly/2AoOis6`

data set	*#complete*	*#red*	*%red*	*time (s)*
horse	6,795	4,775	70.27	8.075
bridges	1,327	411	30.97	0.002
hepatitis	2,933	1,695	57.79	0.179
breast	7,585	712	9.39	0.005
echo	1,584	489	30.87	0.002
plista	39,431	28,827	73.11	18.415
flight	57,062	48,414	84.84	86.585
ncvoter	16,137	3,170	19.64	0.047
china	4,313,980	2,131,677	49.41	412.867
uniprot	11,600,704	1,413,038	12.18	1,777.245
diabetic	1,017,738	543,935	53.45	3,273.183

Table 6: Redundant data value occurrences in benchmarks, and the time in seconds to compute all of them

to compute them shows that this insightful analysis is efficient on even large data sets with large numbers of constraints. Of course, if a team of domain experts selects the meaningful eUCs and eFDs for an application, then the redundant occurrences will likely be fewer and can be computed more efficiently.

Guiding data stewards in their selection of meaningful eFDs from the canonical cover, we can rank the relevance of an eFD by the number of redundant data value occurrences it causes. Figure 1 shows the number of eFDs in a canonical cover that cause not more than a given number of redundant data value occurrences in the data set. The labels on the x-axis indicate the maximum values for 0, 2.5, 5, 10, 15, 20, 40, 60, 80, and 100 percent of the maximum redundant occurrences any eFD causes. The figure illustrates clearly that most eFDs cause few data redundancies, which makes it possible for data stewards to focus their attention to select few eFDs of higher rank.

7.2 Pure eFDs

As indicated in our introduction, pure eFDs occur frequently in real-world data, cause many redundant data value occurrences, and also occur frequently among those eFDs that cause most redundant data value occurrences. This is illustrated on our benchmark data sets in Table 7. Here, we list the percentage of pure eFDs among all eFDs in the canonical covers we computed, the average loss of redundant data value occurrences in percent when transforming a pure eFD $E : X \rightarrow Y$ into a non-pure eFD $\emptyset : E - Y \rightarrow Y$, and the percentage of pure eFDs among those eFDs that ranked within the top-10% according to the number of redundant data values they cause.

7.3 Quality of Decompositions

For the following experiments we created inputs as outlined next. For each fixed size $|E|$ of the completeness requirements, we created different sets of eUCs and eFDs by picking up to 1,000 unique attribute sets E of the fixed size, and then selecting all eUCs and

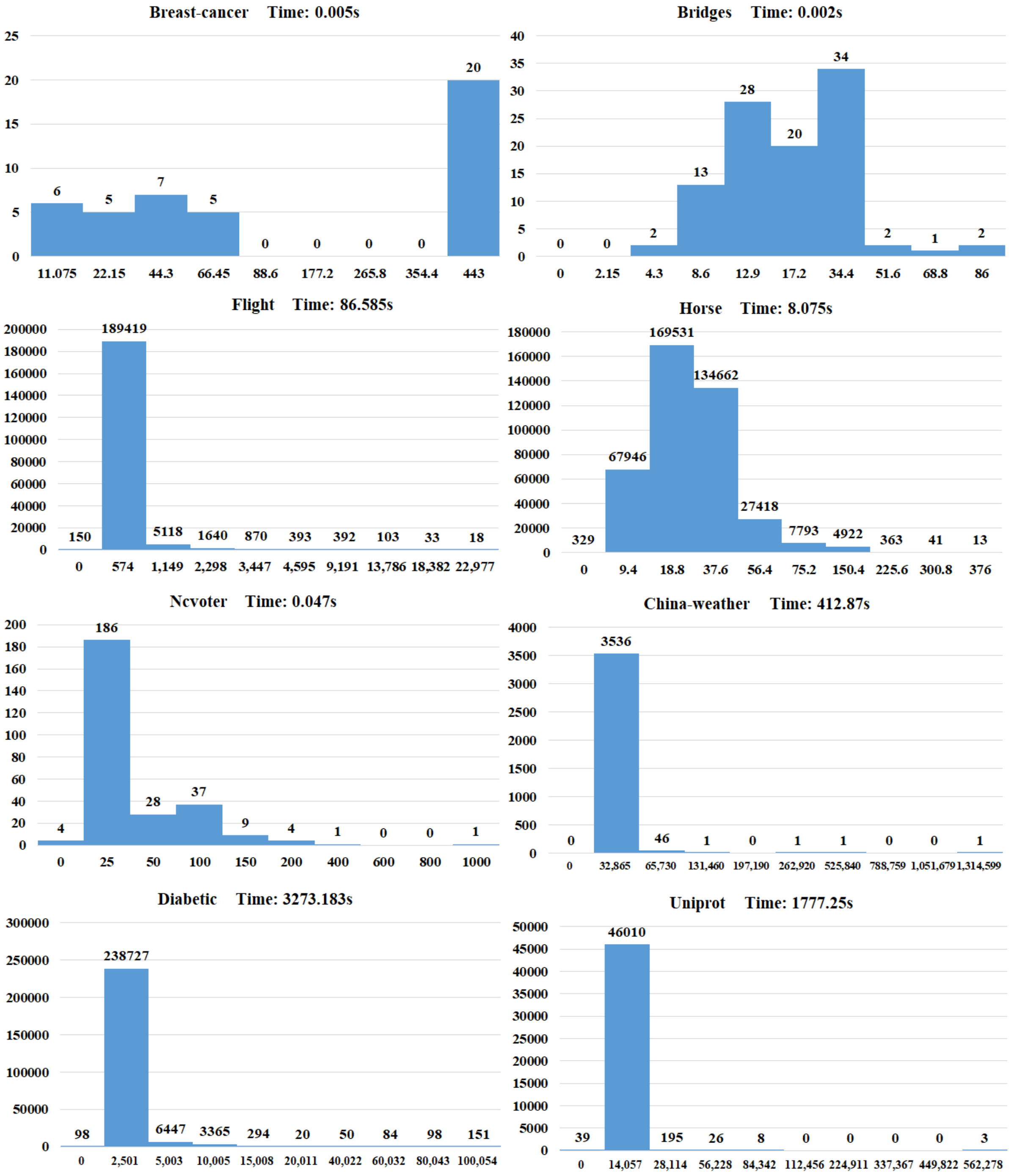


Figure 1: Numbers of eFDs in canonical covers (y-axis) that cause not more than the given number of redundant occurrences (x-axis)

data set	%pure	red loss	%pure top-10%
breast	0	0	0
bridges	29.58	62.85	69.23
china	18.83	81.32	60.08
diabetic	54.02	57.79	64.68
echo	20.32	65.24	46.67
flight	17.73	44.50	16.46
hepatitis	53.21	77.56	67.62
horse	94.25	20.60	94.38
ncvoter	16.35	66.88	50.00
plista	74.18	8.14	69.92
uniprot_512k_30c	50.73	63.50	92.55
ncvoter128k	34.00	45.80	45.89
ncvoter256k	39.88	50.14	58.86
ncvoter512k	40.30	46.57	44.13
ncvoter1024k	39.46	49.20	48.54

Table 7: Statistics on pure eFDs in benchmark data

eFDs that hold on the data set and whose embeddings are subsets of E. Figure 2 shows for each data set and each size of E, the percentage of all input sets that are in E-BCNF and E-3NF, respectively.

Figure 3 shows the average percentages of i) the E-complete data value occurrences that are redundant (blue line), ii) those after E-3NF decomposition (orange line) and iii) eliminated redundancies after E-3NF synthesis (yellow line) all plotted against the LHS vertical axis, and iv) dependencies preserved during E-BCNF decomposition (red dotted line) plotted against the RHS vertical axis. In general, there is no control about the number of E-redundant data values that an E-3NF decomposition must tolerate to preserve all relevant dependencies. Vice versa, there is no control on how many relevant dependencies will be lost to eliminate all E-redundant data values during E-BCNF decomposition. For instance, E-3NF may duplicate non-trivial eFDs across schemata, causing a blow-up of the E-redundancies (orange above blue line), see *diabetic* and *china_weather*.

7.4 Size and time of decompositions

Figure 4 illustrates the impact of the size of E on the cardinality of the decompositions, that is, their total number of attributes (LHS y-axis) and the computation time in seconds (RHS y-axis). Boldly speaking, the larger the decompositions the more updates (less redundancy) and the less queries (more joins required) will be efficient.

7.5 Qualitative analysis

For qualitative insight, we consider two applications for the data set *ncvoter* with 19 columns and 1000 records. The first application has attribute set E_1 with *full_phone_num*,

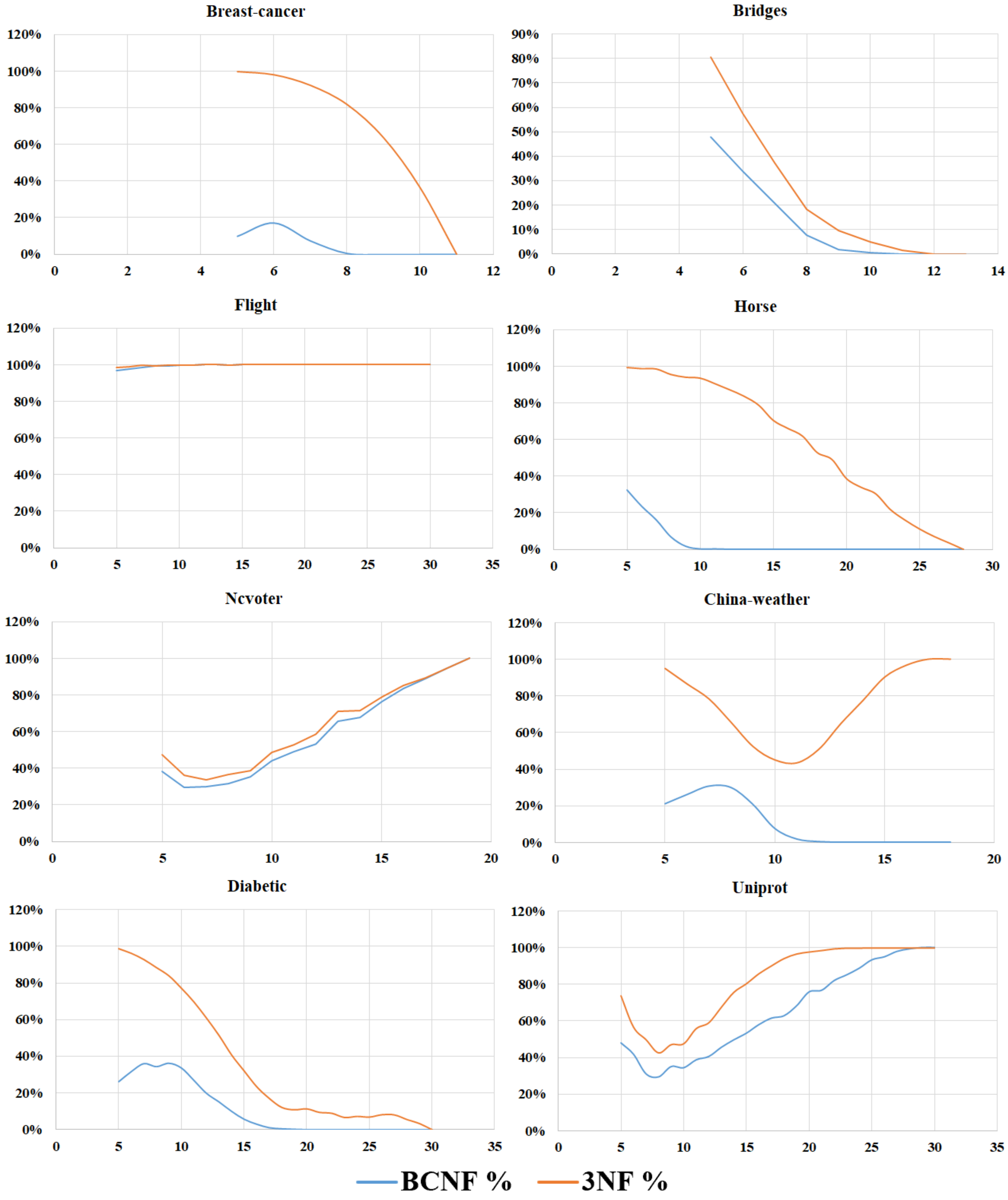


Figure 2: Average percentage of schemata in E-3NF and E-BCNF, respectively, by given size of E

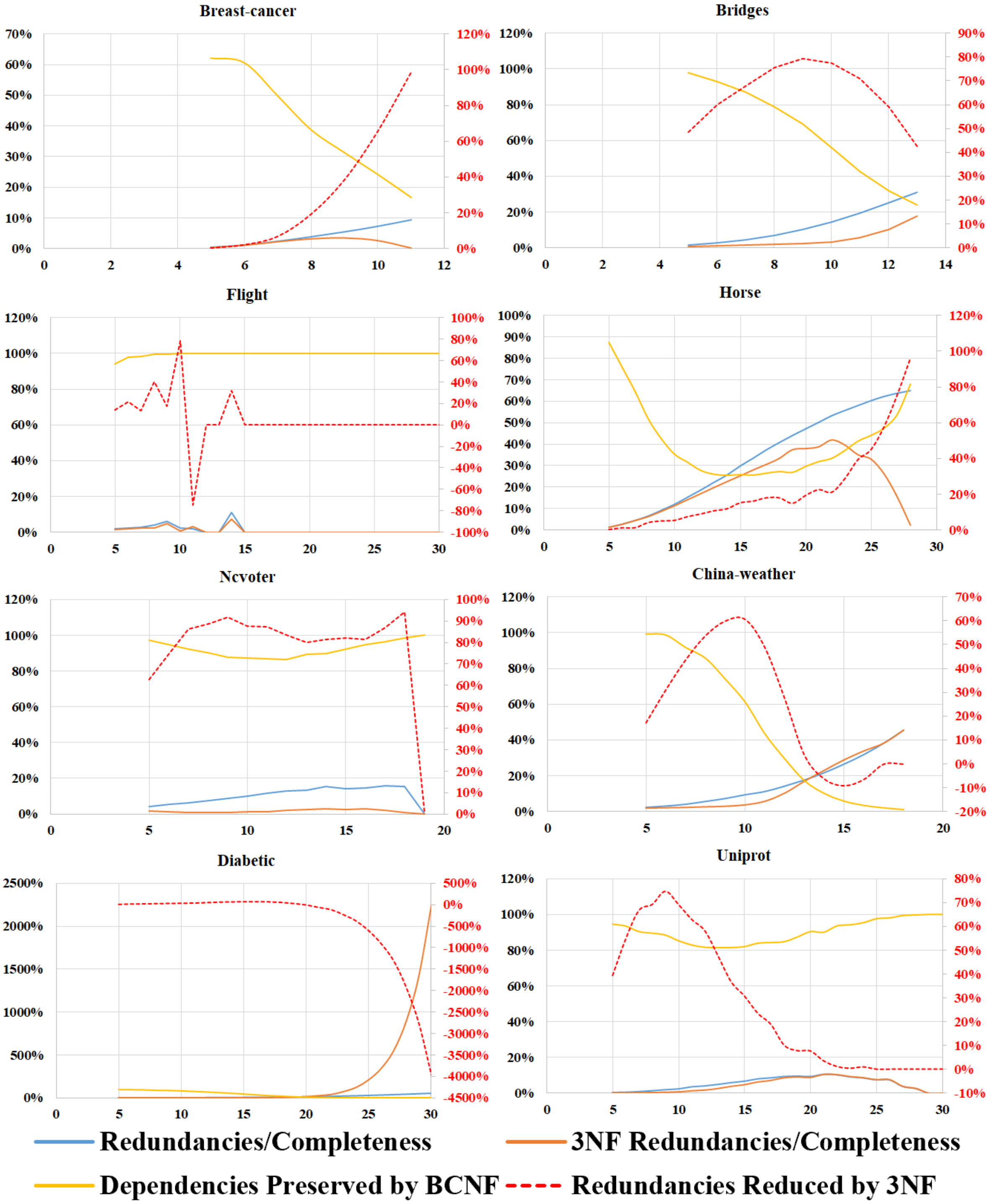


Figure 3: Elimination of E-redundancy by E-3NF (vertical LHS), and E-preservation by E-BCNF (vertical RHS)

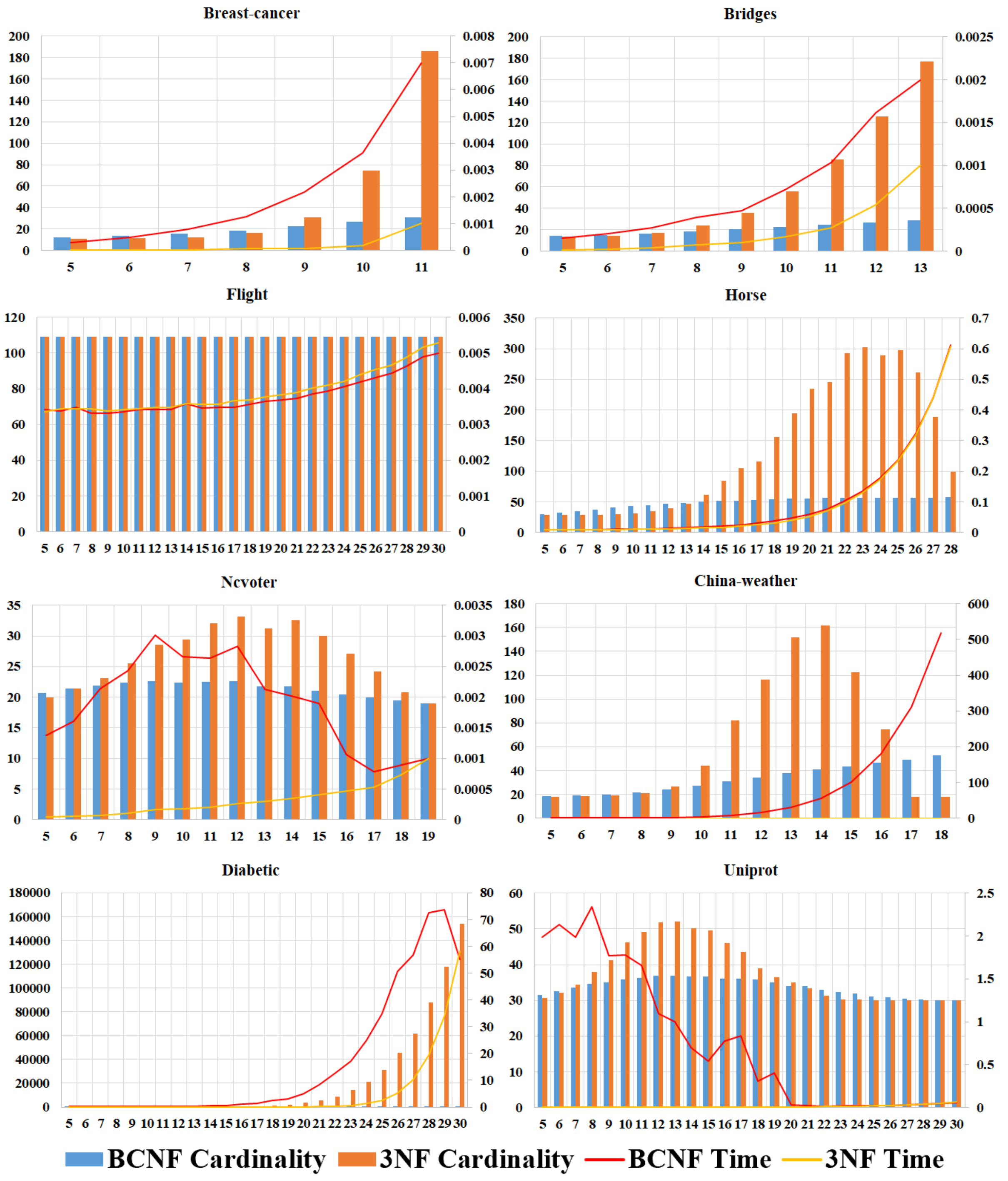


Figure 4: Average cardinality of decompositions and time (s) taken to compute them

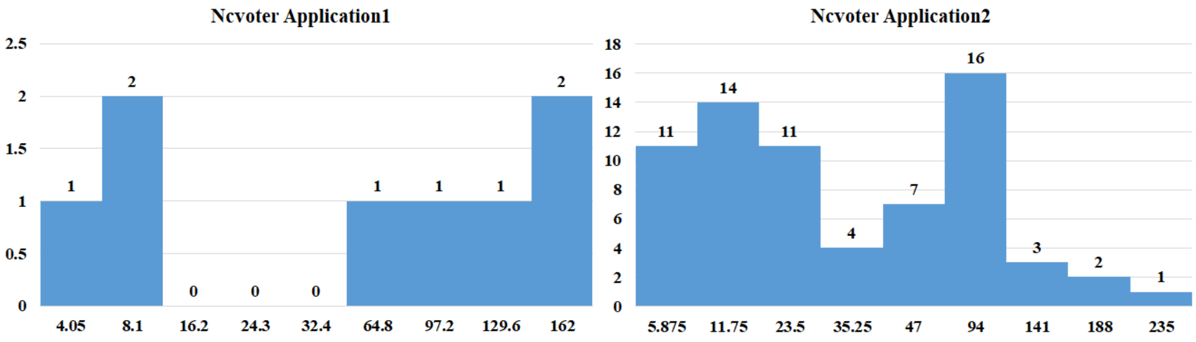
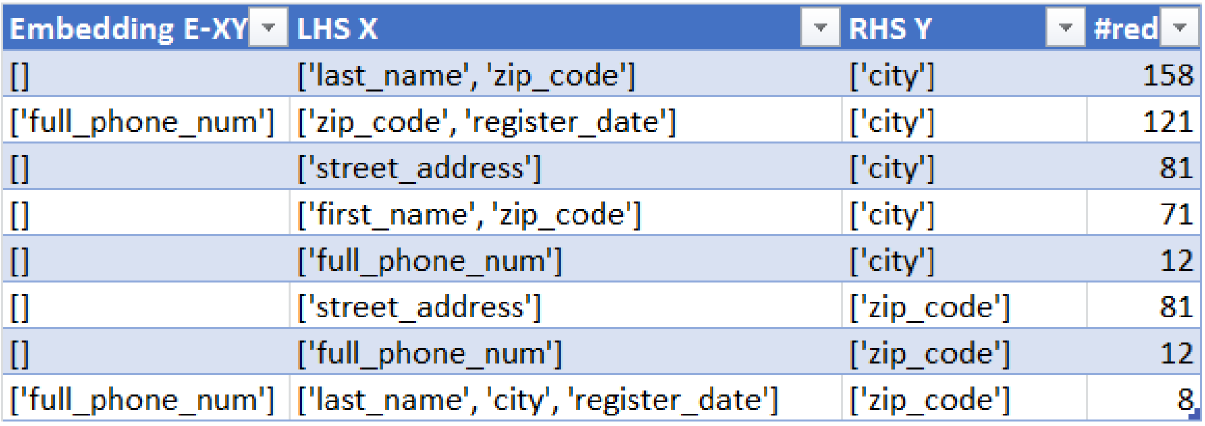


Figure 5: NCVoter applications



Embedding E-XY	LHS X	RHS Y	#red
[]	['last_name', 'zip_code']	['city']	158
['full_phone_num']	['zip_code', 'register_date']	['city']	121
[]	['street_address']	['city']	81
[]	['first_name', 'zip_code']	['city']	71
[]	['full_phone_num']	['city']	12
[]	['street_address']	['zip_code']	81
[]	['full_phone_num']	['zip_code']	12
['full_phone_num']	['last_name', 'city', 'register_date']	['zip_code']	8

Figure 6: Ranking of some eFDs for application E_1

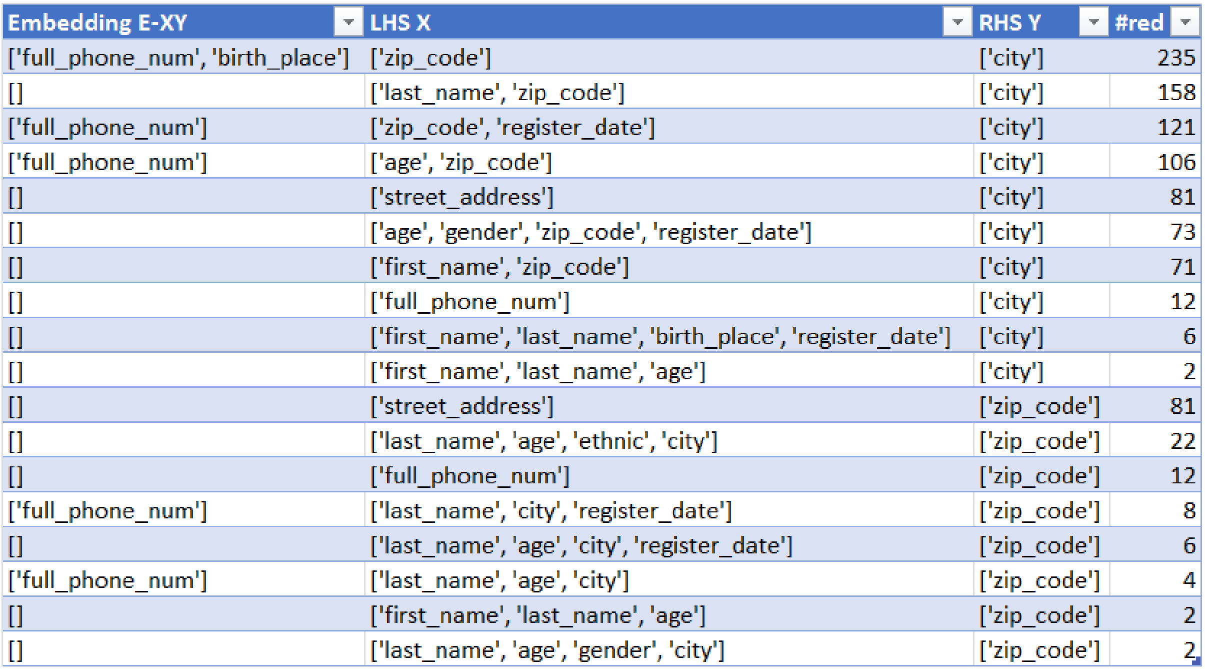
street_address, *register_date*, *last_name*, *first_name*, *city*, and *zip_code*, while the second application uses the attribute set $E_2 \supseteq E_1$ plus *birthplace*, *ethnic*, *race*, *gender*, and *age*. From the canonical cover we then selected only those eFDs $E : X \to Y$ where E contained E_1 or E_2, respectively.

Our rankings help identify eFDs relevant for normalization and pinpointing dirty data. Figure 5 shows the distribution of eFDs in percentiles of the redundant values they cause based on E_1 and E_2, respectively. For growing $|E|$ typically more eFDs need consideration. Here, our ranking offers a convenient measure of relevance for data stewards.

A view that might be particularly useful for data stewards is to fix a column in E, and list the minimal LHSs that functionally determine that column, ranked by the relevance of the corresponding eFD. This is illustrated in Figures 6 and 7, where we list all minimal LHSs for the columns *city* and *zip_code* based on the two completeness requirements E_1 and E_2, respectively.

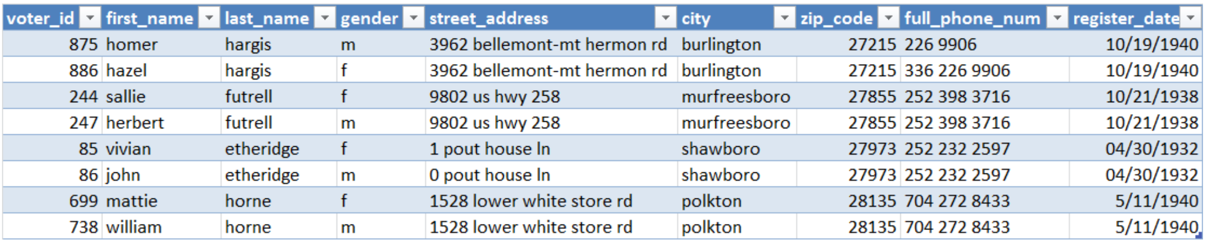
Finally, a data steward can view the records in which the redundant data values actually occur. This helps them decide if the eFD is meaningful for the application, holds just accidentally, or identify records with dirty data. Figure 8 shows some records with E_1-redundant data values.

An inspection of these records reveals some dirty data: i) Hazel and Homer Hargis



Embedding E-XY	LHS X	RHS Y	#red
['full_phone_num', 'birth_place']	['zip_code']	['city']	235
[]	['last_name', 'zip_code']	['city']	158
['full_phone_num']	['zip_code', 'register_date']	['city']	121
['full_phone_num']	['age', 'zip_code']	['city']	106
[]	['street_address']	['city']	81
[]	['age', 'gender', 'zip_code', 'register_date']	['city']	73
[]	['first_name', 'zip_code']	['city']	71
[]	['full_phone_num']	['city']	12
[]	['first_name', 'last_name', 'birth_place', 'register_date']	['city']	6
[]	['first_name', 'last_name', 'age']	['city']	2
[]	['street_address']	['zip_code']	81
[]	['last_name', 'age', 'ethnic', 'city']	['zip_code']	22
[]	['full_phone_num']	['zip_code']	12
['full_phone_num']	['last_name', 'city', 'register_date']	['zip_code']	8
[]	['last_name', 'age', 'city', 'register_date']	['zip_code']	6
['full_phone_num']	['last_name', 'age', 'city']	['zip_code']	4
[]	['first_name', 'last_name', 'age']	['zip_code']	2
[]	['last_name', 'age', 'gender', 'city']	['zip_code']	2

Figure 7: Ranking of some eFDs for application E_2



voter_id	first_name	last_name	gender	street_address	city	zip_code	full_phone_num	register_date
875	homer	hargis	m	3962 bellemont-mt hermon rd	burlington	27215	226 9906	10/19/1940
886	hazel	hargis	f	3962 bellemont-mt hermon rd	burlington	27215	336 226 9906	10/19/1940
244	sallie	futrell	f	9802 us hwy 258	murfreesboro	27855	252 398 3716	10/21/1938
247	herbert	futrell	m	9802 us hwy 258	murfreesboro	27855	252 398 3716	10/21/1938
85	vivian	etheridge	f	1 pout house ln	shawboro	27973	252 232 2597	04/30/1932
86	john	etheridge	m	0 pout house ln	shawboro	27973	252 232 2597	04/30/1932
699	mattie	horne	f	1528 lower white store rd	polkton	28135	704 272 8433	5/11/1940
738	william	horne	m	1528 lower white store rd	polkton	28135	704 272 8433	5/11/1940

Figure 8: E_1-redundancies caused by eFD $full\_phone\_num : last\_name, city, register\_date \rightarrow zip\_code$

live at the same street address, and their phone numbers are different, and ii) Vivian and John Etheridge share the same phone number, but their street address is different. For i) the inconsistency can easily be resolved by giving the full phone number, while for ii) it is more likely that Vivian indicated the correct street number.

Summary. Our experiments illustrate on benchmark schemata and data that eFDs provide effective declarative means to capture and reason about redundant data value occurrences that are fit for application requirements. Our ranking guides data stewards in their selection of eFDs that are relevant for normalization purposes given the application requirements. Our normalization strategies result in a wide spectrum of schemata with clear achievements in terms of the elimination of pertinent data redundancies or the preservation of pertinent dependencies, accommodating tradeoffs between update and query efficiency. More experiments on perfect decompositions and an analysis of our experiments on horizontal fragments on *ncvoter* are available in the appendix.

8 Conclusion and Future Work

Schema design for data with missing values has been an open problem since the 1980s. Previous work has focused on finding suitable extensions of functional dependencies to accommodate different interpretations of null markers. In contrast, we introduced the class of embedded functional dependencies (eFDs) that is only dependant on complete data, can express completeness and integrity requirements of applications, and captures many redundant data values. This has enabled us to establish a fully-fledged normalization framework that is robust under different interpretations of null markers, tailors relational schema design to data-completeness requirements, and generalizes the achievements of classical BCNF and 3NF to applications with missing data. Extensive experiments on real-world benchmark schemata and data exemplify the effectiveness of our framework, the efficiency of our algorithms, and the achievements of our new normal form proposals. In particular, we illustrated the impact of the completeness requirements on trade-offs between data redundancy elimination and dependency-preservation. Next steps include an in-depth investigation into the discovery problem of embedded uniqueness constraints and functional dependencies from given relations. The discovery is important for data profiling and its applications, but can also help identify business rules. For that purpose, the ranking of eFDs can provide effective guidance. Furthermore, an extension of our framework to other data quality dimensions and other classes of data dependencies is important. The ability to declare data accuracy or data timeliness requirements as part of functional and multivalued dependencies would lift important data quality dimensions to first-class citizens that impact schema design considerations based on rich sources of data redundancy.

References

[1] M. Arenas. Normalization theory for XML. *SIGMOD Record*, 35(4):57–64, 2006.

[2] W. W. Armstrong. Dependency structures of data base relationships. In *IFIP Congress*, pages 580–583, 1974.

[3] P. Atzeni and N. M. Morfuni. Functional dependencies and constraints on null values in database relations. *Information and Control*, 70(1):1–31, 1986.

[4] P. A. Bernstein. Synthesizing third normal form relations from functional dependencies. *ACM TODS*, 1(4):277–298, 1976.

[5] J. Biskup. Achievements of relational database schema design theory revisited. In *Semantics in Databases*, pages 29–54, 1995.

[6] J. Biskup. Achievements of relational database schema design theory revisited. In L. Libkin and B. Thalheim, editors, *Semantics in Databases*, volume 1358 of *Lecture Notes in Computer Science*, pages 29–54. Springer, 1998.

[7] J. Biskup, U. Dayal, and P. A. Bernstein. Synthesizing independent database schemas. In *SIGMOD*, pages 143–151, 1979.

[8] E. F. Codd. Further normalization of the database relational model. In *Courant Computer Science Symposia 6: Data Base Systems*, pages 33–64, 1972.

[9] C. J. Date and R. Fagin. Simple conditions for guaranteeing higher normal forms in relational databases. *ACM Trans. Database Syst.*, 17(3):465–476, 1992.

[10] W. F. Dowling and J. H. Gallier. Linear-time algorithms for testing the satisfiability of propositional horn formulae. *J. Log. Program.*, 1(3):267–284, 1984.

[11] R. Fagin. The decomposition versus synthetic approach to relational database design. In *VLDB 1977*, pages 441–446, 1977.

[12] R. Fagin. Multivalued dependencies and a new normal form for relational databases. *ACM TODS*, 2(3):262–278, 1977.

[13] W. Fan, F. Geerts, X. Jia, and A. Kementsietsidis. Conditional functional dependencies for capturing data inconsistencies. *ACM TODS*, 33(2), 2008.

[14] S. Greco, C. Molinaro, and F. Spezzano. *Incomplete Data and Data Dependencies in Relational Databases.* Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2012.

[15] S. Hartmann and S. Link. The implication problem of data dependencies over SQL table definitions: Axiomatic, algorithmic and logical characterizations. *ACM Trans. Database Syst.*, 37(2):13:1–13:40, 2012.

[16] I. J. Heath. Unacceptable file operations in a relational data base. In *SIGFIDET Workshop*, pages 19–33, 1971.

[17] N. D. Jones and W. T. Laaser. Complete problems for deterministic polynomial time. *Theor. Comput. Sci.*, 3(1):105–117, 1976.

[18] H. Köhler and S. Link. SQL schema design: Foundations, normal forms, and normalization. In *SIGMOD*, pages 267–279, 2016.

[19] S. Kolahi. Dependency-preserving normalization of relational and XML data. *J. Comput. Syst. Sci.*, 73(4):636–647, 2007.

[20] S. Kolahi and L. Libkin. An information-theoretic analysis of worst-case redundancy in database design. *ACM Trans. Database Syst.*, 35(1):5:1–5:32, 2010.

[21] M. Levene and G. Loizou. Axiomatisation of functional dependencies in incomplete relations. *Theor. Comput. Sci.*, 206(1-2):283–300, 1998.

[22] M. Levene and G. Loizou. *A guided tour of relational databases and beyond.* Springer, 1999.

[23] M. Levene and M. W. Vincent. Justification for inclusion dependency normal form. *IEEE TKDE*, 12(2):281–291, 2000.

[24] D. Maier. *The Theory of Relational Databases.* Computer Science Press, 1983.

[25] T. Papenbrock and F. Naumann. Data-driven schema normalization. In *EDBT*, pages 342–353, 2017.

[26] R. Ramakrishnan and J. Gehrke. *Database management systems.* McGraw-Hill, 2003.

[27] J. Rissanen. Independent components of relations. *ACM TODS*, 2(4):317–325, 1977.

[28] M. W. Vincent. A corrected 5NF definition for relational database design. *Theor. Comput. Sci.*, 185(2):379–391, 1997.

[29] M. W. Vincent. Semantic foundations of 4NF in relational database design. *Acta Inf.*, 36(3):173–213, 1999.

[30] Z. Wei and S. Link. Embedded cardinality constraints. In *CAiSE*, pages 523–538, 2018.

[31] Z. Wei, S. Link, and J. Liu. Contextual keys. In *ER*, pages 266–279, 2017.

[32] C. Zaniolo. Database relations with null values. *J. Comput. Syst. Sci.*, 28(1):142–166, 1984.

A More experiments

We comment on some additional experiments we have conducted with the benchmark schemata and data. These include an analysis of obtaining perfect E-decompositions, as well as an analysis of the eFDs, E-redundancies, and E-normalization for the *ncvoter* benchmark with different numbers of rows.

A.1 Chances for a Perfect Decomposition

A decomposition is *perfect* when all its schemata are in E-RFNF and all relevant dependencies have been preserved. A perfect decomposition can be obtained by an E-3NF synthesis in which all schemata are actually in E-BCNF, or by an E-BCNF decomposition that is E-dependency-preserving. It is therefore an interesting question to ask what the chances are of obtaining a perfect decomposition by following either of these strategies. Figure 9 shows the outcome of this analysis as average percentages over all the experiments we run on our benchmarks with respect to different sizes of data completeness requirements.

A.2 E-Redundancies on *ncvoter* Fragments

It is interesting to apply our previous analyses to various horizontal fragments of the same data set. In this subsection we include such experiments on the *ncvoter* benchmark with numbers of rows varying from 2^i thousand records for $i = 0, ..., 10$. Figure 10 shows the ranking percentiles of eFDs discovered on those fragments, as well as the times in seconds to compute all the E-redundant data value occurrences.

A.3 E-Normalization on *ncvoter* Fragments

Figure 11 shows for the different horizontal fragments of *ncvoter*, the average percentages of i) the E-complete data value occurrences that are redundant (blue line), ii) those after E-3NF decomposition (orange line) and iii) eliminated redundancies after E-3NF synthesis (yellow line) all plotted against the LHS vertical axis, and iv) dependencies preserved during E-BCNF decomposition (red dotted line) plotted against the RHS vertical axis. It is very interesting to see how stable the results are across the vastly varying numbers of rows in the horizontal fragments.

A.4 Schema design quality

Finally, we quantitatively analyze the impact of the size of E on the average ratio of schemata in an E-3NF decomposition that are in E-BCNF, which we call the *BCNF-ratio*, and on the average ratio of dependencies that have been preserved in an E-BCNF decomposition, which we call the *Preservation*-ratio. It tells us about the chances of finding an optimum decomposition that is free from E-data redundancy and E-dependency-preserving using either E-BCNF decomposition or E-3NF synthesis.

A.5 Quality of Decompositions

For the following experiments we created inputs as outlined next. For each fixed size $|E|$ of the completeness requirements, we created different sets of eUCs and eFDs by picking up to 1,000 unique attribute sets E of the fixed size, and then selecting all eUCs and eFDs that hold on the data set and whose embeddings are subsets of E. Figure 2 shows

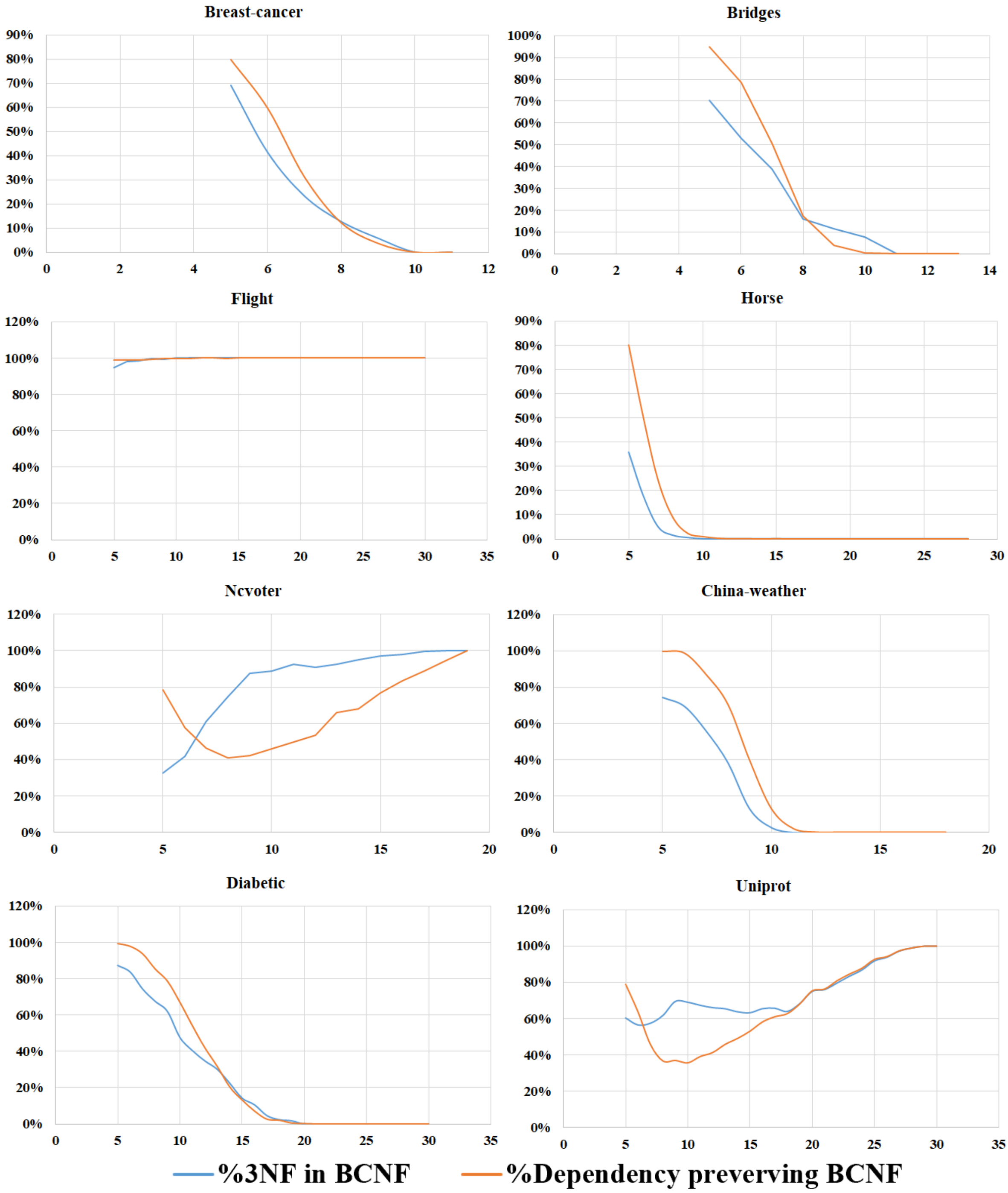
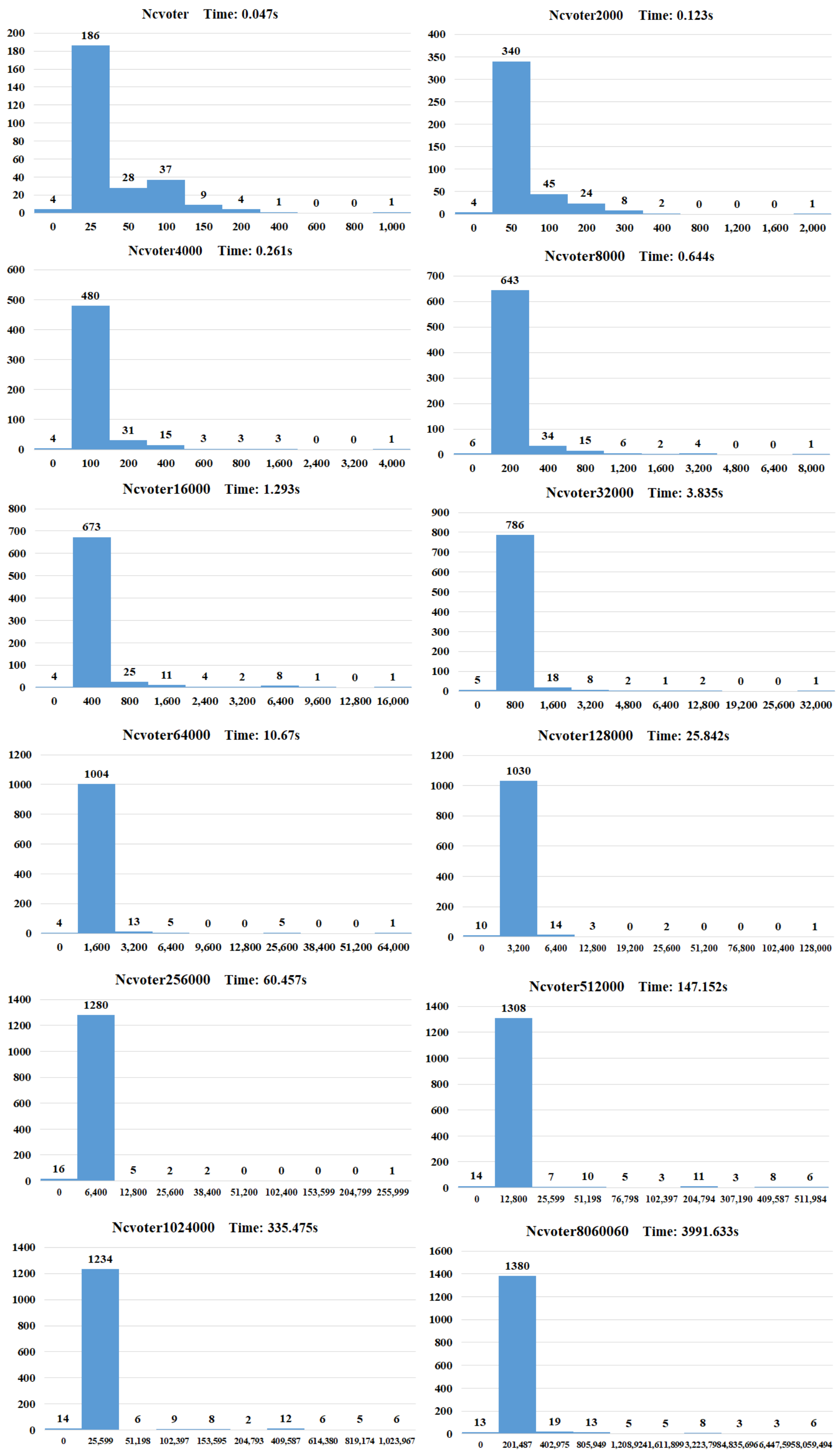


Figure 9: Average Percentages of E-3NF-decompositions in E-BCNF, and of E-BCNF decompositions that are E-dependency-preserving



Ncvoter Time: 0.047s

Ncvoter2000 Time: 0.123s

Ncvoter4000 Time: 0.261s

Ncvoter8000 Time: 0.644s

Ncvoter16000 Time: 1.293s

Ncvoter32000 Time: 3.835s

Ncvoter64000 Time: 10.67s

Ncvoter128000 Time: 25.842s

Ncvoter256000 Time: 60.457s

Ncvoter512000 Time: 147.152s

Ncvoter1024000 Time: 335.475s

Ncvoter8060060 Time: 3991.633s

Figure 10: Ranking percentiles of eFDs in canonical cover for different fragments of *ncvoter* and time to compute all redundancies in seconds

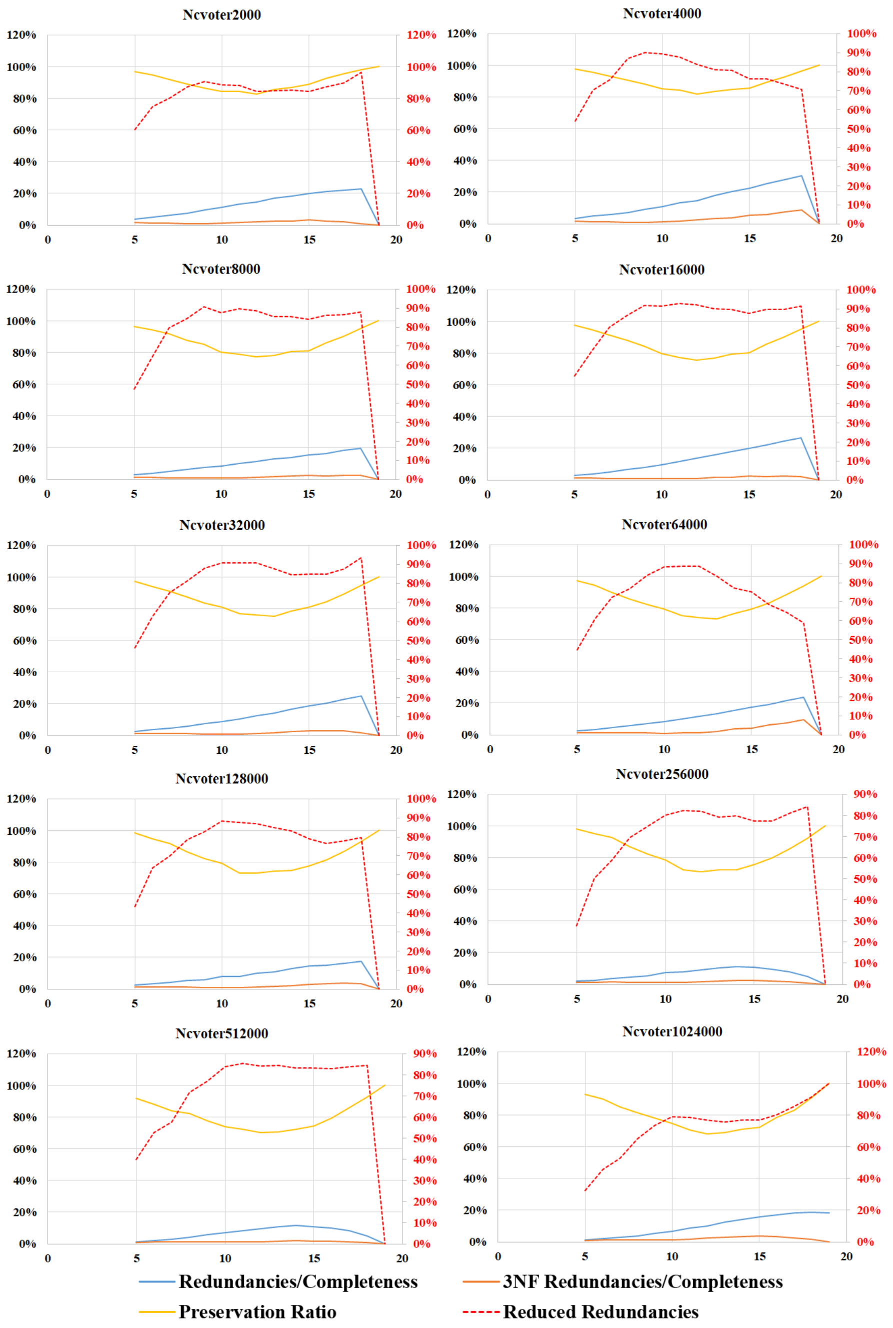


Figure 11: Elimination of E-redundancy by E-3NF (vertical LHS), and E-preservation by E-BCNF (vertical RHS) by horizontal fragments of *ncvoter*

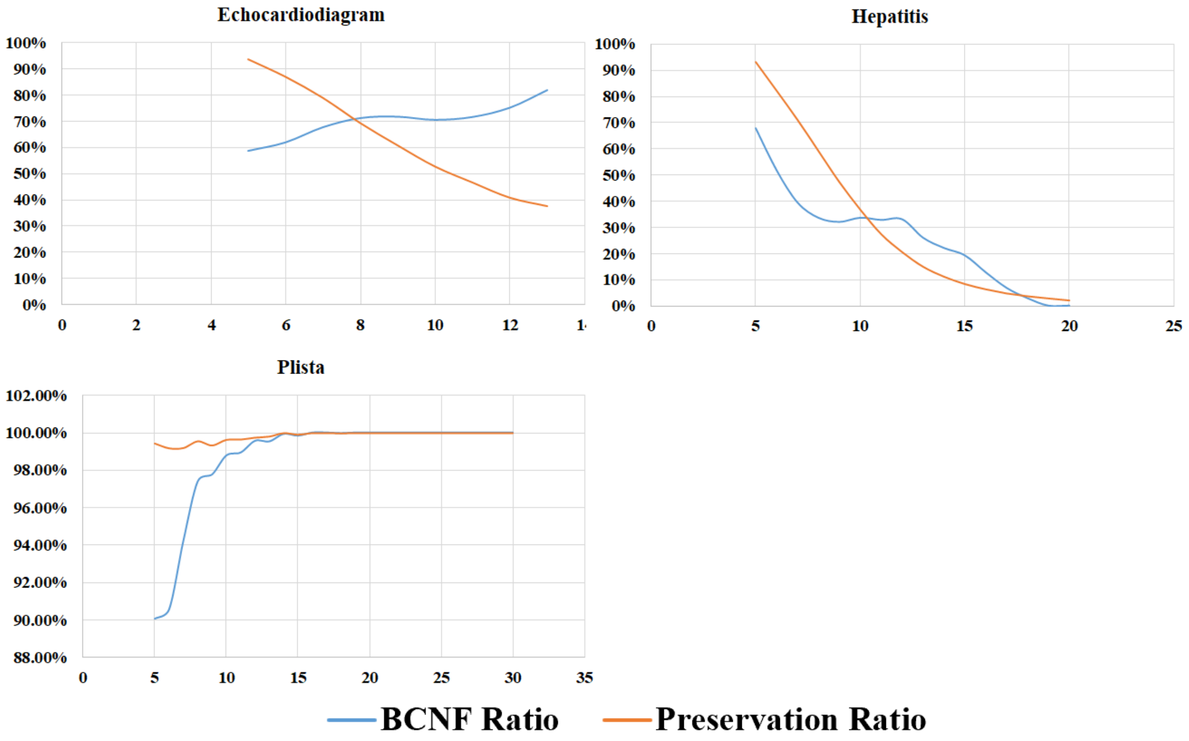


Figure 12: Comparison of Preserving-ratio and BCNF-ratio

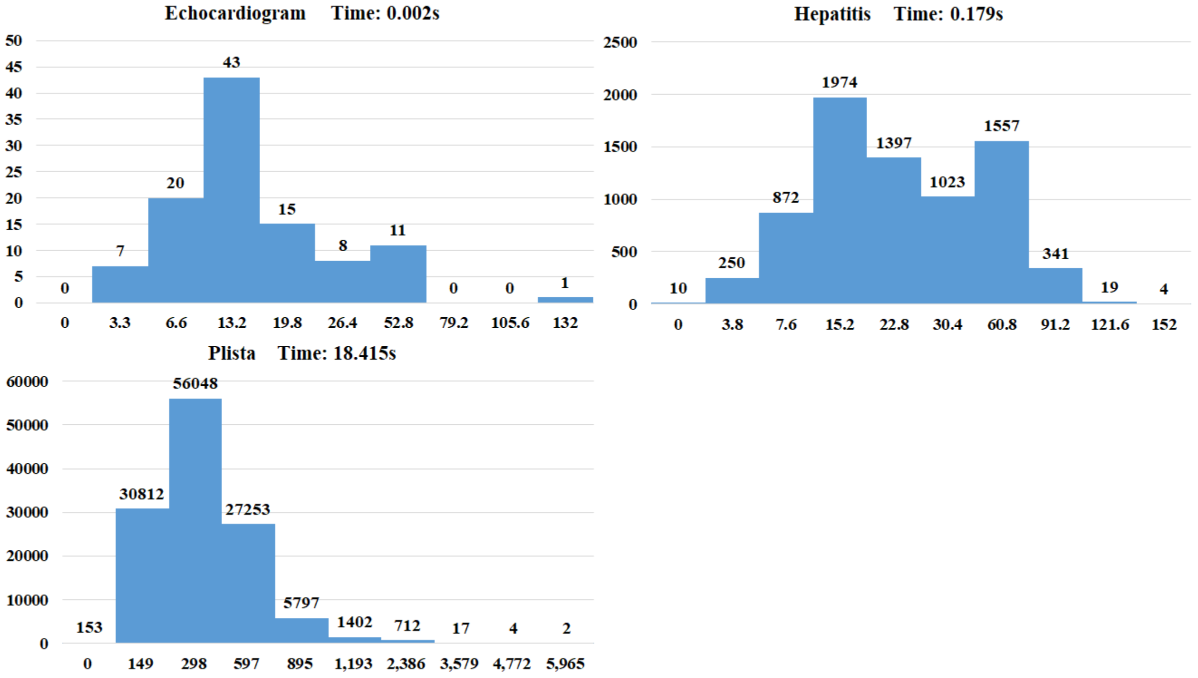


Figure 13: Numbers of eFDs in canonical covers (y-axis) that cause not more than the given number of redundant occurrences (x-axis)

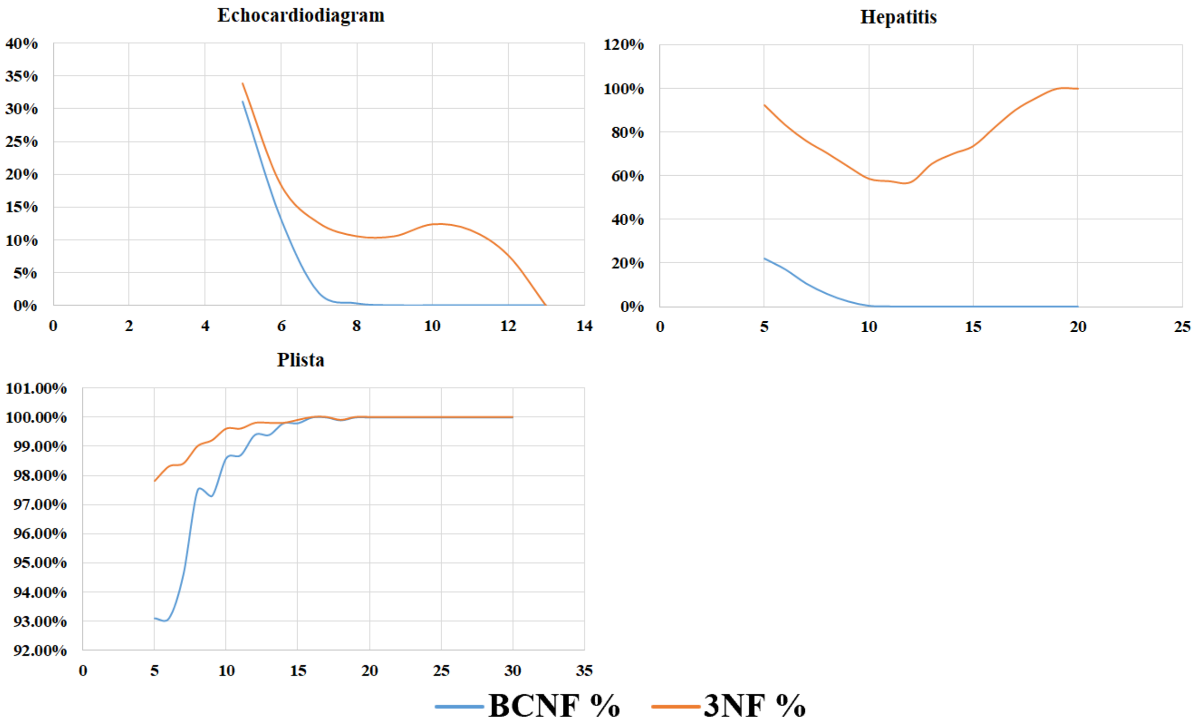


Figure 14: Average percentage of schemata in E-3NF and E-BCNF, respectively, by given size of E

for each data set and each size of E, the percentage of all input sets that are in E-BCNF and E-3NF, respectively.

Figure 3 illustrates the average percentage of the E-complete data value occurrences that are redundant (blue line), the average percentage of redundant occurrences after E-3NF synthesis, and the average percentage of dependencies that have been preserved during E-BCNF decomposition. In general, there is no control about the number of E-redundant data values that an E-3NF synthesized decomposition must tolerate to preserve all relevant dependencies. Vice versa, there is no control on how many relevant dependencies will be lost in order to eliminate all E-redundant data values during E-BCNF decomposition. In bad cases, for example, E-3NF may duplicate non-trivial eFDs across various schemata, resulting in an actual blow-up of the E-redundancies that the eFDs cause, see *diabetic* and *china_weather*.

A.6 Size/time of decompositions

Figure 4 illustrates the impact of the size of E on the cardinality of the decompositions, that is, their total number of attributes. Boldly speaking, the larger the decompositions the more efficient updates (less redundancy) and the less efficient queries (more joins required) we have.

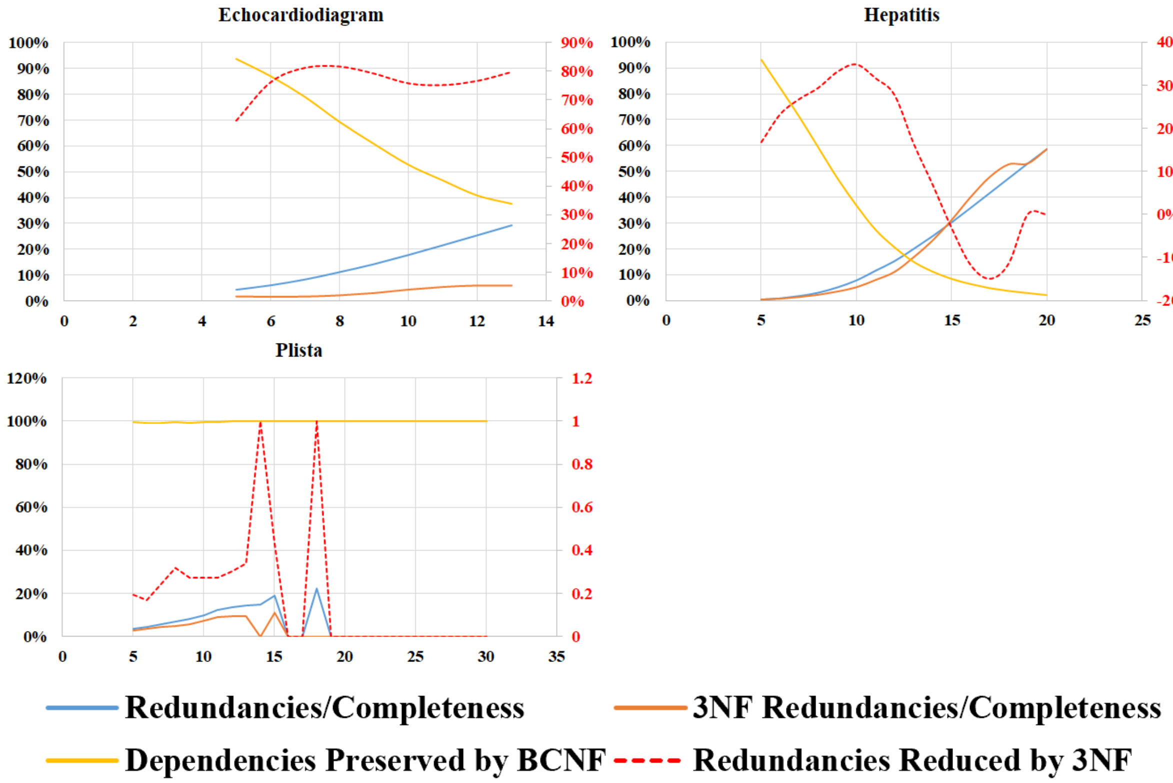


Figure 15: Elimination of E-redundancy by E-3NF (vertical LHS), and E-preservation by E-BCNF (vertical RHS)

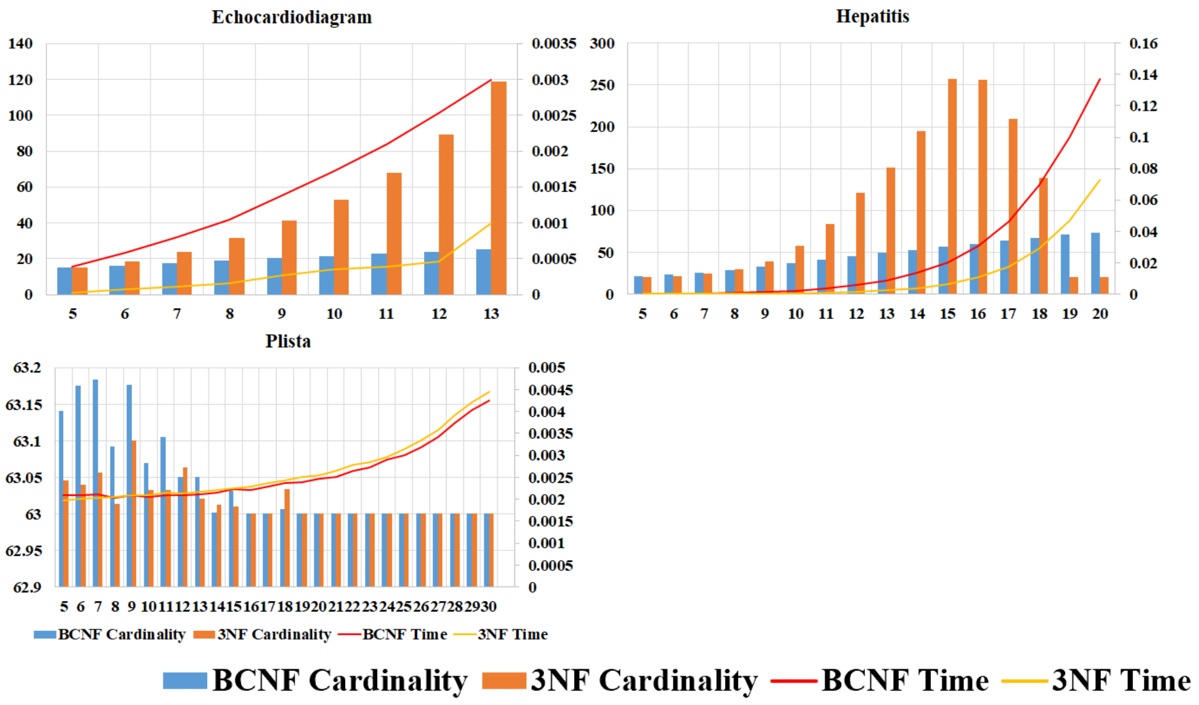


Figure 16: Average cardinality of decompositions and time (s) taken to compute them

B Logical Characterization

We establish a logical characterization showing that eUCs and eFDs interact just like goal and definite clauses in Boolean propositional logic. As a consequence, we can apply linear resolution to reason about eUCs and eFDs, and know that the implication problem is PTIME-complete.

Horn clauses. Let L denote a finite set of propositional variables, and let $L^* \supseteq L$ denote the propositional language over L. An interpretation of L is a total function $\omega : L \to \{\mathbb{F}, \mathbb{T}\}$. Any interpretation ω of L can be lifted to a total function $\Omega : L^* \to \{\mathbb{F}, \mathbb{T}\}$ by:

1. $\Omega(\bar{\varphi}) = \omega(\bar{\varphi})$, if $\bar{\varphi} \in L$,
2. $\Omega(\neg\bar{\varphi}) = \mathbb{T}$ if and only if $\Omega(\bar{\varphi}) = \mathbb{F}$, and
3. $\Omega(\bar{\varphi} \vee \bar{\psi}) = \mathbb{T}$ if and only if $\Omega(\bar{\varphi}) = \mathbb{T}$ or $\Omega(\bar{\psi}) = \mathbb{T}$.

Recall that *literals* are formulae in L^* that are either variables or negations of variables. *Clauses* are disjunctions of literals. A literal is *negative* if it is the negation of a variable, and *positive* otherwise. A clause is a *goal clause* if none of its disjuncts is positive. A clause is a *definite clause* if precisely one of its disjuncts is positive. A clause is *Horn* if at most one of its disjuncts is positive. An interpretation ω is a *model* of a set $\bar{\Sigma}$ of L-formulae if $\Omega(\bar{\sigma}) = \mathbb{T}$ holds for every $\bar{\sigma} \in \bar{\Sigma}$. We say that $\bar{\Sigma}$ *implies* an L-formula $\bar{\varphi}$, denoted by $\bar{\Sigma} \vDash_L \bar{\varphi}$, if every model of $\bar{\Sigma}$ is also a model of $\bar{\varphi}$.

Equivalence. Firstly, we define the fragment of L-formulae that corresponds to eUCs and eFDs over a given relation schema R with respect to $E \subseteq R$. Let $\phi : E \to L$ denote a bijection between E and the set $L = \{\bar{A} \mid A \in E\}$ of propositional variables that corresponds to E. The set $L^{\{\mathbb{F},\mathbb{T}\}}$ consists of all interpretations over L, but if L consists of the variables that correspond to the full set R of attributes, then we exclude the interpretation from $L^{\{\mathbb{F},\mathbb{T}\}}$ that assigns $\mathbb{T}$ to all variables.

We now extend ϕ to a mapping Φ from the set of eUCs and eFDs over R. For an eUC $E' : X'$ over R with $E' \subseteq E = \{A_1, \dots, A_n\}$, let $\Phi(E' : X')$ denote the goal clause $\neg\bar{A}_1 \vee \cdots \vee \neg\bar{A}_n$. For an eFD $E' : B_1, \dots, B_m \to B$ over R with $E' \subseteq E$, let $\Phi(E' : B_1, \dots, B_n \to B)$ denote the definite clause $\neg\bar{B}_1 \vee \cdots \vee \neg\bar{B}_n \vee \bar{B}$. Without loss of generality, eFDs have only a single attribute on their right-hand side. As usual, disjunctions over zero disjuncts are interpreted as $\mathbb{F}$. In what follows, we may simply denote $\Phi(\varphi) = \bar{\varphi}$ and $\Phi(\Sigma) = \{\bar{\sigma} \mid \sigma \in \Sigma[E]\} = \bar{\Sigma}$ where $\Sigma_E = \{E' : X' \mid E' : X' \in \Sigma \wedge E' \subseteq E\} \cup \{E' : X' \to A' \mid E' : X' \to A' \in \Sigma \wedge E' \subseteq E\}$.

Our aim is to show that for every relation schema R, for every set $\Sigma \cup \{\varphi = E : \varphi'\}$ of eUCs and eFDs, there is some E-complete R-relation r that satisfies Σ and violates φ if and only if there is a model $\omega_r \in L^{\{\mathbb{F},\mathbb{T}\}}$ of $\bar{\Sigma}$ that is not a model of $\bar{\varphi}$. For arbitrary relations r it is not obvious how to define the interpretation ω_r. However, for deciding the implication problem $\Sigma \vDash E : \varphi'$, it suffices - by Corollary 3 - to examine two-tuple relations that are E-complete. For two-tuple relations $\{t_1, t_2\}$ that are E-complete, we define the *special interpretation* $\omega_{\{t_1,t_2\}} \in L^{\{\mathbb{F},\mathbb{T}\}}$ by

$$\omega_{\{t_1,t_2\}}(\bar{A}) = \begin{cases} \mathbb{F} & \text{, if } t_1(A) \neq t_2(A) \\ \mathbb{T} & \text{, otherwise} \end{cases}$$

for all $\bar{A} \in L$. In particular, if $\{t_1, t_2\}$ is E-complete, then $\omega_{\{t_1,t_2\}}$ is an interpretation over L. For example, if $E = R$, then the two tuples have non-matching values on some attribute and, consequently, $\omega_{\{t_1,t_2\}}$ does not assign $\mathbb{T}$ to all variables in L. The definition of the special interpretation is further justified semantically by the following lemma.

Lemma 1 *For all relation schemata R, for all two-tuple relations $r = \{t_1, t_2\}$ over R that are E-complete, and for all eUCs and all eFDs φ over R with embedding $E' \subseteq E$, r satisfies φ if and only if ω_r is an L-model of $\bar{\varphi}$.*

Proof We show first that if r satisfies φ, then ω_r is a model of $\bar{\varphi}$. If φ denotes the eUC $E' : \{A_1, \ldots, A_n\}$ with $E' \subseteq E$, then $\bar{\varphi} = \neg\bar{A}_1 \vee \cdots \vee \neg\bar{A}_n$. Since r is E-complete, $E' \subseteq E$, and r satisfies φ, it cannot be that for all $i = 1, \ldots, n$, $t_1(A_i) = t_2(A_i)$ hold. Therefore, it cannot be that for all $i = 1, \ldots, n$, $\omega_r(\neg\bar{A}) = \mathbb{F}$ holds. Hence, ω_r is a model of $\bar{\varphi}$. If φ denotes the eFD $E' : A_1 \cdots A_n \to A$ where $E' \subseteq E$, then $\bar{\varphi} = \neg\bar{A}_1 \vee \cdots \vee \neg\bar{A}_n \vee A$. Since r satisfies φ it cannot be that for all $i = 1, \ldots, n$, $t_1(A_i) = t_2(A_i)$ and $t_1(A) \neq t_2(A)$ holds, too. Therefore, it cannot be that for all $i = 1, \ldots, n$, $\omega_r(\neg\bar{A}_i) = \mathbb{F}$ hold and $\omega_r(\neg\bar{A}) = \mathbb{F}$ holds, too. Hence, ω_r is a model of $\bar{\varphi}$.

We show next that if ω_r is a model of $\bar{\varphi}$, then r satisfies φ. If $\bar{\varphi}$ denotes the goal clause $\neg\bar{A}_1 \vee \cdots \vee \neg\bar{A}_n$, then φ denotes the eUC $E' : \{A_1, \ldots, A_n\}$ for some $E' \subseteq E$. Since ω_r is a model of $\neg\bar{A}_1 \vee \cdots \vee \neg\bar{A}_n$ it cannot be that for all $i = 1, \ldots, n$, $\omega_r(\neg\bar{A}_i) = \mathbb{F}$ holds. Consequently, it cannot be that for all $i = 1, \ldots, n$, $t_1(A_i) = t_2(A_i)$ hold. Hence, r satisfies the eUC $E' : \{A_1, \ldots, A_n\}$. If $\bar{\varphi}$ denotes the definite clause $\neg\bar{A}_1 \vee \cdots \vee \neg\bar{A}_n \vee \bar{A}$, then φ denotes the eFD $E' : A_1, \ldots, A_n \to A$ for some $E' \subseteq E$. Since ω_r is a model of $\neg\bar{A}_1 \vee \cdots \vee \neg\bar{A}_n \vee \bar{A}$ it cannot be that for all $i = 1, \ldots, n$, $\omega_r(\neg\bar{A}_i) = \mathbb{F}$ hold and $\omega_r(\bar{A}) = \mathbb{F}$ holds, too. Consequently, it cannot be that for all $i = 1, \ldots, n$, $t_1(A_i) = t_2(A_i)$ hold and $t_1(A) \neq t_2(A)$ holds, too. Hence, r satisfies the eFD φ. This concludes the proof.

The following corollary follows directly from the proof of Theorem 1.

Corollary 2 *Finite, unrestricted, and two-tuple implication problems for eUCs and eFDs coincide.*

Proof For any given set $\Sigma \cup \{\varphi\}$ of eUCs and eFDs over relation schema R, the proof of Theorem 1 shows that there is always a two-tuple relation that satisfies all elements in Σ and violates φ whenever φ is not implied by Σ. Here, implication may refer to unrestricted or finite implication.

Our main result relies on Corollary 3 and Lemma 1.

Theorem 11 *Let $\Sigma \cup \{\varphi\}$ be a set of eUCs and eFDs over the relation schema R where $\varphi = E : \varphi'$ with $E \subseteq R$. Let L denote the set of propositional variables that corresponds to E, and $\bar{\Sigma} \cup \{\bar{\varphi}\}$ the set of goal and definite clauses over L that corresponds to $\Sigma \cup \{\varphi\}$. Then $\Sigma \models \varphi$ if and only if $\bar{\Sigma} \models_L \bar{\varphi}$.*

Proof According to Corollary 3 it suffices to show that $\Sigma \models_2 \varphi$ if and only if $\bar{\Sigma} \models_L \bar{\varphi}$.

We show first that if $\bar{\Sigma} \models_L \bar{\varphi}$ holds, then $\Sigma \models_2 \varphi$ holds as well. For this purpose suppose that $\Sigma \models_2 \varphi$ does not hold. Consequently, there is some E-complete two-tuple relation r over R that satisfies Σ but violates φ. Following Lemma 1, ω_r is an interpretation that satisfies $\bar{\Sigma}$ and violates $\bar{\varphi}$. Hence, $\bar{\Sigma} \models_L \bar{\varphi}$ does also not hold.

It now remains to show that if $\Sigma \models_2 \varphi$ holds, then $\bar{\Sigma} \models_L \bar{\varphi}$ holds as well. For this purpose, suppose that $\bar{\Sigma} \models_L \bar{\varphi}$ does not hold. Consequently, there is some interpretation ω that is a model of $\bar{\Sigma}$ but not a model of $\bar{\varphi}$. Let $r = \{t_1, t_2\}$ over R be defined as follows: for all $A \in R$ let $t_1(A) \in dom(A) - \{\perp\}$, and let $t_2(A) = t_1(A)$, if $\omega(\bar{A}) = \mathbb{T}$, let $t_2(A) \in dom(A) - \{t_1(A), \perp\}$, if $\omega(\bar{A}) = \mathbb{F}$, and let $t_2(A) = \perp$, if $A \notin E$. It follows that $\omega_r = \omega$. Since $\omega_r = \omega$, Lemma 1 guarantees that r satisfies all elements $E' : \sigma' \in \Sigma$ where $E' \subseteq E$, and r violates φ. However, r also satisfies all $E' : \sigma' \in \Sigma$ where $E' \nsubseteq E$ holds since $r^{E'} = \{t_1\}$ is a singleton in that case. We conclude that $\Sigma \not\models_2 \varphi$.

We analyze our running example from Theorem 11.

Example 15 *As before,* $R = \{f, l, p, d\}$ *and* $\Sigma = \{fld : fld, d : p \to l\}$. *For* $\varphi = l : dfp$ *we obtain* $\Sigma_E = \Sigma$ *and* $\bar{\Sigma} = \{\neg\bar{f} \vee \neg\bar{l} \vee \neg\bar{d}, \neg\bar{p} \vee \bar{l}\}$. *To show that* $\bar{\Sigma}$ *does not logically imply* $\bar{\varphi} = \neg\bar{d} \vee \neg\bar{p} \vee \neg\bar{f}$, *we show that* $\bar{\Sigma} \cup \{\bar{d}, \bar{p}, \bar{f}\}$ *is unsatisfiable. Indeed, applying linear resolution we derive the empty clause. For* $\varphi = f : p \to l$ *we obtain* $\Sigma_E = \emptyset$ *and need to show that* $\bar{\Sigma} = \emptyset$ *does not logically imply* $\bar{\varphi} = \neg\bar{p} \vee \bar{l}$. *Any interpretation* ω *with* $\omega(p) = \mathbb{T}$ *and* $\omega(l) = \mathbb{F}$ *shows that.*

As a final result, we remark that the implication problem for eUCs and eFDs is complete for PTIME. In particular, hardness follows from the satisfiability problem for Horn clauses [10, 17], while membership follows from Corollary 1.

Theorem 12 *The implication problem for the class of eUCs and eFDs is PTIME-complete.*

Proof PTIME-hardness follows that the satisfiability problem for Horn clauses [10, 17], the closure of PTIME under complements, and Theorem 11.

The implication problem is in PTIME follows from the part of Corollary 1 where we establish the linear-time decidability of the implication problem.

C Proofs

We will provide the proofs for all of our results.

C.1 Axiomatization

Lemma 2 *The rules in* $\mathfrak{C}$ *are sound for the implication of eUCs and eFDs.*

Proof We show soundness for each rule of $\mathfrak{E}$ in turn.

The *(trivial key)*-rule is sound since the scope r^R of every relation r over relation schema R with respect to R is a relation, and can therefore not contain two different tuples with matching values on all the attributes in R.

For the soundness of the *(eUC extension)*-rule assume that a relation r over relation schema R violates the eUC $EE' : UU'$. That is, there are two different EE'-total tuples $t, t' \in r^{EE'}$ such that $t(UU') = t'(UU')$ holds. Since $r^{EE'} \subseteq r^E$ and $U \subseteq UU'$ hold, it follows that there are two different E-total tuples $t, t' \in r^E$ such that $t(U) = t'(U)$ holds. Consequently, the relation r also violates the eUC $E : U$.

The *(trivial eFD)*-rule is sound since the scope r^E of every relation r over relation schema R with respect to E satisfies the trivial FD $XY \to Y$. Note that the latter statement follows from the soundness of the reflexivity rule for traditional functional dependencies.

The *(eFD extension)*-rule is sound since the scope r^E of every relation r over relation schema R with respect to E satisfies the FD $X \to XY$ whenever it satisfies the FD $X \to Y$. The latter statement follows from the soundness of the extension rule for traditional functional dependencies.

For the soundness of the *(eFD transitivity)*-rule assume that a relation r over relation schema R violates the eFD $EE' : X \to Z$. That is, there are two different EE'-total tuples $t, t' \in r^{EE'}$ such that $t(X) = t'(X)$ and $t(Z) \neq t'(Z)$ hold. Due to the soundness of the transitivity rule for traditional functional dependencies, it follows that $r^{EE'}$ violates the FD $X \to Y$ or the FD $Y \to Z$. Since $r^{EE'} \subseteq r^E$ and $r^{EE'} \subseteq r^{E'}$ hold, it follows that r^E violates the FD $X \to Y$ or $r^{E'}$ violates the FD $Y \to Z$. Consequently, r violates the eFD $E : X \to Y$ or the eFD $E' : Y \to Z$.

For the soundness of the *(eUC to eFD)*-rule assume that a relation r over relation schema R violates the eFD $E : X \to E$. That is, there are two E-total tuples $t, t' \in r^E$ such that $t(X) = t'(X)$ and $t(E) \neq t'(E)$ hold. In particular, t and t' are different tuples. Consequently, there are two different E-total tuples $t, t' \in r^E$ such that $t(X) = t'(X)$ holds. That is, the relation r over relation schema R violates the eUC $E : X$.

For the soundness of the *(eUC pullback)*-rule assume that a relation r over relation schema R violates the eUC $E : X$. That is, there are two different E-total tuples $t, t' \in r^E$ such that $t(X) = t'(X)$ holds. If r violates the eFD $E : X \to Y$, then we are done. Otherwise, r^E satisfies the FD $X \to Y$. In particular, $t(Y) = t'(Y)$ and therefore we have two different E-total tuples $t, t' \in r^E$ such that $t(XY) = t'(XY)$ holds. Hence, the eUC $E : XY$ is violated, too.

Lemma 3 *The following rules are sound for the implication of eUCs and eFDs.*

$$\frac{E : X \to YZ}{E : X \to Y} \qquad \frac{E : X \to Y \qquad E : X \to Z}{E : X \to YZ}$$

(eFD decompose) *(eFD union)*

$$\frac{E : X \to Y}{EE' : X \to Y}$$

(eFD add-on)

$X^+_{E,\Sigma}$	$E - X^+_{E,\Sigma}$	$R - E$
$0\cdots0$	$0\cdots0$	$0\cdots0$
$0\cdots0$	$1\cdots1$	$\bot\cdots\bot$

Table 8: Counterexample relation from the proof of Theorem 1

Proof The (eFD decompose)-rule can be inferred from the rules in $\mathfrak{E}$ as follows:

$$\dfrac{E:X\to YZ \quad \overline{E:YZ\to Y}}{E:X\to Y}$$

and the (eFD union)-rule can be inferred from the rules in $\mathfrak{E}$ as follows:

$$\dfrac{\dfrac{E:X\to Y}{E:X\to XY} \qquad \dfrac{\dfrac{\dfrac{\overline{E:XY\to X} \quad E:X\to Z}{E:XY\to Z}}{E:XY\to XYZ} \qquad \overline{E:XYZ\to YZ}}{E:XY\to YZ}}{E:X\to YZ}$$

The (eFD add-on)-rule can be inferred from the rules in $\mathfrak{E}$ as follows:

$$\dfrac{E:X\to Y \quad \overline{EE':Y\to Y}}{EE':X\to Y}$$

Theorem 13 (Theorem 1 restated) *$\mathfrak{E}$ is a sound and complete axiomatization for the implication of eUCs and eFDs.*

Proof The soundness of the rules in $\mathfrak{E}$ has been established in Lemma 2. It remains to show completeness. For this purpose we proceed classically by contraposition, and assume for arbitrarily given relation schema R, and an eUC and eFD set $\Sigma\cup\{\varphi\}$ over R that $\varphi\notin\Sigma^+_{\mathfrak{E}}$ holds. We need to show that $\varphi\notin\Sigma^*_{\mathfrak{E}}$ holds, too. For that purpose, we will construct a relation r over R such that r satisfies all eUCs and eFDs in Σ, but does not satisfy φ.

Let φ denote either the eUC $E:X$ or the eFD $E:X\to Y$. In either case, let

$$X^+_{E,\Sigma} = \{A\in E \mid \Sigma\vdash_{\mathfrak{E}} E:X\to A\}$$

denote the *attribute set closure* of X with respect to E and Σ. The soundness of the (eFD-union)-rule, established in Lemma 3, shows that $E:X\to X^+_{E,\Sigma}\in\Sigma^+_{\mathfrak{E}}$ holds.

Let $r := \{t,t'\}$ be a relation over R such that $t(A) := 0$ for all $A\in R$ and $t'(A) :=$ $\begin{cases} 0 \text{ , if } A\in X^+_{E,\Sigma} \\ 1 \text{ , if } A\in E - X^+_{E,\Sigma} \\ \bot \text{ , if } A\in R - E \end{cases}$ The relation r is shown in Table 8.

Case 1. Here, $\varphi = E:X$. If $E = R$ and $X^+_{E,\Sigma} = R$, then $R:R$ and $R:X\to X^+_{E,\Sigma} = R:X\to R\in\Sigma^+_{\mathfrak{E}}$, and thus also $R:X = E:X\in\Sigma^+_{\mathfrak{E}}$. This would be a contradiction to our assumption that $E : X \notin \Sigma^+_{\mathfrak{C}}$. Hence, $E \subset R$ or $X^+_{E,\Sigma} \subset R$. That is, $R - E \neq \emptyset$ or ($E - X^+_{E,\Sigma} \neq \emptyset$ if $E = R$). That is, $t \neq t'$ and r is guaranteed to be a two-tuple relation by construction.

Since $X \subseteq X^+_{E,\Sigma}$ holds and $r^E = r$, it follows from the construction of r that r violates $E : X$. It therefore remains to show in this case that r satisfies every eUC and every eFD in Σ. Let σ denote such an element of Σ.

Case 1.a) Here, σ denotes the eUC $E' : X' \in \Sigma$. Assume, to the contrary, that r violates σ. It follows from the construction of r that $E' \subseteq E$ and $X' \subseteq X^+_{E,\Sigma}$ both hold. Applying the (eUC-extension)-rule to $E' : X' \in \Sigma$ gives us $E : X^+_{E,\Sigma} \in \Sigma^+_{\mathfrak{C}}$. Since $E : X \to X^+_{E,\Sigma} \in \Sigma^+_{\mathfrak{C}}$ holds, we can apply the (eUC pullback)-rule to infer $E : X \in \Sigma^+_{\mathfrak{C}}$. This is a contradiction to our assumption that $\varphi = E : X \notin \Sigma^+_{\mathfrak{C}}$. Consequently, our assumption that r violates σ must have been wrong, and we conclude that r satisfies σ in this case.

Case 1.b) Here, σ denotes the eFD $E' : X' \to Y' \in \Sigma$. Assume again, to the contrary, that r violates σ. It follows from the construction of r that $E' \subseteq E$, $X' \subseteq X^+_{E,\Sigma}$, and $Y' \cap (E - X^+_{E,\Sigma}) \neq \emptyset$ all hold. From $E : X \to X^+_{E,\Sigma} \in \Sigma^+_{\mathfrak{C}}$ and $X' \subseteq X^+_{E,\Sigma}$ we infer $E : X \to X' \in \Sigma^+_{\mathfrak{C}}$ by applying the (eFD-decompose)-rule from Lemma 3. Applying the (eFD-transitivity)-rule to $E : X \to X' \in \Sigma^+_{\mathfrak{C}}$ and $E' : X' \to Y' \in \Sigma$, we infer $EE' : X \to Y' \in \Sigma^+_{\mathfrak{C}}$. Since $E' \subseteq E$, we actually have $E : X \to Y' \in \Sigma^+_{\mathfrak{C}}$. According to the definition of the attribute set closure, we must then have that $Y' \subseteq X^+_{E,\Sigma}$. This, however, is a contradiction to $Y' \cap (E - X^+_{E,\Sigma}) \neq \emptyset$. Consequently, our assumption that r violates σ must have been wrong, and we conclude that r satisfies σ in this case.

Case 2. Here, $\varphi = E : X \to Y$. If $X^+_{E,\Sigma} = R$, then $E = R$ and $E : X \to X^+_{E,\Sigma} \in \Sigma^+_{\mathfrak{C}}$ would mean that $R : X \to R \in \Sigma^+_{\mathfrak{C}}$. The (trivial eFD)-rule means that $R : R \to Y \in \Sigma^+_{\mathfrak{C}}$, and applying the (eFD transitivity)-rule to $R : X \to R \in \Sigma^+_{\mathfrak{C}}$ and $R : R \to Y \in \Sigma^+_{\mathfrak{C}}$, we infer $R : X \to Y \in \Sigma^+_{\mathfrak{C}}$. Since $E = R$, we would then derive the contradiction that $\varphi = E : X \to Y \in \Sigma^+_{\mathfrak{C}}$. Consequently, $X^+_{E,\Sigma} \subset R$. Hence, $E - X^+_{E,\Sigma} \neq \emptyset$ or ($R - E \neq \emptyset$ if $E = X^+_{E,\Sigma}$). Thus, $t \neq t'$ and r is a two-tuple relation in this case, too.

If $Y \subseteq X^+_{E,\Sigma}$, then $E : X \to Y \in \Sigma^+_{\mathfrak{C}}$ contrary to our assumption. Hence, $Y \cap (E - X^+_{E,\Sigma}) \neq \emptyset$. Since $X \subseteq X^+_{E,\Sigma}$, $Y \cap (E - X^+_{E,\Sigma}) \neq \emptyset$ and $r^E = r$, it follows from the construction of r that r violates $E : X \to Y$. It therefore remains to show in this case that r satisfies every eUC and every eFD in Σ. Let σ denote such an element of Σ.

Case 2.a) Here, σ denotes the eUC $E' : X' \in \Sigma$. Assume, to the contrary, that r violates σ. It follows from the construction of r that $E' \subseteq E$ and $X' \subseteq X^+_{E,\Sigma}$ both hold. Applying the (eUC-extension)-rule to $E' : X' \in \Sigma$ gives us $E : X^+_{E,\Sigma} \in \Sigma^+_{\mathfrak{C}}$. Since $E : X \to X^+_{E,\Sigma} \in \Sigma^+_{\mathfrak{C}}$ holds, we can apply the (eUC pullback)-rule to infer $E : X \in \Sigma^+_{\mathfrak{C}}$. From the (eUC to eFD)-rule we can then infer $E : X \to E \in \Sigma^+_{\mathfrak{C}}$, and since $E : E \to Y \in \Sigma^+_{\mathfrak{C}}$, we can apply the (eFD transitivity)-rule to $E : X \to E \in \Sigma^+_{\mathfrak{C}}$ and $E : E \to Y \in \Sigma^+_{\mathfrak{C}}$ to infer the contradiction that $E : X \to Y \in \Sigma^+_{\mathfrak{C}}$. Hence, our assumption must have been wrong, and r satisfies σ in this case.

Case 2.b) Here, σ denotes the eFD $E' : X' \to Y' \in \Sigma$. Assume again, to the contrary, that r violates σ. It follows from the construction of r that $E' \subseteq E$, $X' \subseteq X^+_{E,\Sigma}$, and $Y' \cap (E - X^+_{E,\Sigma}) \neq \emptyset$ all hold. From $E : X \to X^+_{E,\Sigma} \in \Sigma^+_{\mathfrak{C}}$ and $X' \subseteq X^+_{E,\Sigma}$ we infer $E : X \to X' \in \Sigma^+_{\mathfrak{C}}$ by applying the (eFD-decompose)-rule from Lemma 3. Applying the (eFD-transitivity)-rule to $E : X \to X' \in \Sigma^+_{\mathfrak{C}}$ and $E' : X' \to Y' \in \Sigma$, we infer $EE' : X \to Y' \in \Sigma^+_{\mathfrak{C}}$. Since $E' \subseteq E$, we actually have $E : X \to Y' \in \Sigma^+_{\mathfrak{C}}$. According to the definition of the attribute set closure, we must then have that $Y' \subseteq X^+_{E,\Sigma}$. This, however, is a contradiction to $Y' \cap (E - X^+_{E,\Sigma}) \neq \emptyset$. Consequently, our assumption that r violates σ must have been wrong, and we conclude that r satisfies σ in this case.

This concludes the completeness proof.

Corollary 3 *Finite, unrestricted, and two-tuple implication problems for eUCs and eFDs coincide.*

Proof For any given set $\Sigma \cup \{\varphi\}$ of eUCs and eFDs over relation schema R, the proof of Theorem 1 shows that there is always a two-tuple relation that satisfies all elements in Σ and violates φ whenever φ is not implied by Σ. Here, implication may refer to unrestricted or finite implication.

C.2 Algorithmic characterization

Theorem 14 (Theorem 2 restated) *Let $\Sigma \cup \{E : X, E : X \to Y\}$ denote a set of eUCs and eFDs over relation schema R. Then:*

1. *$\Sigma \models E : X \to Y$ if and only if $\Sigma[E] \models X \to Y$*
2. *$\Sigma \models E : X$ if and only if a) $E = R = X$, or b) there is some $E' : X' \in \Sigma$ such that $E' \subseteq E$ and $X' \subseteq X^*_{\Sigma[E]}$.*

Proof We start by proving (1), and show first that $\Sigma \models E : X \to Y$ implies $\Sigma[E] \models X \to Y$. For that purpose we proceed by contraposition, assuming that $\Sigma[E] \not\models X \to Y$. Consequently there is some total two-tuple relation $r = \{t, t'\}$ over R such that r violates $X \to Y$ and r satisfies every FD in $\Sigma[E]$. We now define the E-complete two-tuple relation $r_E := \{t, t'_E\}$ over R where, for all $A \in R$,

$$t'_E(A) := \begin{cases} t'(A) & \text{, if } A \in E \\ \perp & \text{, otherwise} \end{cases}$$

Since r violates $X \to Y$, it follows that $r^E_E = r_E$ violates $X \to Y$ since $X, Y \subseteq E$ and the tuple t is unchanged and $t'_E(A) = t'(A)$ for all attributes $A \in E$. It remains to show that $r^E_E = r_E$ satisfies every $\sigma \in \Sigma$. For all $E' \subseteq R$ where $E' - E \neq \emptyset$ it follows that $r^{E'}_E = \{t\}$, and it therefore follows that r_E satisfies every $\sigma \in \Sigma$ where $\sigma = E' : X'$ or $\sigma = E' : X' \to Y'$ for $E' \subseteq R$ with $E' - E \neq \emptyset$. If $\sigma = E' : X' \to Y'$ with $E' \subseteq E$, then $X' \to Y' \in \Sigma[E]$. In this case, however, r_E satisfies σ since r satisfies $X' \to Y'$. This leaves us to consider the case where $\sigma = E' : X'$ with $X' \subseteq E' \subseteq E$. If r_E violates σ, then $t(X') = t'_E(X')$. However, $X' \to R \in \Sigma[E]$ holds in this case, and since r satisfies $X' \to R$, we would arrive at the contradiction that $t = t'$. Consequently, r_E also satisfies σ in this case. We have therefore shown that $\Sigma \not\models E : X \to Y$ holds.

We will now show that $\Sigma[E] \models X \to Y$ implies $\Sigma \models E : X \to Y$. For that purpose we proceed by contraposition, assuming that $\Sigma \not\models E : X \to Y$. Hence, there is an E-complete two-tuple relation $r_E = \{t_E, t'_E\}$ such that r_E violates $E : X \to Y$ and r_E satisfies every $\sigma \in \Sigma$. Define $r := \{t, t'\}$ where

$$t(A) := \begin{cases} t_E(A) & \text{, if } A \in E \\ 0 & \text{, otherwise} \end{cases}$$

and

$$t'(A) := \begin{cases} t'_E(A) & \text{, if } A \in E \\ 1 & \text{, otherwise} \end{cases} .$$

It follows that r is a total two-tuple relation over R, that r violates $X \to Y$ since $X, Y \subseteq E$ and r^E_E violates $X \to Y$, and that r satisfies every $X' \to R, X' \to Y' \in \Sigma[E]$ since r_E satisfies every $E' : X', E' : X' \to Y' \in \Sigma$ where $E' \subseteq E$. It follows that $\Sigma[E] \not\models X \to Y$. This completes the proof of (1).

We will now prove (2), and show first directly that $\Sigma \models E : X$ is implied by a) $E = R = X$, or b) there is some $E' : X' \in \Sigma$ such that $E' \subseteq E$ and $X' \subseteq X^*_{\Sigma[E]}$. If a) $E = R = X$ holds, then the soundness of the (trivial ekey)-rule shows that $\Sigma \models E : X$. If b) holds, then (1) shows that $E : X \to X'$ is implied by Σ since $E' \subseteq E$ and $X' \subseteq X^*_{\Sigma[E]}$. From $E' : X' \in \Sigma$ we conclude $\Sigma \models E : XX'$ by soundness of the (eUC extension)-rule. From $\Sigma \models E : XX'$ and $\Sigma \models E : X \to X'$ we conclude $\Sigma \models E : X$ by soundness of the (eUC pullback)-rule.

It remains to show that $\Sigma \models E : X$ implies a) or b). For that purpose we proceed by contraposition, assuming that neither a) nor b) hold. Since $E \neq R \neq X$ we conclude that E is a proper subset of R. Since b) does not hold, we conclude that for every $E' : X' \in \Sigma$ where $E' \subseteq E$ holds, X' is not a subset of $X^*_{\Sigma[E]}$. Let $r = \{t, t'\}$ denote the following E-complete two-tuple relation over R:

$X^+_{\Sigma[E]}$	$E - X^+_{\Sigma[E]}$	$R - E$
$0 \cdots 0$	$0 \cdots 0$	$0 \cdots 0$
$0 \cdots 0$	$1 \cdots 1$	$\perp \cdots \perp$

Since $X \subseteq X^+_{\Sigma[E]}$ it follows immediately that r violates the eUC $E : X$. It remains to show that r satisfies every $\sigma \in \Sigma$. Let us look first at the case where $\sigma = E' : X'$. If $E' \nsubseteq E$, then $r^{E'}$ consists of only one tuple, so satisfies σ. If $E' \subseteq E$, then X' intersects non-trivially with $E - X^*_{\Sigma[E]}$ since b) does not hold. Consequently, $t(X') \neq t'(X')$. Hence, r satisfies σ in this case, too. Let us now look at the case where $\sigma = E' : X' \to Y'$. If $E' \nsubseteq E$, then $r^{E'}$ consists of only one tuple, so satisfies σ. It remains to consider the case where $E' \subseteq E$. We show the following: if $t(X') = t'(X')$, then $t(Y') = t'(Y')$. Indeed, if $t(X') = t'(X')$, then $X' \subseteq X^+_{\Sigma[E]}$ and by (1) we conclude $\Sigma \models E : X \to X'$. From $E' : X' \to Y' \in \Sigma$ and $\Sigma \models E : X \to X'$ we conclude that $\Sigma \models EE' : X \to Y'$ by the soundness of the (eUC pullback)-rule. Since $E' \subseteq E$ holds, we conclude that $\Sigma \models E : X \to Y'$. According to (1) we obtain that $Y' \subseteq X^*_{\Sigma[E]}$ holds. Consequently, the construction of r shows that $t(Y') = t'(Y')$ holds, too. We conclude that r satisfies every $\sigma \in \Sigma$. This completes the proof of (2).

C.3 Normal Forms

Theorem 15 (Theorem 3 restated) *For all sets* Σ *of eUCs and eFDs over* R*,* R *is in* E*-RFNF for* Σ *if and only if* R *is in RFNF for* $\Sigma[E]$*.*

Proof We show first the following: if R is not in E-RFNF for Σ, then R is in not in RFNF for $\Sigma[E]$. According to our hypothesis, there is some relation r over R that satisfies Σ, an attribute $A \in E$, and a tuple $t \in r^E$ such that $t(A)$ is E-redundant for Σ. Hence, for every E-replacement $\bar{t}$ of $t(A)$, $\bar{r} := (r - \{t\}) \cup \{\bar{t}\}$ violates some eUCs or eFD in Σ. Consequently, there must be some tuple $t' \in r^E$ such that $t \neq t'$, $t(X'A) = t'(X'A)$ and $A \in Y' - X'$ for some non-trivial eFD $E' : X' \to Y' \in \Sigma$ with $E' \subseteq E$. Let $r_C := \{t_C, t'_C\}$ be the complete relation where $t_C(A') = t(A')$ whenever $t(A') \neq \perp$, $t'_C(A') = t'(A')$ whenever $t'(A') \neq \perp$, and $t_C(A')$, $t'_C(A')$ can be arbitrary complete values otherwise such that $t_C(A') \neq t'_C(A')$. In particular, $t_C(E) = t(E)$ and $t'_C(E) = t'(E)$ since $t, t' \in r^E$. Since $t \neq t'$ it follows that $t_C \neq t'_C$. Consequently, r_C is a two-tuple R-relation that satisfies $\Sigma[E]$ since $\{t, t'\}$ is a two-tuple E-complete relation that satisfies Σ. The data value occurrence $t_C(A)$ in r_C is redundant for $\Sigma[E]$ since every replacement $\bar{t}$ of $t(A)$, $\bar{r}_C := (r_C - \{t\}) \cup \{\bar{t}\}$ violates the FD $X' \to Y'$ in $\Sigma[E]$. Hence, R is not in RFNF for $\Sigma[E]$.

We now show: if R is in not in RFNF for $\Sigma[E]$, then R is not in E-RFNF for Σ. According to our hypothesis, there is some complete relation r over R that satisfies $\Sigma[E]$, some tuple $t \in r$, and attribute $A \in R$ such that the data value occurrence $t(A)$ is redundant for $\Sigma[E]$. That is, for every replacement $\bar{t}$ of $t(A)$, $\bar{r} := (r - \{t\}) \cup \{\bar{t}\}$ violates some constraint in $\Sigma[E]$. Consequently, there must be some tuple $t' \in r$ such that $t \neq t'$, $t(X'A) = t'(X'A)$ and $A \in Y' - X'$ for some FD $X' \to Y' \in \Sigma[E]$. In particular, $Y' \subset R$ as otherwise $t = t'$. Hence, there is some $E' : X' \to Y' \in \Sigma$. In particular, $A \in Y' \subseteq E' \subseteq E$. Let $r_I := \{t, t'_I\}$ such that $t'_I(A') = t'(A')$ whenever $A' \in E$, and $t'_I(A') =\perp$ whenever $A' \notin E$. Since r satisfies $\Sigma[E]$, r_I satisfies Σ. In summary, there are an E-complete relation r_I over R that satisfies Σ, an attribute $A \in E$, and a tuple $t \in r_I^E = r_I$ such that every E-replacement $\bar{t}$ of $t(A)$, $\bar{r}_I := (r_I - \{t\}) \cup \{\bar{t}\}$ violates $E' : X' \to Y'\Sigma$. Hence, R is not in E-RFNF for Σ.

The following lemma provides a key insight into the FD-reduct $\Sigma[E]$ of a set Σ of eUCs and eFDs. Indeed, $\Sigma[E]$ implies the key dependency $X \to R$ if and only if the eUC $E : X$ is implied by Σ.

Lemma 4 *Let* Σ *be a set of eUCs and eFDs over relation schema* R*, and* $E \subseteq R$ *an attribute subset of* R*. Then* $\Sigma[E] \models X \to R$ *if and only if* $\Sigma \models E : X$*.*

Proof Assume first that $E = R$. Then the first property of Theorem 2 shows that $\Sigma[R] \models X \to R$ if and only if $\Sigma \models R : X \to R$. However, $\Sigma \models R : X \to R$ if and only if $\Sigma \models R : X$. This shows the lemma for the case where $E = R$.

Assume now that E is a proper subset of R. We show that $\Sigma[E] \models X \to R$ if and only if $(*)$ there is some $E' : X' \in \Sigma$ such that $E' \subseteq E$ and $\Sigma \models E : X \to X'$. The second property of Theorem 2 then ensures that $(*)$ is equivalent to $\Sigma \models E : X$.

If $(*)$ holds, then $X' \to R \in \Sigma[E]$ by definition of $\Sigma[E]$ and $\Sigma[E] \models X \to X'$ by the first property of Theorem 2. Consequently, $\Sigma[E] \models X \to R$ by the soundness of the transitivity rule for traditional FDs. Vice versa, assume that $(*)$ does not hold. That is, for all $X' \to R \in \Sigma[E]$ we have $X' \notin X^+_{\Sigma[E]}$. We will show that $X \to R$ is not implied by $\Sigma[E]$. Let $r = \{t, t'\}$ denote the following complete two-tuple relation over R.

$X^+_{\Sigma[E]}$	$R - X^+_{\Sigma[E]}$
$0 \cdots 0$	$0 \cdots 0$
$0 \cdots 0$	$1 \cdots 1$

For $X' \to R$ to be in $\Sigma[E]$ there are two possibilities: 1) There is some $E' : X' \to R \in \Sigma$ such that $E' \subseteq E$, and 2) There is some $E' : X' \in \Sigma$ such that $E' \subseteq E$. For 1) it follows that $R \subseteq E' \subseteq E$, that is, $R = E' = E$ which would contradict our assumption. Hence, only 2) is possible by assumption. Furthermore, for every $X' \to Y \in \Sigma[E]$ with $Y \neq R$ it follows that there is some $E' : X' \to Y \in \Sigma$ such that $E' \subseteq E$. Hence, $X'Y \subseteq E$. That is, 3) for every $X' \to Y \in \Sigma[E]$ with $Y \neq R$ it follows that $X'Y \subseteq E$.

Since $X \subseteq E$ and E is assumed to be a proper subset of R it follows that X is a proper subset of R. Due to (3) we conclude that $X^+_{\Sigma[E]} \subseteq E$ is also a proper subset of R. Hence, r is a two-tuple relation. Since $X \subseteq X^+_{\Sigma[E]}$ we conclude that $X \to R$ is not satisfied by r. We show now that r satisfies every FD $U \to V \in \Sigma[E]$. Assume that $t(U) = t'(U)$ as otherwise there is nothing to show. It follows that $U \subseteq X^+_{\Sigma[E]}$ and therefore $X \to U \in \Sigma[E]^+_{\mathfrak{A}}$. From $U \to V \in \Sigma[E]$ we conclude that $X \to V \in \Sigma[E]^+_{\mathfrak{A}}$, which means that $V \subseteq X^+_{\Sigma[E]}$ holds as well. Consequently, $t(V) = t'(V)$, so r satisfies $U \to V$. We just showed that $X \to R$ is not implied by $\Sigma[E]$.

Theorem 16 (Theorem 4 restated) *Relation schema R is in E-BCNF for the set Σ of eUCs and eFDs if and only if R is in BCNF for $\Sigma[E]$.*

Proof Let R be in E-BCNF for a set Σ of eUCs and eFDs over R. We show that R is in BCNF for $\Sigma[E]$. For that purpose, let $X \to Y \in \Sigma[E]$ be a non-trivial FD over R. We need to show that $X \to R \in \Sigma[E]^+_{\mathfrak{A}}$ holds. If $Y = R$, then there is nothing to show. Otherwise, since $X \to Y \in \Sigma[E]$ and $Y \neq R$, the definition of $\Sigma[E]$ means that there is some non-trivial $E' : X \to Y \in \Sigma$ such that $E' \subseteq E$. Consequently, $E : X \to Y \in \Sigma^+_{\mathfrak{E}}$. Since R is in E-BCNF for Σ, we also have $E : X \in \Sigma^+_{\mathfrak{E}}$. According to Lemma 4 that means $X \to R \in \Sigma[E]^+_{\mathfrak{A}}$, which is what we had to show.

Vice versa, let R be in BCNF for $\Sigma[E]$. We need to show that R is in E-BCNF for Σ. For that purpose, let $E : X \to Y \in \Sigma^+_{\mathfrak{E}}$ be a non-trivial eFD over R. We need to show that $E : X \in \Sigma^+_{\mathfrak{E}}$ holds. We first observe that $X \neq R$ as other the given eFD would be trivial. Due to Theorem 1 and Theorem 2, $E : X \to Y \in \Sigma^+_{\mathfrak{E}}$ is equivalent to $X \to Y \in \Sigma[E]^+_{\mathfrak{A}}$. Since R is in BCNF for $\Sigma[E]$, $X \to R \in \Sigma[E]^+_{\mathfrak{A}}$ holds, too. According Lemma 4, $E : X \in \Sigma^+_{\mathfrak{E}}$ holds.

Theorem 17 (Theorem 5 restated) *For all relation schemata R, all attribute subsets $E \subseteq R$, and all sets Σ of eUCs and eFDs over R, R is in E-RFNF for Σ if and only if R is in E-BCNF for Σ.*

Proof By Theorem 3, R is in E-RFNF for Σ if and only if R is in RFNF for $\Sigma[E]$. However, the latter is equivalent to R being in BCNF for $\Sigma[E]$. By Theorem 4, R being in BCNF for $\Sigma[E]$ is equivalent to R being in E-BCNF for Σ.

Theorem 18 (Theorem 6 restated) *A relation schema R is in E-BCNF for a set Σ of eUCs and eFDs if and only if for every eFD $E' : X \to Y \in \Sigma$ where $E' \subseteq E$ and $Y \nsubseteq X$, $E : X \in \Sigma^+_{\mathfrak{C}}$. Hence, deciding if a schema is in E-BCNF for Σ is quadratic in Σ.*

Proof Since every eFD $E' : X \to Y \in \Sigma$ where $E' \subseteq E$ implies that $E : X \to Y \in \Sigma^+_{\mathfrak{C}}$, the condition is necessary for R to be in E-BCNF for Σ. It remains to show the opposite.

Assume that for every eFD $E' : X \to Y \in \Sigma$ where $E' \subseteq E$ and $Y \nsubseteq X$, we have $E : X \in \Sigma^+_{\mathfrak{C}}$. Show that R is in E-BCNF for Σ. By Theorem 4, it suffices to show that R is in BCNF for $\Sigma[E]$. That is, for every non-trivial FD $X \to Y \in \Sigma[E]$ we need to show that $X \to R \in \Sigma^+_{\mathfrak{A}}$ holds. Let $X \to Y \in \Sigma[E]$ be a non-trivial FD. By definition of $\Sigma[E]$ it follows that i) $Y = R$ or ii) there is some $E' : X \to Y \in \Sigma$ such that $E' \subseteq E$. If i) holds, then there remains nothing to show. Otherwise, ii) holds and our assumption implies that $E : X \in \Sigma^+_{\mathfrak{C}}$. Lemma 4 shows that $X \to R \in \Sigma[E]^+_{\mathfrak{A}}$. This completes the proof.

Lemma 5 *An attribute $A \in R$ is E-prime for a given set Σ of eUCs and eFDs over R if and only if the attribute A is prime for $\Sigma[E]$.*

Proof Lemma 4 shows that K is a minimal E-key for R for Σ if and only if K is a minimal key for R for $\Sigma[E]$. Consequently, an attribute $A \in R$ is E-prime for Σ if and only if A is prime for $\Sigma[E]$.

Theorem 19 (Theorem 7 restated) *For all relation schemata R, all $E \subseteq R$, and all sets Σ of eUCs and eFDs over R, R is in E-3NF for Σ if and only if R is in 3NF for $\Sigma[E]$.*

Proof Let R be in E-3NF for Σ. We show that R is in 3NF for $\Sigma[E]$.

Let $X \to Y \in \Sigma[E]$ be a non-trivial FD over R. If $Y = R$, then there remains nothing to show. Otherwise, there is some $E' : X \to Y \in \Sigma$ such that $E' \subseteq E$ and $Y \nsubseteq X$. Hence, $E : X \to Y \in \Sigma^+_{\mathfrak{C}}$ and $Y \nsubseteq X$. Since R is in 3NF for Σ it follows that i) $E : X \in \Sigma^+_{\mathfrak{C}}$, or ii) every attribute in $Y - X$ is E-prime. Lemma 4 shows that $X \to R \in \Sigma[E]^+_{\mathfrak{A}}$. Furthermore, ii) implies that every attribute in $Y - X$ is prime for $\Sigma[E]$ by Lemma 5. That is, R is in 3NF for $\Sigma[E]$.

Vice versa, let R be in 3NF for $\Sigma[E]$. We show that R is in E-3NF for Σ.

Let $E : X \to Y \in \Sigma^+_{\mathfrak{C}}$ be non-trivial. If $Y = R$, then there remains nothing to show. So let $Y \subset R$. From $E : X \to Y \in \Sigma^+_{\mathfrak{C}}$ we conclude that $X \to Y \in \Sigma[E]^+_{\mathfrak{A}}$ is non-trivial. Since R is in 3NF for $\Sigma[E]$, it follows that i) $X \to R \in \Sigma[E]^+_{\mathfrak{A}}$ or ii) every attribute in $Y - X$ is prime for $\Sigma[E]$. By Lemma 4 it follows that i) implies $E : X \in \Sigma^+_{\mathfrak{C}}$. By Lemma 5 it follows that ii) implies that every attribute in $Y - X$ is E-prime for Σ. Consequently, R is in E-3NF for Σ.

Theorem 20 (Theorem 8 restated) *R is in E-3NF for a set Σ of eUCs and eFDs over R if and only if for every eFD $E' : X \to Y \in \Sigma$ where $E' \subseteq E$ and $Y \nsubseteq X$, $E : X \in \Sigma^+_{\mathfrak{C}}$ or every attribute in $Y - X$ is E-prime.*

Proof Since every eFD $E' : X \to Y \in \Sigma$ where $E' \subseteq E$ implies that $E : X \to Y \in \Sigma^+_{\mathfrak{C}}$, the condition is necessary for R to be in E-3NF for Σ. It remains to show the opposite.

Assume that for every eFD $E' : X \rightarrow Y \in \Sigma$ where $E' \subseteq E$ and $Y \nsubseteq X$, we have $E : X \in \Sigma^+_{\mathfrak{C}}$ or every attribute in $Y - X$ is E-prime. Show that R is in E-BCNF for Σ. By Theorem 7, it suffices to show that R is in 3NF for $\Sigma[E]$. That is, for every non-trivial FD $X \rightarrow Y \in \Sigma[E]$ we need to show that $X \rightarrow R \in \Sigma^+_{\mathfrak{A}}$ holds or every attribute in $Y - X$ is prime for $\Sigma[E]$. Let $X \rightarrow Y \in \Sigma[E]$ be a non-trivial FD. By definition of $\Sigma[E]$ it follows that i) $Y = R$ or ii) there is some $E' : X \rightarrow Y \in \Sigma$ such that $E' \subseteq E$. If i) holds, then there remains nothing to show. Otherwise, ii) holds and our assumption implies that $E : X \in \Sigma^+_{\mathfrak{C}}$ or every attribute in $Y - X$ is E-prime. Lemma 4 and Lemma 5 show that $X \rightarrow R \in \Sigma[E]^+_{\mathfrak{A}}$ or every attribute in $Y - X$ is prime for $\Sigma[E]$. Hence, R is in 3NF for $\Sigma[E]$. This completes the proof.

C.4 Normalization

Theorem 21 (Theorem 10) *Let $E : X \rightarrow Y$ be an eFD that satisfies the relation r over relation schema R. Then the set of E-complete records of r is the lossless join of its projections on XY and $X(R - Y)$, that is, $r^E = r^E[XY] \bowtie r^E[X(R - Y)]$. Also, r is the disjoint union of the set of E-complete records of r, and the set of records of r with missing data on some column in E, that is, $r = r^E \uplus (r - r^E)$.*

Proof A relation r satisfies $E : X \rightarrow Y$ if and only if r^E satisfies the FD $X \rightarrow Y$. Consequently, the classical decomposition theorem [27] covers the first case that $r^E = r^E[XY] \bowtie r^E[X(R - Y)]$. The second case is trivial.