SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •
From CAS & CAE Members

July 2020, Vol. 63 172001:1–172001:19
https://doi.org/10.1007/s11432-020-2917-1

Unifying logic rules and machine learning for entity enhancing

Wenfei FAN[1,2,3], Ping LU[3*] & Chao TIAN[4]

[1]*School of Informatics, University of Edinburgh, Edinburgh* EHA 9AB*, UK;*
[2]*Shenzhen Institute of Computing Sciences, Shenzhen University, Shenzhen* 518000*, China;*
[3]*Beijing Advanced Innovation Center for Big Data and Brain Computing, Beihang University, Beijing* 100191*, China;*
[4]*Alibaba Group, Hangzhou* 311121*, China*

Received 1 April 2020/Accepted 16 April 2020/Published online 8 June 2020

Abstract This paper proposes a notion of entity enhancing, which unifies entity resolution and conflict resolution, to identify tuples that refer to the same real-world entity and at the same time, correct semantic inconsistencies. We propose to unify rule-based and machine learning (ML) methods for entity enhancing, by embedding ML classifiers as predicates in logic rules. We model entity enhancing by extending the chase. We show that the chase warrants correctness justification and the Church-Rosser property. Moreover, we settle fundamental problems associated with entity enhancing, including the enhancing, consistency, satisfiability, and implication problems, ranging from NP-complete and coNP-complete to Π_2^p-complete. Taken together, these provide a new theoretical framework for unifying entity resolution and conflict resolution.

Keywords logic rules, machine learning, entity enhancing, entity resolution, conflict resolution

Citation Fan W F, Lu P, Tian C. Unifying logic rules and machine learning for entity enhancing. Sci China Inf Sci, 2020, 63(7): 172001, https://doi.org/10.1007/s11432-020-2917-1

1 Introduction

Big data is often characterized by its volume, variety, velocity, and veracity. Among these, veracity concerns the trustworthiness of data in terms of quality and accuracy, and is often considered the most challenging issue among the 4V's (4V: volume, velocity, variety, and veracity). We cannot get away from it no matter whether we like it or not. For instance, poor data quality costs the US economy $3.1 trillion a year [1], and is too costly to ignore.

There are two active research topics about data quality: (1) entity resolution (ER), to identify tuples that refer to the same real-world entity, and (2) conflict resolution (CR), to resolve (semantic) consistencies pertaining to an entity. There has been a large body of work on ER [2–7] and CR [8–12]. To develop effective tools for ER and CR, however, several issues remain to be resolved.

(1) *Logic or machine learning (ML)*? Both logic rules and ML methods have been studied for ER and CR, and none is superb overall. A survey [13] shows that for information extraction in industry, 67% of systems are rule-based, 17% are ML-based, while 16% are hybrids of the two. Is it possible to develop a framework that unifies logic rules and ML classifiers, and takes advantages of both methods?

(2) *ER or CR* ? There has been a host of work on ER and CR, treating them as separate issues. However, it has long been recognized that ER and CR interact with each other and work the best when the two are taken together [14–16]. Is there a uniform framework to conduct ER and CR at the same time?

* Corresponding author (email: luping@buaa.edu.cn)

Table 1 Relation D_1 of schema product

tid	id	seller*	description*	weight	type*	price	online_year*
t_1	p_1	s_1	MacBook Air 13″, mid 2019, 1.6 GH Intel i5, 128 GB SSD	1.3 kg	computer	8.3K	2019
t_2	p_2	s_1	MacBook MVFK2CH/A, 128 GB	1.3 kg*	computer	8.3K*	2019
t_3	p_3	s_2	MacbookAir8,1, i5, 8 GB SDRAM, 128 GB SSD	9.5 kg	computer	7.3K	2018
t_4	p_4	s_1	MacBook MREE2CH/A, Intel i5, 8 GB LPDDR3, 128 GB	1.3 kg*	computer	9.6K	2018
t_5	p_5	s_2	iMac MRQY2CH/A 27″, 1 TB	9.5 kg	computer	12.9K	2019

Table 2 Relation D_2 of schema shop

tid	id	name	date_created*	on_sale_product*
t_6	s_1	Apple official	2015/1/1	p_1
t_7	s_2	Apple	2015/1/1	p_2

Table 3 Relation D_3 of schema delivery

tid	id	item*	quantity*	city*	fee
t_8	d_1	p_3	5K	Beijing	1K*
t_9	d_2	p_5	5K	Beijing	3K

(3) *Heuristic or certain*? Prior methods for ER and CR are typically heuristic: they update data to fix errors but offer no correctness guarantees. That is, the changes may not fix the target errors and worse yet, may introduce new errors. Is it possible to justify fixes with correctness guarantee?

(4) *Collective or separated*? It is known that to accurately identify entities, ER should be conducted collectively across multiple tables [4]. However, most of the existing ER rules are defined on a single relation [8,11] or at most two relations [2,3]; similarly for ML methods [6]. Is it possible to develop logic rules for collective ER and CR, while striking a balance between the expressive power and complexity?

Example 1. To illustrate the challenges, consider three (simplified) relations D_1-D_3 shown in Tables 1–3 (tid for tuple id). E-commerce platforms often use such tables to collect information of products, online shops and deliveries (for ordered goods), respectively. They also want to identify products and shops, and resolve conflicts between various attributes values, e.g., price; these are nontrivial.

(1) A pure rule-based method may check value equality between the seller (id), description, and year for online sale of products to decide whether they refer to the same real-world entity. Among these, the descriptions are usually long texts and contain various terminologies. It is more reasonable to check the semantic similarity for descriptions by ML models, e.g., the descriptions of t_1 and t_2 refer to the same laptop and the two products should be identified. But how can we integrate ML into logic rules?

(2) It is a common sense that if a shop sells two computers of the same series, then the older version (with earlier year for online sale) should be cheaper. However, t_1 and t_4 are both of the MacBook Air 13″ series, while the older t_4 is more expensive. To fix this inconsistency (CR), ML models are needed again to identify the series of the computers t_1 and t_4 based on their descriptions (ER).

(3) The need for fixing conflicts with correctness guarantees is evident, especially when maintaining vital attributes, like the delivery fee for e-commerce platforms. In relation D_3, deliveries t_8 and t_9 are both sent to Beijing, and the corresponding ordered goods are of the same type, total weight and shop. However, they differ in the fee. Simply changing t_1.fee to $3K$ may even lose the original correct value.

(4) Entity resolution often goes across multiple relations. For instance, tuples t_6 and t_7 in D_2 can be matched if t_1 and t_2 in D_1 are identified, since t_6 and t_7 are created on the same date and have the same product on sale. Similarly, products t_3 and t_4 in D_1 are matched if so are the shops t_6 and t_7 in D_2. □

Contributions & challenges. This paper proposes a framework to tackle these challenges.

(1) *A uniform framework* (Section 2). We propose an entity enhancing framework to unity ER and CR. The framework is based on a class of rules, referred to as entity enhancing rules and denoted by REEs. As opposed to traditional data quality rules, REEs (a) may carry ML classifiers as predicates, (b) subsume matching dependencies (MDs) for ER [2] and conditional dependencies (CFDs) for CR [8] as special cases, and (c) extend MDs and CFDs across multiple relations. In light of these, REEs unify rule-based and

Table 4 Complexity for entity enhancing and reasoning about REEs

Entity enhancing	Consistency	Satisfiability	Implication
NP-complete (Theorem 2)	coNP-complete (Theorem 3)	NP-complete (Theorem 4)	Π_2^p-complete (Theorem 5)

ML-based methods, to improve the accuracy of data quality rules and provide logic interpretation of ML classifications. Moreover, REEs support collective ER and CR across multiple relations.

(2) *Entity enhancing* (Section 3). We model entity enhancing by extending the chase [17]. Given a set Σ of REEs and a set E_0 of entities picked by users, it deduces fixes pertaining to E_0, i.e., matches and corrections related to entities in E_0. As opposed to the classical chase, it applies an REE only if its precondition is satisfied by a collection Γ of "ground truth", i.e., data validated by master data, crowdsourcing or user inspection. The fixes are logical consequences of Σ and Γ; hence as long as the REEs and ground truth are correct, so are the fixes. That is, the chase guarantees the correctness of the fixes.

(3) *Theoretical results* (Section 4). We settle the fundamental problems associated with entity enhancing. The complexity bounds are shown in Table 4, annotated with corresponding theorems.

(a) The entity enhancing process is Church-Rosser, i.e., it guarantees to converge at the same fixes no matter what REEs in Σ are used and in what order the rules are applied.

(b) It is NP-complete to deduce a fix by chasing with REEs.

(c) It is coNP-complete to check whether Σ (REEs) and Γ (ground truth) have no conflicts themselves.

(d) The satisfiability and implication problems for REEs are NP-complete and Π_2^p-complete, to check whether REEs are "dirty" themselves and whether an REE is entailed by other REEs, respectively.

Taken together, these provide a uniform framework for collective CR and ER, and propose a simple strategy to take advantage of both rule-based and ML-based methods. It warrants to find correct fixes when provided with correct REEs Σ and ground truth Γ. Better yet, the fundamental problems for entity enhancing with REEs retain the same complexity as their counterparts for CR alone with CFDs [8,10], except the implication problem unless P = NP; the extra complexity of the implication problem is introduced by "collective" REEs. Hence the price is not very high compared to the increased expressivity of REEs.

Related work. A number of rule-based methods have been developed for ER. For instance, a class of MDs and relative candidate keys for matching records are introduced in [2]; ERBlox employs MDs as blocking keys and ML models for entity classification [18]. Collective entity resolution improves the accuracy of ER by determining co-occurring entities together [4]. Datalog-like rules are proposed for collective ER across multiple tables [5]. In addition, Ref. [19] presents a framework that allows us to plug in existing ER algorithms for joint ER on multiple datasets. There has also been work on ER based on crowdsourcing, e.g., Ref. [20] combines pairwise tasks and multi-item tasks in crowd ER, and Ref. [21] studies how to maximize progressive recall in online ER by using an oracle. Moreover, a host of ML techniques have been studied for ER, e.g., deep learning [6,22], active learning [7,23], and unsupervised learning [24].

There has also been a large body of work on CR, notably by employing CFDs [8, 10] and denial constraints (DCs) [11,12]. User interaction is a common practice in CR [25–27], e.g., Ref. [25] incorporates user feedback in data repairing. Two data cleaning systems, namely, Falcon and DANCE, are presented in [26] and [27], respectively. Falcon uses SQL update queries and user interaction to repair data, and DANCE helps domain experts fix violations of integrity constraints. The need for unifying ER and CR was advocated in [14–16], where Ref. [14] studies the interaction between record matching (ER) and data repairing (CR), Ref. [15] identifies and enriches different references in complex information spaces, and Ref. [16] removes data inconsistencies during the ER process by using negative rules. Moreover, both uniqueness constraints and erroneous values are considered for ER in [28]. A notion of certain fixes was introduced in [29] by employing an extension of CFDs as data quality rules. Certain fixes were recently studied for graphs [30]. Moreover, machine learning has also been used to clean databases (CR), e.g., [31].

This study differs from the prior work in the following. (1) This study provides a framework to unify ER and CR in the same process. It also makes a first effort to unify logic and ML by plugging ML classifiers in logic rules to leverage well-trained ML models, and provide logic interpretation to certain ML predictions.

Moreover, our framework does not separate the process into logical and ML stages as opposed to [18]. (2) As opposed to CFDs, MDs, and DCs, REEs can be defined across multiple relations for collective entity enhancing. (3) This study also makes a first effort to provide correctness guarantees for unified CR and ER on relations. (4) We settle the complexity of fundamental problems for entity enhancing, in the presence of ML classifiers and across multiple relations, in contrast to CFDs, MDs, and DCs.

2 Logic rules with machine learning models

In this section, we introduce REEs, to support collective entity resolution and conflict resolution across multiple relations, and to unify logic-based and ML-based methods.

We will define REEs over a database schema $\mathcal{R} = (R_1, \ldots, R_m)$, where each R_i is a relation schema with a fixed set of attributes, and each attribute A has an atomic type, e.g., integer, string, and Boolean. We assume a countably infinite set U from which elements populating databases are drawn; the set U is further partitioned into domains of the atomic types. A tuple t of R_i consists a value for each attribute of R_i from its corresponding domain. A relation of R_i is a set of tuples of R_i. A database $\mathcal{D}$ of $\mathcal{R}$ is $(D_1, \ldots, D_m)$, where D_i is a relation of R_i for $i \in [1, m]$ (see [32] for details). In particular, we assume a designated attribute id for R_i, such that a tuple of R_i represents an entity with identity id.

Predicates. Following tuple relational calculus [32], we define atomic formulas over $\mathcal{R}$ as follows:

$$p ::= R(t) \mid t.A \otimes c \mid t.A \otimes s.B,$$

where $\otimes$ is a comparison operator $=, \neq, <, \leqslant, >, \geqslant$, and c is a constant in U. Here (a) $R(t)$ says that t is a tuple of R; it is well defined if $R \in \mathcal{R}$; (b) $t.A$ denotes an attribute of t; it is well-defined if $R(t)$ is specified and A is an attribute of R; (c) $t.A \otimes c$ is well-defined if $t.A$ is well-defined, $\otimes$ is defined in the domain of A, and c is a value of the type of A in R; and (d) $t.A \otimes s.B$ is well defined if $t.A$ and $s.B$ are compatible, i.e., $R(t)$ and $R'(s)$ are specified, and $A \in R$ and $B \in R'$ have the same type. In particular, $t.\mathsf{id} = s.\mathsf{id}$ (resp. $t.\mathsf{id} \neq s.\mathsf{id}$) denotes that the entities represented by t and s match (resp. do not match).

We refer to $R(t)$ as a relation atom over $\mathcal{R}$, and to t as a tuple variable bounded by $R(t)$.

REEs. An REE φ over $\mathcal{R}$ is a first-order logic formula of the form:

$$X \to Y,$$

where X and Y are conjunctions of predicates over $\mathcal{R}$. Here a predicate is either a well-defined atomic formula over $\mathcal{R}$, or an ML predicate $\mathcal{M}(\bar{x}, \bar{y})$, where $\mathcal{M}$ is an ML classifier for CR or ER, e.g., [6, 7, 23, 33, 34], and $\bar{x}$ and $\bar{y}$ are vectors of pairwise compatible attributes pertaining to two tuple variables t and s that appear in X, respectively. All tuple variables that appear in φ are bounded in X.

We refer to X as the precondition of φ and Y as the consequence of φ.

Example 2. We use the following four REEs to carry out ER and CR discussed in Example 1, where the three relation schemas are denoted as product, shop, and delivery, respectively.

(1) φ_1: $\mathsf{product}(t_a) \wedge \mathsf{product}(t_b) \wedge t_a.\mathsf{online\_year} = t_b.\mathsf{online\_year} \wedge \mathcal{M}_1(t_a.\mathsf{description}, t_b.\mathsf{description}) \wedge X_1 \to t_a.\mathsf{id} = t_b.\mathsf{id} \wedge t_a.\mathsf{weight} = t_b.\mathsf{weight}$. Here X_1 is $\mathsf{shop}(t'_a) \wedge \mathsf{shop}(t'_b) \wedge t_a.\mathsf{seller} = t'_a.\mathsf{id} \wedge t_b.\mathsf{seller} = t'_b.\mathsf{id} \wedge t'_a.\mathsf{id} = t'_b.\mathsf{id}$. It says that if t_a and t_b have the same year for online purchase and "similar" descriptions, and their selling shops are identified by X_1, then they refer to the same product and hence have the same weight. It uses ML model $\mathcal{M}_1$ to check description similarity. As opposed to DCs, CFDs, and MDs, this rule is defined on multiple relations and can conduct collective entity resolution together with φ_2 below.

(2) φ_2: $\mathsf{shop}(t_a) \wedge \mathsf{shop}(t_b) \wedge t_a.\mathsf{date\_created} = t_b.\mathsf{date\_created} \wedge X_2 \to t_a.\mathsf{id} = t_b.\mathsf{id}$. Here X_2 is $\mathsf{product}(t'_a) \wedge \mathsf{product}(t'_b) \wedge t_a.\mathsf{on\_sale\_product} = t'_a.\mathsf{id} \wedge t_b.\mathsf{on\_sale\_product} = t'_b.\mathsf{id} \wedge t'_a.\mathsf{id} = t'_b.\mathsf{id}$. It identifies two shops if they have the same creation date and product on sale (by X_2). It is also collective on multiple relations.

(3) φ_3: $\mathsf{product}(t_a) \wedge \mathsf{product}(t_b) \wedge X_1 \wedge \mathcal{M}_2(t_a.\mathsf{description}, t_b.\mathsf{description}) \wedge t_a.\mathsf{type} =$ "computer" $\wedge$ $t_b.\mathsf{type} =$ "computer" $\wedge\, t_a.\mathsf{online\_year} < t_b.\mathsf{online\_year} \to t_a.\mathsf{price} < t_b.\mathsf{price}$. It states that the price of t_a

is lower than that of t_b if both t_a and t_b are computers of the same series and sold by the same shop, and moreover, the year for online purchase of t_a is earlier than that of t_b. Here another ML model $\mathcal{M}_2$ is adopted to check whether t_a and t_b belong to the same series. Note that φ_3 detects errors based on the timeliness of prices.

(4) φ_4: $\mathsf{delivery}(t''_a) \wedge \mathsf{delivery}(t''_b) \wedge \mathsf{product}(t_a) \wedge \mathsf{product}(t_b) \wedge t''_a.\mathsf{city} = t''_b.\mathsf{city} \wedge t''_a.\mathsf{quantity} = t''_b.\mathsf{quantity} \wedge t''_a.\mathsf{item} = t_a.\mathsf{id} \wedge t''_b.\mathsf{item} = t_b.\mathsf{id} \wedge t_a.\mathsf{weight} = t_b.\mathsf{weight} \wedge t_a.\mathsf{type} = t_b.\mathsf{type} \wedge X_1 \rightarrow t''_a.\mathsf{fee} = t''_b.\mathsf{fee}$. It says that the delivery fee of t''_a and t''_b must be the same if they are sent to the same city and there are equal amount of goods of the same type and weight ordered from the same shop in t''_a and t''_b. This rule is defined collectively across three distinct relations to fix the vital attribute fee of delivery.

In addition, REEs allow us to interpret certain ML predictions in logic.

(5) Consider an REE $\varphi_5 = \mathsf{product}(t_a) \wedge \mathsf{product}(t_b) \wedge X_5 \rightarrow \mathcal{M}_2(t_a.\mathsf{description}, t_b.\mathsf{description})$, where $\mathcal{M}_2$ is the ML classifier of φ_3, X_5 is defined as $\bigwedge_{A_s \in \mathcal{T}} t_a.A_s = t_b.A_s$ and $\mathcal{T}$ denotes a designated set of attributes for technical specifications, e.g., brand, screen size and processor family (not shown in the simplified schema product), whose values are extracted from the textual description of products in applying $\mathcal{M}_2$. Note that X_5 interprets the prediction of $\mathcal{M}_2(t_a.\mathsf{description}, t_b.\mathsf{description})$ in logic.

REEs also allow us to take advantage of both rule-based and ML-based methods. As another example, consider REEs φ_6–φ_8 below, which are defined over relation schemas user(id, name, product_preference, purchase_history) and order(id, buyer, address, time, total_price, discount, final_price, status). The two (simplified) relation schemas can help e-commerce platforms maintain information of user accounts and orders.

(6) φ_6: $\mathsf{order}(t) \wedge t.\mathsf{discount} \geqslant 0 \rightarrow t.\mathsf{total\_price} \geqslant t.\mathsf{final\_price}$. The simple pure rule-based REE says the final price of an order t must not exceed the original total price when there is a discount. Since the prices are exact numeric values, rule-based methods work better than ML models in this case.

(7) φ_7: $\mathsf{user}(t_a) \wedge \mathsf{user}(t_b) \wedge X_7 \rightarrow t_a.\mathsf{id} = t_b.\mathsf{id}$, where X_7 denotes the conjunction of three ML predicates, i.e., $\mathcal{M}_3(t_a.\mathsf{name}, t_b.\mathsf{name})$, $\mathcal{M}_4(t_a.\mathsf{product\_preference}, t_b.\mathsf{product\_preference})$, and $\mathcal{M}_5(t_a.\mathsf{purchase\_history}, t_b.\mathsf{purchase\_history})$. It identifies users by employing an ML method to check the "similarity" between their names, preferences of products and purchase history. For similarity checking and classification of topics, the ML method work better than conventional logical approaches, especially on text fields.

(8) φ_8: $\mathsf{order}(t'_a) \wedge \mathsf{order}(t'_b) \wedge \mathsf{user}(t_a) \wedge \mathsf{user}(t_b) \wedge t'_a.\mathsf{buyer} = t_a.\mathsf{id} \wedge t'_b.\mathsf{buyer} = t_b.\mathsf{id} \wedge \mathcal{M}_6(t'_a.\mathsf{address}, t'_b.\mathsf{address}) \wedge \mathcal{M}_7(t'_a.\mathsf{time}, t'_b.\mathsf{time}) \wedge X_7 \wedge t'_a.\mathsf{status} = t'_b.\mathsf{status} \rightarrow t'_a.\mathsf{id} = t'_b.\mathsf{id}$. It identifies two orders if they are placed by the same user account (see φ_7 above) at similar time and are delivered to similar addresses (which are all determined using ML classifiers), and moreover, they have the same order status, e.g., completed, refunded or canceled. In practice, different yet duplicated orders are likely created at the same time by the same user account using different devices. Here the logical constraint that is enforced on the attribute of status in φ_8 complements pure ML methods and can reduce the false positives. □

As opposed to existing data quality rules, REEs have the following properties.

(1) As shown in Example 2, on the one hand, an REE may plug in ML classifiers for ER or CR as predicates, e.g., REE φ_3 above or $R_1(t_1) \wedge R_2(t_2) \wedge \mathcal{M}(\bar{x}_1, \bar{x}_2) \rightarrow t_1.\mathsf{id} = t_2.\mathsf{id}$. On the other hand, one can write $X \rightarrow \mathcal{M}(\bar{x}_1, \bar{x}_2)$ to interpret ML predication $\mathcal{M}(\bar{x}_1, \bar{x}_2)$ in terms of a logic condition X, e.g., φ_5.

(2) REEs subsume rules for ER (e.g., MDs) and CR (e.g., CFDs and DCs) as special cases. More specifically, (a) DCs [11] are REEs defined with relation atoms $R(t)$ of the same relation R, and atomic formulas of the form $t.A \otimes c$ and $t.A \otimes s.B$. (b) CFDs [8] are REEs defined in terms of two relation atoms $R(t_1)$ and $R(t_2)$ of the same relation R, and equality predicates $t.A = s.B$ and $t.A = c$. (c) MDs [2] can be expressed as REEs $X \rightarrow Y$ with two relation atoms $R_1(t_1)$ and $R_2(t_2)$, equality atoms $x.A = y.B$, $\mathcal{M}(\bar{x}_1, \bar{x}_2)$ that simulates similarity checking, all in X, and $t_1.\mathsf{id} = t_2.\mathsf{id}$ in Y for tuples in R_1 and R_2, respectively.

(3) REEs support collective entity enhancing. Indeed, an REE may carry multiple relation atoms $R_i(t)$ for distinct relations R_i in $\mathcal{R}$, e.g., φ_1–φ_4, beyond CFDs, DCs, and MDs. Moreover, REEs may carry atomic formulas with comparison predicates $=, \neq, <, \leqslant, >, \geqslant$, which are not supported by CFDs and MDs.

Semantics. An REE $\varphi = X \rightarrow Y$ is interpreted along the same lines as tuple relational calculus (see,

e.g., [32]). Consider a database $\mathcal{D}$ of $\mathcal{R}$. A valuation h of tuple variables of φ in $\mathcal{D}$, or simply a valuation of φ, is a mapping that instantiates t in each relation atom $R(t)$ of φ with a tuple in the relation of R in $\mathcal{D}$.

We say that h satisfies a predicate l over $\mathcal{R}$, written as $h \models l$, if the following conditions are satisfied. (1) If l is an atomic formula $R(t)$, $t.A \otimes c$ or $t.A \otimes s.B$ (including $t.\mathsf{id} = s.\mathsf{id}$ and $t.\mathsf{id} \neq s.\mathsf{id}$), then $h \models l$ is interpreted as in tuple relational calculus following the standard semantics of first order logic (see, e.g., [32]). (2) If l is $\mathcal{M}(\bar{x}_1, \bar{x}_2)$, then $h \models l$ if the ML classifier $\mathcal{M}$ predicts true when provided with $(h(\bar{x}_1), h(\bar{x}_2))$, where $h(\bar{x}_1)$ substitutes $h(t).A$ for each $t.A$ in $\bar{x}_1$; similarly for $h(\bar{x}_2)$.

For a conjunction X of predicates over $\mathcal{R}$, we write $h \models X$ if $h \models l$ for all predicates l in X.

A database $\mathcal{D}$ of $\mathcal{R}$ satisfies φ, denoted by $\mathcal{D} \models \varphi$, if for all valuations h of tuple variables of φ in $\mathcal{D}$, if $h \models X$, then $h \models Y$. We say that $\mathcal{D}$ satisfies a set Σ of REEs, denoted by $\mathcal{D} \models \Sigma$, if for all $\varphi \in \Sigma$, $\mathcal{D} \models \varphi$.

Example 3. Let $\mathcal{D}$ be database (D_1, D_2, D_3) of Example 1 and Σ consist of REEs φ_1–φ_4 of Example 2. Then $\mathcal{D} \not\models \varphi_4$ as witnessed by valuation h_4 that maps variable t_a (resp. t_b, t'_a, t'_b, t''_a, t''_b) to tuple t_3 (resp. t_5, t_7, t_7, t_8, t_9). Hence $\mathcal{D} \not\models \Sigma$. Similarly, $\mathcal{D} \not\models \varphi_3$ and $\mathcal{D} \not\models \varphi_1$ (assuming that $p_1 \neq p_2$ in D_1). □

3 Entity enhancing with extended chase

In this section, we show how to conduct entity enhancing with REEs. Consider a database $\mathcal{D}$ of schema $\mathcal{R}$, a set Σ of REEs over $\mathcal{R}$ and a set Γ of ground truth (validated data, see below). Users can iteratively pick a small set E_0 of entities of interest, e.g., tuples to be identified and corrected. Given E_0, we deduce a set of fixes pertaining to E_0, by chasing $\mathcal{D}$ with Σ and Γ, i.e., by applying the rules of Σ to data validated by Γ. Here a fix either identifies entities or updates an attribute value. The fixes are justified as logical consequences of Σ and Γ, such that if Σ and Γ are correct, then so are the fixes.

We first extend the chase (Subsection 3.1). We then show how to enhance entities by the chase (Subsection 3.2).

3.1 Extending the chase

We first specify fixes and ground truth. We then extend the chase.

Fixes. We enhance entities pertaining to E_0 by deducing fixes. We keep track of such fixes in $\bar{U}_{(\mathcal{D},E_0)} = (\mathcal{E}_=, \mathcal{E}_\neq, \mathcal{E}_<, \mathcal{E}_\leqslant, \mathcal{E}_>, \mathcal{E}_\geqslant)$, denoted by $\bar{U}$ when $(\mathcal{D}, E_0)$ is clear from the context, where each $\mathcal{E}_\otimes$ is a relation. More specifically, for each tuple in $\mathcal{D}$ with tuple id tid, (a) a set $[\mathsf{tid}]_=$ (resp. $[\mathsf{tid}]_\neq$) is in $\mathcal{E}_=$ (resp. $\mathcal{E}_\neq$), including the ids of tuples that are validated to be the same as tid (resp. distinct from tid). (b) For each A-attribute of tid, a set $[\mathsf{tid}.A]_\otimes$ is in $\mathcal{E}_\otimes$, where $\otimes$ ranges over $=, \neq, <, \leqslant, >, \geqslant$. Here $[\mathsf{tid}.A]_\otimes$ includes attributes $\mathsf{tid}_1.B$ (resp. constants c) such that $\mathsf{tid}.A \otimes \mathsf{tid}_1.B$ (resp. $\mathsf{tid}.A \otimes c$) is validated.

Observe the following: (a) $\mathcal{E}_=$ is reflexive, symmetric and transitive; (b) $\mathcal{E}_\neq$ is symmetric and "semi-transitive", e.g., from $z \in [y]_=$ and $y \in [x]_\neq$, it follows that $z \in [x]_\neq$; (c) When $\otimes$ is $\leqslant$ or $\geqslant$, $\mathcal{E}_\otimes$ is reflexive and transitive; (d) If $\otimes$ is $<$ or $>$, $\mathcal{E}_\otimes$ is transitive. We deduce fixes with $\mathcal{E}_\otimes$. For instance, if $c \in [\mathsf{tid}.A]_=$ and $\mathsf{tid}_1.B \in [\mathsf{tid}.A]_=$, then c is also in $[\mathsf{tid}_1.B]_=$; and if $c \in [\mathsf{tid}.A]_\leqslant \cap [\mathsf{tid}.A]_\geqslant$, then c is in $[\mathsf{tid}.A]_=$.

For each $x \in [y]_=$, we refer to (x, y) as a fix.

Intuitively, fixes tell us what entities should be identified and what value an attribute should take.

A fix is enforced on $\mathcal{D}$ as follows. If the fix is $\mathsf{tid}_1 \in [\mathsf{tid}]_=$, then the ids of the two tuples are identified. If it is $c \in [\mathsf{tid}.A]_=$, then $\mathsf{tid}.A$ takes value c. If it is $\mathsf{tid}_1.B \in [\mathsf{tid}.A]_=$, then $\mathsf{tid}_1.B$ and $\mathsf{tid}.A$ are equalized, taking value c if some constant $c \in [\mathsf{tid}.A]_=$, or a special value # otherwise indicating a value yet to be determined. Enforcing the fixes of $\bar{U}$ on $\mathcal{D}$ yields an enhanced database of $\mathcal{R}$, denoted by $\mathcal{D}(\bar{U})$.

Validity. We say that $\bar{U}$ is valid if there exist no tuple tid and attribute A such that one of the following happens: (1) $[\mathsf{tid}.A]_=$ includes distinct constants c and d; i.e., an attribute should carry a unique value; (2) $[\mathsf{tid}]_= \cap [\mathsf{tid}]_\neq \neq \emptyset$, $[\mathsf{tid}.A]_= \cap [\mathsf{tid}.A]_\neq \neq \emptyset$; $[\mathsf{tid}.A]_> \cap [\mathsf{tid}.A]_= \neq \emptyset$, $[\mathsf{tid}.A]_< \cap [\mathsf{tid}.A]_= \neq \emptyset$, $[\mathsf{tid}.A]_> \cap [\mathsf{tid}.A]_< \neq \emptyset$. (3) $[\mathsf{tid}.A]_> \not\subseteq [\mathsf{tid}.A]_\geqslant$ and $[\mathsf{tid}.A]_< \not\subseteq [\mathsf{tid}.A]_\leqslant$. Obviously, if $\bar{U}$ is valid, then $\mathcal{D}(\bar{U})$ is well defined.

Ground truth. To justify the correctness of fixes, we employ a block Γ of validated data. Block Γ is initialized by means of (a) master data, "a single repository of high-quality data" for "core business

entities" [35], which is typically maintained by an enterprise, and (b) high-quality knowledge bases. Block Γ is expanded with (i) data validated during entity enhancing via user interaction (see Subsection 3.2), and (ii) crowdsourcing [20,21]. It is periodically checked by domain experts. Block Γ is enclosed in $\mathcal{E}_=$.

The chase. Given a small set E_0 of tuples, the chase deduces fixes pertaining to E_0 by chasing $\mathcal{D}$ with REEs in Σ and ground truth in Γ. It uses a set E to keep track of entities pertaining to E_0, and stores fixes and changes in $\bar{U}$. More specifically, a chase step of $\mathcal{D}$ by Σ at $(\bar{U}, E)$ is

$$(\bar{U}, E) \Rightarrow_{(\varphi,h)} (\bar{U}', E').$$

Here $\varphi = X \rightarrow Y$ is an REE in Σ, and h is a valuation of φ in $\mathcal{D}(\bar{U})$ pertaining to E, i.e., h maps at least one tuple variable of φ to a tuple in E. Moreover, the conditions below are satisfied.

(1) All predicates l in the precondition X are validated. If l is $t.A \otimes c$, then $c \in [\mathsf{tid}.A]_\otimes$; similarly for $t.A \otimes s.B$. If l is $\mathcal{M}(\bar{x}_1, \bar{x}_2)$, then either (a) each $x \in \bar{x}_i$ is validated in the corresponding $[\mathsf{tid}_i.B_j]_=$ for $i \in [1,2]$, and $\mathcal{M}(\bar{x}_1, \bar{x}_2) = \mathsf{true}$; or (b) $\mathcal{M}(\bar{x}_1, \bar{x}_2)$ is annotated to be true (see (2) below).

(2) A predicate l_0 in Y extends $\bar{U}$ to $\bar{U}'$. If h maps t to tuple tid and l_0 is $t.A \otimes c$, then add c to $[\mathsf{tid}.A]_\otimes$; similarly for predicates $t.A \otimes s.B$. In particular, the set $[\mathsf{tid}]_=$ (resp. $[\mathsf{tid}]_\neq$) is expanded with a tuple id when l_0 is $t.\mathsf{id} = s.\mathsf{id}$ (resp. $t.\mathsf{id} \neq s.\mathsf{id}$). If l_0 is $\mathcal{M}(\bar{x}_1, \bar{x}_2)$, then annotate the ML classifier with $\mathcal{M}(\bar{x}_1, \bar{x}_2) = \mathsf{true}$. The changes are propagated, e.g., if $c \in [\mathsf{tid}.A]_\leqslant \cap [\mathsf{tid}.A]_\geqslant$, then add c to $[\mathsf{tid}.A]_=$.

(3) The set E' extends E by including all tuples "updated" by l_0, i.e., either its id is identified with another or one of its attributes is modified. Intuitively, E' is the area affected when enhancing entities in E.

Chasing. A chasing sequence ξ of $\mathcal{D}$ by (Σ, Γ) from E_0 is

$$(\bar{U}_0, E_0), \ \ldots, \ (\bar{U}_k, E_k).$$

In $\bar{U}_0$, $\mathcal{E}_=$ is Γ and $\mathcal{E}_\otimes = \emptyset$ for other $\otimes$. Moreover, for each $i \in [1,k]$, there exist an REE φ in Σ and a valuation h of φ in $\mathcal{D}(\bar{U}_{i-1})$ such that $(\bar{U}_{i-1}, E_{i-1}) \Rightarrow_{(\varphi,h)} (\bar{U}_i, E_i)$ is a valid chase step, i.e., $\bar{U}_i$ is valid.

The chasing sequence is terminal if there exist no φ in Σ, valuation h of φ in $\mathcal{D}(\bar{U}_k)$ and $(\bar{U}_{k+1}, E_{k+1})$ such that the step $(\bar{U}_k, E_k) \Rightarrow_{(\varphi,h)} (\bar{U}_{k+1}, E_{k+1})$ is valid.

Example 4. Continuing with Example 3, let $E_0 = \{t_6, t_8\}$. Assume that the ground truth Γ includes all entity id attribute values and those marked $*$ in Tables 1–3. If an attribute is marked $*$, then all the values in its column have been validated. From E_0, we have the following chase steps of $\mathcal{D}$ by Σ.

(1) $(\bar{U}_0, E_0) \Rightarrow_{(\varphi_1,h_1)} (\bar{U}_1, E_1)$, where $\bar{U}_0$ is derived from Γ as stated above; h_1 maps the variables of φ_1 to tuples t_1, t_2, t_6 and t_7; $\bar{U}_1$ extends $\bar{U}_0$ by adding $t_2.\mathsf{id}$ to $[t_1.\mathsf{id}]_=$ and $t_2.\mathsf{weight}$ to $[t_1.\mathsf{weight}]_=$; and E_1 extends E_0 by including tuples t_1 and t_2. Intuitively, this chase step identifies t_1 and t_2.

(2) The chase proceeds to (a) add $t_7.\mathsf{id}$ to $[t_6.\mathsf{id}]_=$ by applying REE φ_2, i.e., shops t_6 and t_7 are identified; (b) extend $[t_3.\mathsf{weight}]_=$ (resp. $[t_3.\mathsf{id}]_=$) with $t_4.\mathsf{weight}$ (resp. $t_4.\mathsf{id}$) by using φ_1; and (c) add $t_4.\mathsf{price}$ to both $[t_1.\mathsf{price}]_<$ and $[t_2.\mathsf{price}]_<$ by applying φ_3 twice. Note that there is no chase step with REE φ_4. This is because the only valuation of φ_4 in $\mathcal{D}$ involves products t_3 and t_5 but their weights do not satisfy precondition of φ_4. In fact, $t_3.\mathsf{weight}$ is fixed with the correct value from t_4. □

A chasing sequence ξ terminates in one of the following two cases.

(a) No more REEs in Σ can be applied. If so, we say that ξ is valid, with $(E_k, \bar{U}_k, \mathcal{D}(\bar{U}_k))$ as its result.

(b) Either $\bar{U}_0$ is invalid or there exist φ, h, $\bar{U}_{k+1}$ and E_{k+1} such that $(\bar{U}_k, E_k) \Rightarrow_{(\varphi,h)} (\bar{U}_{k+1}, E_{k+1})$ but $\bar{U}_{k+1}$ is invalid. Such ξ is invalid, and the result of the chase is $\perp$ (undefined).

For instance, the chasing sequence of Example 4 is valid. It terminates with enhanced $\mathcal{D}(\bar{U})$ in which (a) (t_1, t_2), (t_3, t_4), and (t_6, t_7) are pairwise identified, and (b) $t_3.\mathsf{weight}$ is updated to $t_4.\mathsf{weight}$=1.3 kg.

Church-Rosser property. Following [32], we say that chasing with REEs is Church-Rosser if for any schema $\mathcal{R}$, any instance $\mathcal{D}$ of $\mathcal{R}$, any set Σ of REEs on $\mathcal{R}$, any block Γ of ground truth and any set E_0 of entities in $\mathcal{D}$, all chasing sequences of $\mathcal{D}$ by (Σ, Γ) from E_0 are terminal, and all terminal sequences converge at the same result, no matter what REEs in Σ are used and in what order the REEs are applied.

We will show that chasing with REEs is Church-Rosser (Section 4). Hence we define the result of chasing $\mathcal{D}$ by (Σ, Γ) from E_0, denoted by $\mathsf{Chase}(\mathcal{D}, \Sigma, \Gamma, E_0)$, as the result of any such terminal chasing sequence.

We say that (Σ, Γ) is consistent if $\mathsf{Chase}(\mathcal{D}, \Sigma, \Gamma, E_0) \neq \perp$ for any database $\mathcal{D}$ of $\mathcal{R}$ and any set E_0 of tuples in $\mathcal{D}$. That is, the rules in Σ and ground truth in Γ have no conflicts.

3.2 An entity enhancing framework

Based on the chase, we present a framework for entity enhancing, in online mode or offline mode.

Online mode. A user may pick a small set $E_0 \subset \mathcal{D}$ of tuples of her interest, and request to enhance entities pertaining to E_0. Upon receiving the request, the framework does the following.

(a) Compute $\mathsf{Chase}(\mathcal{D}, \Sigma, \Gamma, E_0)$ and enhance entities pertaining to E_0 by using the deduced fixes; expand ground truth Γ with the validated fixes. These make one round.

(b) If not all errors pertaining to E_0 are fixed, e.g., if some attributes still have value # or carry distinct values, then interact with users and invite them to validate some values, further expand Γ with the validated data, check the consistency of REEs in Σ and the expanded Γ, and repeat steps (a) and (b).

It ends up with a set E of enhanced entities such that for each tuple t pertaining to E_0 (i.e., in the set E) and each attribute A of t, $t.A$ carries a validated value c. That is, the process only enhances entities pertaining to E_0. It guarantees that each fix deduced is justified, as a logical consequence of the REEs in Σ and the ground truth in Γ. The process also accumulates ground truth in Γ.

Example 5. Suppose that the e-commerce platform wants to inspect and fix all the errors related to shop t_6 and delivery t_8. The user can put tuples t_6 and t_8 in E_0, and the chase runs as shown in Example 4. When the chase terminates, the inconsistency between the prices of t_1 and t_4 still exists (Example 1). Now the user is invited to validate the price of t_4, and the verified value could further expand the ground truth for subsequent processing. Note that the price of t_1 can be readily obtained from $t_2.\mathsf{price}$, since that value has already been validated, and t_1 and t_2 are identified in the chase. □

Offline mode. After Γ accumulates sufficient ground truth, the framework may run in the offline mode to enhance all entities in $\mathcal{D}$. This is essentially step (a) above when E_0 is $\mathcal{D}$, without user interaction. The process may not find all fixes to $\mathcal{D}$ if Γ is not inclusive, but it guarantees all fixes deduced to be justified.

Remark. As opposed to the classical chase [17], (1) an REE is applied only when its precondition is validated with ground truth Γ. Hence any fix in $\mathsf{Chase}(\mathcal{D}, \Sigma, \Gamma, E_0)$ is justified as long as Σ and Γ are correct. (2) Entity enhancing embeds ML classifiers in logic rules; it collectively resolves both entities and conflicts simultaneously by extending CFDs, DCs, and MDs across multiple relations.

4 Fundamental problems

In this section we first show that the extended chase is Church-Rosser. We then establish the complexity of entity enhancing, consistency, satisfiability, and implication problems. We assume w.l.o.g. that ML classifiers are in polynomial time (PTIME) for testing as commonly found in practice.

(1) Church-Rosser. We show that the chase is Church-Rosser with REEs carrying ML models.

Theorem 1. Chasing with REEs is Church-Rosser. □

Proof. It suffices to show that (i) all chasing sequences are finite; and (ii) they have the same result.

(i) *All chasing sequences are finite.* Consider a chasing sequence $\xi = (\bar{U}_0, E_0), \ldots, (\bar{U}_k, E_k)$ of $\mathcal{D}$ by (Σ, Γ) from E_0. Observe that each chase step in ξ either extends $\bar{U}_i$ w.r.t. an atomic formula, or annotates an ML prediction (see Subsection 3.1). The total number of prediction annotations is bounded by $|\Sigma| \times (|\mathcal{D}| + |\Gamma|)^2$ considering all pair combinations of data values in $\mathcal{D}$ and Γ. In addition, since each attribute $\mathsf{tid}.A$ of $\mathcal{D}$ and Γ can appear in at most 6 relations $\mathcal{E}_\otimes$ of the fixes $\bar{U}_i$ (similarly for tid and constant c), where $\otimes$ ranges over $=, \neq, <, \leqslant, >, \geqslant$, we have that the size of each $\bar{U}_i$ is bounded by $6 \times (|\mathcal{D}| + |\Sigma| + |\Gamma|)^2$. Putting these together, the length of ξ is bounded by $25 \times |\Sigma|^2 \times (|\mathcal{D}| + |\Gamma|)^2$, i.e., ξ is finite.

(ii) *All terminal chasing sequences have the same results.* We prove this by contradiction. Suppose that two terminal chasing sequences $\xi_1 = (\bar{U}_0, E_0), \ldots, (\bar{U}_k, E_k)$ and $\xi_2 = (\bar{U}_0, E_0), \ldots, (\bar{U}'_l, E'_l)$ have different

results. Then (a) there must exist operator $\otimes$ $(=, \neq, <, \leqslant, >, \geqslant)$ such that $\mathcal{E}_\otimes \setminus \mathcal{E}'_\otimes \neq \emptyset$ or $\mathcal{E}'_\otimes \setminus \mathcal{E}_\otimes \neq \emptyset$, where $\mathcal{E}_\otimes$ (resp. $\mathcal{E}'_\otimes$) is in $\bar{U}_k$ (resp. $\bar{U}'_l$); or (b) an ML prediction $\mathcal{M}(\bar{x}, \bar{y})$ is annotated true by ξ_1 but not by ξ_2, or vice versa. Assume w.l.o.g. that ξ_1 is valid, and either $l' \in \mathcal{E}'_\otimes \setminus \mathcal{E}_\otimes$ or there is $\mathcal{M}(\bar{x}, \bar{y})$ annotated true by ξ_2 but not by ξ_1. Let $\varphi_j = X \rightarrow Y$ be the REE and h be the valuation of φ_j that are used to deduce l' for the first time in ξ_2 or to annotate $\mathcal{M}(\bar{x}, \bar{y})$. Then by induction on the length of ξ_2, it is easy to verify that $(\bar{U}_k, E_k) \Rightarrow_{(\varphi_j, h)} (\bar{U}_{k+1}, E_{k+1})$ is also a valid chase step, i.e., ξ_1 is not terminal, a contradiction. Similarly, this can be verified when $\mathcal{E}_\otimes \setminus \mathcal{E}'_\otimes \neq \emptyset$ or $\mathcal{M}(\bar{x}, \bar{y})$ is annotated true by ξ_1 but not by ξ_2.

Note that we can treat an ML predicate $\mathcal{M}(\bar{x}, \bar{y})$ as a traditional predicate since it is checked only after either both $\bar{x}$ and $\bar{y}$ have been validated, or it has been annotated in previous chase steps. □

(2) Entity enhancing. We next settle the complexity of the entity enhancing problem, stated as follows.

∘ Input: A database schema $\mathcal{R}$, a database $\mathcal{D}$ of $\mathcal{R}$, a set Σ of REEs over $\mathcal{R}$, a block Γ of ground truth, a set E_0 of tuples in $\mathcal{D}$, and a fix (x, y).

∘ Question: Is $(x, y) \in \bar{U}$, i.e., a fix deduced in the chase? Here $\mathsf{Chase}(\mathcal{D}, \Sigma, \Gamma, E_0) = (E, \bar{U}, \mathcal{D}(\bar{U}))$.

One can verify that chasing with CFDs alone is NP-complete, i.e., CR alone when Σ consists of CFDs only. Below we show that unifying ER and CR with REEs does not make our lives harder.

Theorem 2. The entity enhancing problem is NP-complete with REEs. □

Proof. We first provide an NP algorithm for entity enhancing, and then show that it is NP-hard.

• Upper bound. We provide an NP algorithm for the entity enhancing problem that works as follows.

(a) Guess a chasing sequence $\xi_c = (\bar{U}_0, E_0) \Rightarrow_{(\varphi_1, h_1)} (\bar{U}_1, E_1) \Rightarrow \cdots \Rightarrow (\bar{U}_{m-1}, E_{m-1}) \Rightarrow_{(\varphi_m, h_m)} (\bar{U}_m, E_m)$ of $\mathcal{D}$ by (Σ, Γ) from E_0 such that $m \leqslant 25 \times |\Sigma|^2 \times (|\mathcal{D}| + |\Gamma|)^2$.

(b) Check whether the sequence is terminal and moreover, for each $j \in [0, m-1]$, check whether $(\bar{U}_j, E_j) \Rightarrow_{(\varphi_{j+1}, h_{j+1})} (\bar{U}_{j+1}, E_{j+1})$ is a valid chase step; if not, reject the guess; otherwise continue.

(c) Check whether the fix (x, y) exists in $\bar{U}_m$; if so, return true.

The correctness of the algorithm follows from Theorem 1, i.e., the Church-Rosser property. For its complexity, step (b) is in PTIME by the definition of chase steps; step (c) is in PTIME, since $|\bar{U}_m| \leqslant 25 \times |\Sigma|^2 \times (|\mathcal{D}| + |\Gamma|)^2$ as shown above. Thus the algorithm is in NP. So is the entity enhancing problem.

• Lower bound. We next show that the problem is NP-hard by reduction from the Boolean conjunctive query evaluation (BCQE) problem, which is known to be NP-complete [36]. The BCQE problem is to decide, given a Boolean conjunctive query Q and a database $\mathcal{D}'$, whether $Q(\mathcal{D}')$ is true.

Given Q and $\mathcal{D}'$, we construct a database schema $\mathcal{R}$, a database $\mathcal{D}$ of $\mathcal{R}$, a set Σ of REEs over $\mathcal{R}$, a block Γ of validated data, a set E_0 of entities in $\mathcal{D}$ and a fix (x, y) such that $Q(\mathcal{D}') = \mathsf{true}$ if and only if (x, y) is in $\mathsf{Chase}(\mathcal{D}, \Sigma, \Gamma, E_0)$. We use REEs in Σ to encode the query Q, the database $\mathcal{D}$ to encode $\mathcal{D}'$, and the valuations of REEs in $\mathcal{D}$ to encode $Q(\mathcal{D}')$. More specifically, the instance is built as follows.

(1) The database schema $\mathcal{R}$ consists of all relation schemas that are referenced in Q and $\mathcal{D}'$, and an extra relation schema R' that includes a single attribute A_1 plus id. Intuitively, R' is used to deduce the fix.

(2) The database $\mathcal{D}$ includes all tuples of $\mathcal{D}'$, and two extra tuples t'_1 and t'_2 of schema R' with $t'_1.A_1 = a_1$ and $t'_2.A_1 = a_2$, where a_1 and a_2 are two distinct constants that do not appear in $\mathcal{D}'$.

(3) The set Σ consists of only one REE, namely, $Q \wedge R(t_1) \wedge R(t_2) \rightarrow t_1.\mathsf{id} = t_2.\mathsf{id}$. Intuitively, when $Q(\mathcal{D}') = \mathsf{true}$, we can enforce $t_1.\mathsf{id} = t_2.\mathsf{id}$ in $\mathcal{D}$.

(4) The set E_0 includes $\mathcal{D}$; and Γ consists of all the data in $\mathcal{D}'$ (i.e., for each tuple t in $\mathcal{D}'$ and each attribute A of t, if $t.A$ is c in $\mathcal{D}'$, then $t.A = c$ is in Γ), and none of the data in tuples t'_1 and t'_2.

(5) The fix is $(t'_1.\mathsf{id}, t'_2.\mathsf{id})$, i.e., the two extra tuples t'_1 and t'_2 that aim to refer to the same entity.

It is easy to show that $(t'_1.\mathsf{id}, t'_2.\mathsf{id}) \in \mathsf{Chase}(\mathcal{D}, \Sigma, \Gamma, E_0)$ if and only if $Q(\mathcal{D}') = \mathsf{true}$. □

(3) Consistency. In practice REEs in Σ are discovered from (possibly dirty) real-life data, and the block Γ of ground truth is accumulated in user interactions. While Σ and Γ are periodically inspected by domain experts, we still need to check whether (Σ, Γ) is consistent (see Subsection 3.2), i.e., they have no conflicts themselves. This highlights the need for studying the consistency problem, stated as follows.

∘ Input: A database schema $\mathcal{R}$, a set Σ of REEs over $\mathcal{R}$ and a block Γ of ground truth.

∘ Question: Is (Σ, Γ) consistent?

A similar problem has been shown coNP-complete for a revision of MDs and CFDs [2]. We show that the problem retains the same complexity for REEs, even for REEs defined across multiple relations.

Theorem 3. The consistency problem is coNP-complete. □

Proof. We establish the upper bound and lower bound of the consistency problem. For the upper bound we need the following small model property, which will be proved at the end of the proof.

Lemma 1. Given a set Σ of REEs and a set Γ of ground truth, we can build in PTIME a database $\mathcal{D}_{(\Sigma,\Gamma)}$ and a set $E_0^{(\Sigma,\Gamma)}$ of tuples such that $\mathsf{Chase}(\mathcal{D}_{(\Sigma,\Gamma)}, \Sigma, \Gamma, E_0^{(\Sigma,\Gamma)}) \neq \perp$ if and only if (Σ, Γ) is consistent. □

• Upper bound. The following NP algorithm suffices to check whether (Σ, Γ) is not consistent.

(1) Construct the database $\mathcal{D}_{(\Sigma,\Gamma)}$ and the set $E_0^{(\Sigma,\Gamma)}$ of tuples based on Lemma 1.

(2) Guess a chasing sequence $(\bar{U}_0, E_0), \ldots, (\bar{U}_k, E_k)$ of $\mathcal{D}_{(\Sigma,\Gamma)}$ from $E_0^{(\Sigma,\Gamma)}$ with $k \leqslant 25 \times |\Sigma|^2 \times (|\mathcal{D}| + |\Gamma|)^2$.

(3) For each $j \in [0, k-2]$, check whether $(\bar{U}, E_j) \Rightarrow_{(\varphi_{j+1}, h_{j+1})} (\bar{U}_{j+1}, E_{j+1})$ in the sequence is a valid chase step; if so, continue; otherwise, reject the current guess.

(4) Check whether $(\bar{U}_{k-1}, E_{k-1}) \Rightarrow_{(\varphi_k, h_k)} (\bar{U}_k, E_k)$ is invalid; if so, return true.

The correctness of the algorithm follows from Lemma 1 and Theorem 1. For the time complexity, step (1) is in PTIME by Lemma 1. Both steps (3) and (4) are also in PTIME by the definition of the chase step. Therefore, the algorithm is in NP, and hence the consistency problem is in coNP.

• Lower bound. The coNP-hardness can be verified by reduction from the complement of BCQE (see the proof of Theorem 2 for BCQE). Given Q and $\mathcal{D}'$, we construct a schema $\mathcal{R}$, a set Σ of REEs on $\mathcal{R}$ and a block Γ of validated data such that $Q(\mathcal{D}') = \mathsf{true}$ if and only (Σ, Γ) is not consistent. The construction is similar to the one given in the proof of Theorem 2, except that Γ contains an additional fact $t'_1.\mathsf{id} = t'_2.\mathsf{id}$, and the consequence Y of the only REE in Σ is $t_1.\mathsf{id} \neq t_2.\mathsf{id}$. Intuitively, we use Σ to encode Q, and Γ to encode the tuples in $\mathcal{D}'$. When $Q(\mathcal{D}') = \mathsf{true}$, we can apply Σ to Γ, to deduce $t'_1.\mathsf{id} \neq t'_2.\mathsf{id}$, which contradicts the fact $t'_1.\mathsf{id} = t'_2.\mathsf{id}$ in Γ. Conversely, if (Σ, Γ) is inconsistent, we know that $t'_1.\mathsf{id} \neq t'_2.\mathsf{id}$ must be deduced. By Lemma 1, we can further apply rules in Σ to Γ and show that $Q(\mathcal{D}') = \mathsf{true}$.

Proof of Lemma 1. Observe that each chasing sequence starts from the given ground truth Γ and tuples E_0, and "propagates" the changes, i.e., fixes induced by Γ and Σ iteratively. In light of this, we construct the database $\mathcal{D}_{(\Sigma,\Gamma)}$ and tuples $E_0^{(\Sigma,\Gamma)}$ based on Γ such that a mapping can be established between the chasing sequences of arbitrary database $\mathcal{D}$ by Σ and the chasing sequences of $\mathcal{D}_{(\Sigma,\Gamma)}$, from which the consistency of (Σ, Γ) can be readily verified using $\mathcal{D}_{(\Sigma,\Gamma)}$ and $E_0^{(\Sigma,\Gamma)}$ alone.

- Construction of $\mathcal{D}_{(\Sigma,\Gamma)}$. For each validated attribute value $t.A = c$ in Γ, we include a tuple t' in $\mathcal{D}_{(\Sigma,\Gamma)}$ and keep $t'.A = c$ in $\mathcal{D}_{(\Sigma,\Gamma)}$. If Γ contains the validated fact $s.A = r.B$ without any $s.A = c$ or $r.B = c$, then we add both tuples s' and r' to $\mathcal{D}_{(\Sigma,\Gamma)}$ and assign # to $s'.A$ and $r'.B$. If Γ contains $s.\mathsf{id} = r.\mathsf{id}$ that is not instantiated with validated value, then we add tuple s' and r' to $\mathcal{D}_{(\Sigma,\Gamma)}$ with # as their id value. Besides these, we also add an additional tuple t_R with a distinct id to $\mathcal{D}_{(\Sigma,\Gamma)}$ for each relation schema R of $\mathcal{R}$, where $\mathcal{R}$ denotes the database schema over which Σ is defined. The attributes of tuples in $\mathcal{D}_{(\Sigma,\Gamma)}$ are given distinct values if they cannot be decided as above. In fact, a mapping h_d from arbitrary database $\mathcal{D}$ of $\mathcal{R}$ to $\mathcal{D}_{(\Sigma,\Gamma)}$ is determined via this construction, in which (a) $h_d(t) = t'$ if tuple t in $\mathcal{D}$ has validated attributes as specified in Γ; and $h_d(t) = t_R$ when t is a tuple of schema R and has no validated attribute.

- Construction of $E_0^{(\Sigma,\Gamma)}$. The set $E_0^{(\Sigma,\Gamma)}$ simply contains all tuples in $\mathcal{D}_{(\Sigma,\Gamma)}$.

Obviously the construction of both $\mathcal{D}_{(\Sigma,\Gamma)}$ and $E_0^{(\Sigma,\Gamma)}$ can be done in PTIME.

We next show that $\mathsf{Chase}(\mathcal{D}_{(\Sigma,\Gamma)}, \Sigma, \Gamma, E_0^{(\Sigma,\Gamma)}) \neq \perp$ if and only if (Σ, Γ) is consistent. (1) When $\mathsf{Chase}(\mathcal{D}_{(\Sigma,\Gamma)}, \Sigma, \Gamma, E_0^{(\Sigma,\Gamma)}) = \perp$, (Σ, Γ) is inconsistent by the definition of consistency (Subsection 3.1). (2) Conversely, when (Σ, Γ) is not consistent, there exist a database $\mathcal{D}$ of $\mathcal{R}$ and a set E_0 of tuples in $\mathcal{D}$ such that $\mathsf{Chase}(\mathcal{D}, \Sigma, \Gamma, E_0) = \perp$. Assume that $\xi = (\bar{U}_0, E_0), \ldots, (\bar{U}_k, E_k)$ is an invalid chasing sequence of $\mathcal{D}$. We consider the following two cases: (a) $\bar{U}_0$ is invalid, and (b) there exist REE $\varphi \in \Sigma$ and valuation

h of variables of φ such that $(\bar{U}_{k-1}, E_{k-1}) \Rightarrow_{(\varphi,h)} (\bar{U}_k, E_k)$ is a chase step but $\bar{U}_k$ is invalid. For case (a), obviously we have that $\mathsf{Chase}(\mathcal{D}_{(\Sigma,\Gamma)}, \Sigma, \Gamma, E_0^{(\Sigma,\Gamma)}) = \bot$ since Γ is invalid itself. For case (b), based on ξ and the mapping h_d described above, we can construct a chasing sequence ξ_1 of $\mathcal{D}_{(\Sigma,\Gamma)}$ by (Σ, Γ) from $E_0^{(\Sigma,\Gamma)}$ such that ξ_1 is also invalid, i.e., $\mathsf{Chase}(\mathcal{D}_{(\Sigma,\Gamma)}, \Sigma, \Gamma, E_0^{(\Sigma,\Gamma)}) = \bot$. Here ξ_1 and ξ have the same length and apply the same REE φ_i in every corresponding pair of their chase steps. The only difference is that each tuple variable t_x in φ_i is mapped to $h_d(h_i(t_x))$ in ξ_2, where h_i denotes the valuation adopted in ξ_1. □

We next study two classical decision problems that are associated with any class of logic rules.

(4) Satisfiability. Given a set Σ of REEs, we want to know whether the rules in Σ are dirty themselves. This motivates us to study the satisfiability problem for REEs, stated as follows.

∘ Input: A database schema $\mathcal{R}$ and a set Σ of REEs over $\mathcal{R}$.

∘ Question: Does there exist a nonempty instance $\mathcal{D}$ of $\mathcal{R}$ such that $\mathcal{D} \models \Sigma$?

We assume w.l.o.g. that the ML models used in REEs output infinitely many true and false results for different pairs of compatible vectors $\bar{x}$ and $\bar{y}$ taken from any practical dense domain. This is guaranteed by most high dimensional ML classifiers, e.g., decision tree and support vector machine [37].

For functional dependencies (FDs), the satisfiability analysis is trivial: for any set Σ of FDs over $\mathcal{R}$, there exists a nonempty instance $\mathcal{D}$ of $\mathcal{R}$ that satisfies Σ [38]. It is easy to verify that this also holds for equality-generating dependencies (EGDs). However, this no longer holds for CFDs [8] and thus, not for REEs. For instance, consider a set Σ that includes a single REE $R(t)\rightarrow$false, where false is a syntactic sugar for a conflicting condition, e.g., $t.A_1 = 0 \wedge t.A_1 = 1$. Obviously, there is no nonempty instance that satisfies Σ.

Nonetheless, we show that the satisfiability problem for REEs is NP-complete. It is the same as for CFDs, a special case of REEs for CR [8], despite that REEs can be defined across multiple relations.

Theorem 4. The satisfiability problem is NP-complete for REEs. □

Proof. We first show that the satisfiability problem in NP, and then prove that the problem is NP-hard.

• Upper bound. To show the upper bound, we first show a small model property. We transform REEs into Horn formulas, in which all the predicates in REEs are simply treated as Boolean variables. By doing so, the satisfiability checking of REEs can be reduced to testing the satisfiability of a set of Horn formulas.

Normal forms. An REE rule $\varphi = X \rightarrow Y$ is in the normal form if Y consists of only one predicate l, i.e., $\varphi = X \rightarrow l$. Given any REE $\varphi = X \rightarrow Y$, we can construct an equivalent set $\Sigma_\varphi = \{X \rightarrow l \mid l \in Y\}$ of REEs such that for any database $\mathcal{D}$, $\mathcal{D} \models \varphi$ if and only if $\mathcal{D} \models \Sigma_\varphi$. That is, REEs and REEs in the normal form have the same expressive power. In the sequel, we assume that all REEs are in the normal form.

Canonical databases. We adopt canonical databases to verify the satisfiability problem. Assume that the set Σ of REEs is defined over a relational schema $\mathcal{R} = (R_1, \ldots, R_n)$. A canonical database $\mathcal{D}_i^s$ is a database of $\mathcal{R}$ that includes only a single tuple t_i^s in one of its relations, say instance D_i of schema R_i, where all the other relations in $\mathcal{D}_i^s$ are empty. Here tuple t_i^s has distinct variables x_j for all the attributes specified in schema R_i. Obviously, if there exists a canonical database satisfying Σ, then Σ is satisfiable.

Horn formulas. We now show how to construct a set $\mathcal{H}_i^s$ of Horn formulas for each canonical database $\mathcal{D}_i^s$ $(i \in [1, n])$, where each formula has the form of $A_1 \wedge \cdots \wedge A_m \rightarrow B$ and each A_i $(i \in [1, m])$ denotes a Boolean variable; similarly for B. More specifically, $\mathcal{H}_i^s$ includes three types of Horn formulas.

(1) For each attribute A_j specified in relation schema R_i, we add a Horn formula of $\emptyset \rightarrow Z_{t_i^s.A_j=x_j}$ to $\mathcal{H}_i^s$, assuming that the only tuple t_i^s in canonical database $\mathcal{D}_i^s$ carries a distinct variable x_j for attribute A_j. Here $Z_{t_i^s.A_j=x_j}$ is a Boolean variable to indicate the fact that $t_i^s.A_j = x_j$.

(2) For each REE of the form of $R_i(t_1) \wedge \cdots \wedge R_i(t_k) \wedge l_1 \wedge \cdots \wedge l_m \rightarrow l$ in Σ, we add a formula $Z_{h(l_1)} \wedge \cdots \wedge Z_{h(l_m)} \rightarrow Z_{h(l)}$ to $\mathcal{H}_i^s$. Here h refers to the only valuation of the REE in $\mathcal{D}_i^s$, i.e., all variables are mapped to tuple t_i^s; and each $Z_{h(l_j)}$ $(j \in [1, m])$ (resp. $Z_{h(l)}$) is a Boolean variable representing the instantiated predicate $h(l_j)$ (resp. $h(l)$), in which tuple variables in l_j (resp. l) are replaced by t_i^s.

For instance, consider REE $\varphi = R_1(t_1) \wedge R_1(t_2) \wedge (t_1.E = t_2.F) \rightarrow t_1.D = t_2.D$ in the normal form. Suppose that the canonical database $\mathcal{D}_1^s$ has a single tuple t_1^s of R_1. Then we build a Horn formula $Z_{t_1^s.E=t_1^s.F} \rightarrow Z_{t_1^s.D=t_1^s.D}$. Note that both t_1 and t_2 are mapped to tuple t_1^s in the valuation.

(3) To avoid assigning distinct constants c and d that appears in Σ to each attribute A_j in schema R_i, we also include the following Horn formula in $\mathcal{H}_i^s$: $Z_{t_i^s.A_j=c} \wedge Z_{t_i^s.A_j=d} \rightarrow \mathsf{false}$.

To check the satisfiability of Σ on canonical databases, it suffices to consider REEs in Σ that are defined over a single relation. Indeed, if an REE $\varphi = X \rightarrow l$ carries atoms of different relation schemas, then for any canonical database $\mathcal{D}_i^s$ of $\mathcal{R}$, we have that $\mathcal{D}_i^s \models \varphi$. This is because $\mathcal{D}_i^s$ consists of a single tuple. Hence if X carries atoms of distinct relation schemas, one of them cannot be instantiated in $\mathcal{D}_i^s$ and thus $\mathcal{D}_i^s \not\models X$; as a result, $\mathcal{D}_i^s \models \varphi$. Note that all relation atoms are bounded in precondition X (page 4) and hence the consequence l does not contain any new relation atoms. For example, if φ is $R_1(t_1) \wedge R_2(t_2) \wedge t_1.A = t_2.A \rightarrow t_1.B = t_2.B$, then no canonical $\mathcal{D}_i^s$ can instantiate both tuple variables t_1 of R_1 and t_2 of R_2.

Number of variables. Since we aim to give an NP algorithm for the satisfiability problem by testing the satisfiability of Horn formulas built as above, it is necessary to analyze the number of Boolean variables in each $\mathcal{H}_i^s$. To this end, we bound the number of all possible predicates that can induce Boolean variables for the formulas. Observe the following, (1) the number of attributes in each canonical database $\mathcal{D}_i^s$ is at most $|\mathcal{R}|$; (2) for each comparison operator $\otimes$ ranging over $=, \neq, <, \leqslant, >, \geqslant$, the number of predicates in the form of $t.A \otimes c$ and $t.A \otimes s.B$ is bounded by $|\mathcal{R}|^2 + |\mathcal{R}| \times |\Sigma|$, where c is a constant in Σ; and (3) there exist at most $|\Sigma|$ Boolean variables induced by ML predicates $\mathcal{M}(\bar{x}, \bar{y})$, since $\bar{x}$ and $\bar{y}$ are pairwise compatible attributes of tuples, i.e., $\bar{x}$ (resp. $\bar{y}$) is from some tuple t_x (resp. t_y) in $\mathcal{D}_i^s$, and moreover, there exists only one tuple in $\mathcal{D}_i^s$. Therefore, there exist polynomially many possible predicates to induce the Boolean variables, and the construction of each set $\mathcal{H}_i^s$ of Horn formulas can be done in PTIME.

It is easy to verify the following small model property for the satisfiability of REEs.

Lemma 2. Given a set Σ of REEs, (1) if Σ is satisfiable, then there exists a truth assignment of the Boolean variables in some $\mathcal{H}_i^s$ that satisfies the Horn formulas of $\mathcal{H}_i^s$; and (2) if an instance $\mathcal{I}(\mathcal{D}_i^s)$ of some canonical database $\mathcal{D}_i^s$ induces a truth assignment that can satisfy $\mathcal{H}_i^s$, then $\mathcal{I}(\mathcal{D}_i^s) \models \Sigma$. □

Here an instance $\mathcal{I}(\mathcal{D}_i^s)$ of $\mathcal{D}_i^s$ is a database obtained by instantiating variables in $\mathcal{D}_i^s$ with constants. Moreover, the truth assignment of Boolean variables in $\mathcal{H}_i^s$ induced by $\mathcal{I}(\mathcal{D}_i^s)$ is such defined that $Z_{l'} = \mathsf{true}$ (resp. $Z_{l'} = \mathsf{false}$) if and only if the instantiated predicate l' holds (resp. does not hold) in $\mathcal{I}(\mathcal{D}_i^s)$.

Proof of Lemma 2. (1) When Σ is satisfiable, there exists a nonempty instance $\mathcal{D}$ of $\mathcal{R}$ such that $\mathcal{D} \models \Sigma$. Let t be a tuple in the relation of some schema R_i in $\mathcal{D}$, and consider a database $\mathcal{D}_i$ that consists of only one tuple t. It is easy to verify that $\mathcal{D}_i \models \Sigma$. Then we can derive a truth assignment μ for the variables in $\mathcal{H}_i^s$ induced by $\mathcal{D}_i$ as follows. Let h be a valuation of REEs of Σ in $\mathcal{D}_i$, which is unique since $\mathcal{D}_i$ has only one tuple t. For each Boolean variable Z_l of $\mathcal{H}_i^s$, Z_l is assigned true (resp. false) by μ if and only if the instantiated predicate $h_i(l)$ is true (resp. false) in $\mathcal{D}_i$. Then μ is a satisfying truth assignment for $\mathcal{H}_i^s$. Indeed, (a) μ satisfies each formula of the form $\emptyset \rightarrow Z_{t_i^s.A_i=x_i}$ in $\mathcal{H}_i^s$, which just indicates the association of attribute and value; (b) μ satisfies Horn formulas $Z_{h(l_1)} \wedge \cdots \wedge Z_{h(l_m)} \rightarrow Z_{h(l)}$ that are constructed w.r.t. REEs $R_i(t_1) \wedge \cdots \wedge R_i(t_k) \wedge l_1 \wedge \cdots \wedge l_m \rightarrow l$, since such formulas specify conditions that guarantee a nonempty database to satisfy Σ and $\mathcal{D}_i \models \Sigma$; and (c) formulas of the form $Z_{t_i^s.A_j=c} \wedge Z_{t_i^s.A_j=d} \rightarrow \mathsf{false}$ are also satisfied by μ since $\mathcal{D}_i$ is well defined and no attribute is given two distinct values.

(2) Conversely, assume that $\mathcal{I}(\mathcal{D}_i^s)$ induces a truth assignment μ that satisfies $\mathcal{H}_i^s$. Then for any REE $\varphi = R_1(t_1) \wedge \cdots \wedge R_l(t_l) \wedge l_1 \wedge \cdots \wedge l_m \rightarrow l$ in Σ and any valuation h of φ in $\mathcal{I}(\mathcal{D}_i^s)$, if $Z_{h(l_1)} \wedge \cdots \wedge Z_{h(l_m)}$ is true by μ, then $Z_{h(l)}$ must also be true since $Z_{h(l_1)} \wedge \cdots \wedge Z_{h(l_m)} \rightarrow Z_{h(l)}$ is a formula in H_i^s by its construction and μ is a satisfying truth assignment. That is, $h \models \varphi$. Thus $\mathcal{I}(\mathcal{D}_i^s) \models \Sigma$. □

In addition to the small model property, we also need the following Lemma to prove the upper bound.

Lemma 3. Given a truth assignment μ of the Boolean variables in $\mathcal{H}_i^s$, it is in PTIME to check whether there exists a corresponding instance $\mathcal{I}(\mathcal{D}_i^s)$ of the canonical database $\mathcal{D}_i^s$ that can induce μ. □

Proof of Lemma 3. We present the following algorithm to verify the existence of such an instance $\mathcal{I}(\mathcal{D}_i^s)$.

(1) Divide the Boolean variables of $\mathcal{H}_i^s$ into two disjoint sets $L^{\otimes}$ and $L^{\mathcal{M}}$. Here $L^{\otimes}$ includes those that are specified with predicates in the form of $t.A \otimes c$ or $t.A \otimes s.B$, where $\otimes$ is a comparison operator $(=, \neq, <, \leqslant, >, \geqslant)$; and $L^{\mathcal{M}}$ consists of ML predicates $\mathcal{M}(\bar{x}, \bar{y})$.

(2) Compute the ranges ϕ for the variables in canonical database $\mathcal{D}_i^s$ that can induce the truth as-

signment μ for $L^{\otimes}$. That is, the instance $\mathcal{I}(\mathcal{D}_i^s)$ should bind values from ϕ to the variables in $\mathcal{D}_i^s$ and it induces truth assignment μ as stated above. If ϕ is empty, return false; otherwise, continue.

(3) Check whether the truth assignment μ for $L^{\mathcal{M}}$ can be induced by the instance $\mathcal{I}(\mathcal{D}_i^s)$ that binds values from domain ϕ only. If so, return true; otherwise, return false.

The correctness of the algorithm follows the definitions of $L^{\otimes}$ and $L^{\mathcal{M}}$. We next analyze its complexity.

Obviously step (1) can be done in PTIME. For step (2), the predicates in $L^{\otimes}$ specify a set of inequalities based on the truth assignment μ, from which the range ϕ can be deduced in PTIME [39]. In step (3), we do the checking as follows. For each ML predicate $\mathcal{M}(\bar{x}, \bar{y})$ in $L^{\mathcal{M}}$ having $\bar{x}$ and $\bar{y}$ bound to a fixed pair of values in step (2), i.e., the domain of $\bar{x}$ (resp. $\bar{y}$) is a singleton set, we compute the result using $\mathcal{M}$ directly in PTIME. If the result and the truth assignment μ for Boolean variable $Z_{\mathcal{M}(\bar{x},\bar{y})}$ are not consistent, then return false; otherwise true is returned. Note that when $\bar{x}$ or $\bar{y}$ has a dense domain, we do not need to check $\mathcal{M}(\bar{x}, \bar{y})$ by the assumption of the ML models given earlier. □

Algorithm. Given a set Σ of REEs, we provide the following algorithm to check whether Σ is satisfiable.

(1) Construct the the set $\mathcal{H}_i^s$ of Horn formulas for each $i \in [1, n]$.

(2) Guess a truth assignment μ for the Boolean variables that appear in $\mathcal{H}_j^s$ for some $j \in [1, n]$.

(3) Check whether $\mathcal{H}_j^s$ is satisfied by μ; if so, continue; otherwise, reject the current guess.

(4) Check whether there exists an instance $\mathcal{I}(\mathcal{D}_j^s)$ of the canonical database $\mathcal{D}_j^s$ that can induce the truth assignment μ; if so, return true.

The correctness follows from the small model property (Lemma 2). For its complexity, step (1) can be done in PTIME as analyzed above. Step (3) is in PTIME since there exist polynomially many formulas; and step (4) is in PTIME by Lemma 3. Hence the algorithm is in NP, and so is the satisfiability problem.

• Lower bound. Consider constraint-generating dependencies with equality and ordering [40], denoted by CGDs. CGDs are REEs of the form $R_1(t_1) \wedge \ldots \wedge R_n(t_n) \wedge l_1 \wedge \ldots \wedge l_m \rightarrow l$, where $l_1, \ldots, l_m, l$ are predicates of the form $t.A \oplus c$ or $t.A \oplus t'.B$, and $\oplus$ is one of $=, \neq, <, \leqslant$. It is known that the satisfiability problem is NP-complete for CGDs with a fixed number of atom formulas [40]; hence so is the problem for REEs.

As remarked earlier, the satisfiability problem is also NP-complete for CFDs [8], another special case of REEs. But the intractability only holds when CFDs are defined with attributes having a finite domain. In contrast, the satisfiability problem is NP-complete for REEs regardless of the domains of attributes. □

(5) Implication. Another classical problem is the implication problem for REEs. We say that a set Σ of REEs defined over schema $\mathcal{R}$ entails another REE φ over $\mathcal{R}$, denoted by $\Sigma \models \varphi$, if for any database $\mathcal{D}$ of $\mathcal{R}$, if $\mathcal{D} \models \Sigma$ then $\mathcal{D} \models \varphi$. Intuitively, the implication analysis helps us eliminate redundant rules and hence, speed up the entity enhancing process. Hence we study the implication problem for REEs.

◦ Input: A database schema $\mathcal{R}$, a set Σ of REEs and another REE φ, both defined over $\mathcal{R}$.

◦ Question: Does $\Sigma \models \varphi$?

For conventional relational dependencies, it is known that the implication problem for FDs is in linear time [41] and is coNP-complete for CFDs [8]. We show the implication problem for REEs is Π_2^p-complete. Here Π_2^p is the class of problems in $\mathsf{coNP}^{\mathsf{NP}}$. That is, when it comes to the implication analysis, the extra expressive power of REEs does come with a (slightly) higher complexity unless P = NP. As stated below, the higher complexity arises from REEs defined over multiple relations for collective entity enhancing.

Theorem 5. The implication problem is Π_2^p-complete for REEs. It is coNP-complete when checking $\Sigma \models \varphi$ and φ is not collective, i.e., φ does not carry relation atoms over different schemas. □

Proof. Below we first show that (i) the implication problem is Π_2^p-complete. We then prove that (ii) the problem is NP-complete when the REE φ is not defined collectively.

(i) General case. Given a set Σ of REEs and another REE φ, we first give a Π_2^p algorithm to check whether $\Sigma \models \varphi$. We then show that the problem is Π_2^p-hard.

• Upper bound. Similar to the proof of Theorem 4, we also define a canonical database, construct a set of Horn formulas for the canonical database, and establish a small model property for the implication problem by using the truth assignments of Horn formulas. These differ from their counterparts for Theorem 4 in the following. (1) When determining whether $\Sigma \models \varphi$, the canonical database $\mathcal{D}^s$ is defined

w.r.t. those relation schemas that appear in φ only. (2) If there are m occurrences of a relation schema R_i in φ, then we include m tuples in the canonical database w.r.t. R_i, and they still carry distinct variables as in the proof of Theorem 4. (3) The Horn formulas encode REE $\varphi_i \in \Sigma$ and all the valuations of φ_i in the canonical database. (4) For $\varphi = X \rightarrow Y$, we use two groups of extra Horn formulas: (a) for each predicate l in X, a Horn formula $\emptyset \rightarrow Z_l$ to enforce the precondition of φ; and (b) a Horn formula $Z_{l_1} \wedge \cdots \wedge Z_{l_k} \rightarrow \mathsf{false}$, where $l_1, \ldots, l_k$ are predicates in Y, and Y refers to the consequence of φ.

Observe that the canonical database only includes tuples of relation schemas that appear in φ. We do not consider relation schemas R' that appear in Σ but not in φ. In fact, the canonical database $\mathcal{D}^s$ satisfies those REEs that have R' in their preconditions, since the instance of R' in $\mathcal{D}^s$ is empty (see proof of Theorem 4). That is, such REEs have no impact on the small model property for $\Sigma \not\models \varphi$ (see below).

Denote by $\mathcal{H}^s$ (resp. $\mathcal{D}^s$) the union of Horn formulas (resp. canonical database) built as above, and denote by V^s the set of Boolean variables in $\mathcal{H}^s$. Based on the construction of Horn formulas and the statement of the implication problem, we have the following small model property for the implication problem, which can be readily verified along the same lines as the proof of Lemma 2.

Lemma 4. Given a set Σ of REEs and an REE φ, (1) if $\Sigma \not\models \varphi$, then V^s has a truth assignment that satisfies the Horn formulas in $\mathcal{H}^s$, and (2) if an instance $\mathcal{I}(\mathcal{D}^s)$ of the canonical database $\mathcal{D}^s$ induces a truth assignment that can satisfy $\mathcal{H}^s$, then $\Sigma \not\models \varphi$. □

Different from the proof of Theorem 4, there may exist exponentially many Horn formulas in $\mathcal{H}^s$ here since the canonical database can include multiple tuples and the combinations of all valuations have to be considered when constructing the formulas. However, V^s is of polynomial size and we can validate the truth assignments of $\mathcal{H}^s$ by simply guessing unsatisfied formulas in $\mathcal{H}^s$, which results in the following.

Lemma 5. Given a truth assignment μ for V^s, it is in coNP to check whether μ satisfies $\mathcal{H}^s$. □

Algorithm. We now provide the following algorithm to check whether $\Sigma \not\models \varphi$.

(1) Construct the set V^s of Boolean variables for $\mathcal{H}^s$.

(2) Guess a truth assignment μ of V^s.

(3) Check whether μ satisfies all the Horn formulas that can be included in $\mathcal{H}^s$; if so, continue.

(4) Check whether there exists an instance $\mathcal{I}(\mathcal{D}^s)$ of canonical $\mathcal{D}^s$ that can induce μ; if so, return true.

The correctness follows from Lemma 4. For the complexity, step (1) is in PTIME due to polynomial-size V^s. Step (3) is in coNP by Lemma 5 and step (4) is in PTIME by Lemma 3. Hence the algorithm is in $\mathsf{NP}^{\mathsf{coNP}}$, i.e., Σ_2^p. That is, checking $\Sigma \not\models \varphi$ is in Σ_2^p. Therefore, the implication problem is in Π_2^p.

• Lower bound. We prove that the problem is Π_2^p-hard by reduction from the complement of the generalized graph coloring problem, denoted by GGCP, which is known to be Σ_2^p-complete [42,43]. GGCP is to decide, given two undirected graphs $F = (V_F, E_F)$ and $G = (V_G, E_G)$, whether there exists a two-coloring of F such that there is no monochromatic subgraph of F isomorphic to G. Here a monochromatic subgraph of F is a subgraph whose nodes are assigned the same color. It is known that GGCP remains Σ_2^p-complete even when G is a complete graph and F does not contain any self cycles [42].

Given two undirected graphs $F = (V_F, E_F)$ and $G = (V_G, E_G)$, we construct a database schema $\mathcal{R}$, a set Σ of REEs and another REE φ over $\mathcal{R}$ such that $\Sigma \not\models \varphi$ if and only if there exists a 2-coloring of F so that no monochromatic subgraph of F is isomorphic to G. Intuitively, (1) we use two REEs φ_1 and φ_2 in Σ to check the existence of the monochromatic subgraph and to specify the domain of colors, respectively; and (2) REE φ is to encode the structure of F. When $\Sigma \not\models \varphi$, we can deduce a 2-coloring of F from the valuation of φ and further conclude that there does not exist a requested monochromatic subgraph via φ_1. Conversely, suppose that there exists a 2-coloring of F such that F contains no monochromatic subgraph that is isomorphic to G, we can construct an instance $\mathcal{D}$ of $\mathcal{R}$ witnessing $\Sigma \not\models \varphi$.

More specifically, we define $\mathcal{R}$, Σ and φ as follows.

(1) The relational schema $\mathcal{R}$ consists of three relation schemas, namely, R_V, R_E and R_O that are defined with attributes $\{A_V, A_C\}$, $\{A_{V_1}, A_{V_2}\}$ and A_O, respectively. Here we use (a) R_V to encode nodes and their colors, (b) R_E to store edges of F and G, and (c) R_O to witness $\Sigma \models \varphi$, which will be clear soon.

(2) Σ is composed of two REEs φ_1 and φ_2, which are to check the existence of the monochromatic subgraph and enforce the domain of colors, respectively.

(i) REE φ_1 is defined as $X_1^1 \wedge X_2^1 \wedge X_3^1 \rightarrow Y^1$, where (a) X_1^1 is $\bigwedge_{(u,v)\in E_G}(R_V(t_u) \wedge R_V(t_v) \wedge t_u.A_C = t_v.A_C)$, recording nodes of G and their colors; here tuple variables t_u and t_v represent nodes u and v, respectively, and the equality requires that the nodes on each edge of G have the same color; (b) X_2^1 is $\bigwedge_{(u,v)\in E_G}\{(R_E(t_{(u,v)}) \wedge t_{(u,v)}.A_{V_1} = t_u.A_V \wedge t_{(u,v)}.A_{V_2} = t_v.A_V) \wedge (R_E(t_{(v,u)}) \wedge t_{(v,u)}.A_{V_1} = t_v.A_V \wedge t_{(v,u)}.V_2 = t_u.A_V)\}$, which encodes the edges in G; (c) X_3^1 is a relation atom $R_O(t)$ to witness the implication; and (d) Y^1 is to ensure that the A_O attribute value of the tuple t is 1, i.e., $t.A_O = 1$.

(ii) REE φ_2 enforces that the colors can only be 0 or 1. More specifically, φ_2 is defined as $R_V(t_x) \wedge t_x.A_C \neq 0 \wedge t_x.A_C \neq 1 \rightarrow t_x.A_C \neq t_x.A_C$. Since $t_x.A_C \neq t_x.A_C$ never holds, $t_x.C$ is either 0 or 1.

(3) REE φ is to represent the structure of F. Similar to the construction of φ_1, φ is defined as $X_1 \wedge X_2 \wedge X_3 \rightarrow Y_1$. Here X_1, X_2 and X_3 are to encode the nodes and edges of F and enforce a constraint on attribute A_O, respectively, similar to their counterparts in φ_1. More specifically, (a) X_1 is $\bigwedge_{u\in V_F}(R_V(t_u) \wedge t_u.A_V = c_u)$, where c_u is a constant other than 0 or 1, to represent node u of F; (b) X_2 is $\bigwedge_{(u,v)\in E_F}\{(R_E(t_{(u,v)}) \wedge t_{(u,v)}.A_{V_1} = c_u \wedge t_{(u,v)}.A_{V_2} = c_v) \wedge (R_E(t_{(v,u)}) \wedge t_{(v,u)}.A_{V_1} = c_v \wedge t_{(v,u)}.A_{V_2} = c_u)\}$, where each edge (u, v) in F is encoded with two atoms $R_E(t_{(u,v)})$ and $R_E(t_{(v,u)})$; (c) X_3 has only one predicate $R_O(t)$; and (d) Y_1 is defined as $t.A_O = 1$, i.e., the A_O attribute of tuple t is requested to be 1.

Since the structures of G and F are encoded in Σ and φ, we can easily prove that $\Sigma \not\models \varphi$ if and only if there exists a 2-coloring of F such that F does not contain a monochromatic G as a subgraph.

(ii) Special case. When the REE φ is not collective, it does not carry relation atoms over different relation schemas. We show that the implication problem becomes coNP-complete in this case.

For the upper bound, since φ is not collective, the canonical database $\mathcal{D}^s$ consists of only one tuple, and hence $\mathcal{H}^s$ has polynomially many Horn formulas. Then the algorithm for the general case still works, except that now step (3) is in PTIME as in the proof of the satisfiability problem. Hence the algorithm is in NP; so is the implication problem. Note that the REEs in Σ may still carry atoms over different relation schemas, since when establishing the small model property for $\Sigma \not\models \varphi$, the canonical database $\mathcal{D}^s$ and Horn formulas $\mathcal{H}^s$ are constructed w.r.t. the (single) relation schema in φ only (Lemma 4).

The lower bound follows from [40], which shows that the implication problem for CGDs is already coNP-complete. As remarked in the proof of Theorem 4, CGDs are a special case of REEs. □

5 Conclusion

We have proposed a framework that unifies collective ER and CR, extends logic rules with ML classifiers, and warrants correctness for fixes deduced. We have extended the chase underlying the framework, and verified its Church-Rosser property. We have also settled the fundamental problem associated with the framework, having complexity the same as or slightly higher than their CFD counterparts.

To put the framework in action, there is naturally much more to be done. First, efficient algorithms need to be developed for discovering useful REEs from real-life data. Second, parallel algorithms are needed for implementing the chase, to scale with both the number and size of relational datasets. Moreover, parallel algorithms should be in place for incremental CR and ER in response to updates to the data. Such algorithms are already in place for CFDs and are being used in practice. Hence practical algorithms for REEs are also within the reach given that REEs bear complexity comparable to CFDs.

Acknowledgements This work was supported in part by Shenzhen Institute of Computing Sciences, Beijing Advanced Innovation Center for Big Data and Brain Computing (Beihang University), Royal Society Wolfson Research Merit Award (Grant No. WRM/R1/180014), European Research Council (Grant No. 652976), Engineering and Physical Sciences Research Council (Grant No. EP/M025268/1).

References

1 Wikibon. A comprehensive list of big data statistics, 2012. http://wikibon.org/blog/big-data-statistics/

2 Fan W F, Gao H, Jia X B, et al. Dynamic constraints for record matching. VLDB J, 2011, 20: 495–520

3 Bertossi L, Kolahi S, Lakshmanan L V S. Data cleaning and query answering with matching dependencies and matching functions. Theory Comput Syst, 2013, 52: 441–482
4 Bhattacharya I, Getoor L. Collective entity resolution in relational data. ACM Trans Knowl Discov Data, 2007, 1: 5
5 Arasu A, Ré C, Suciu D. Large-scale deduplication with constraints using Dedupalog. In: Proceedings of the 25th International Conference on Data Engineering, 2009
6 Mudgal S, Li H, Rekatsinas T, et al. Deep learning for entity matching: a design space exploration. In: Proceedings of International Conference on Management of Data, 2018
7 Arasu A, Götz M, Kaushik R. On active learning of record matching packages. In: Proceedings of International Conference on Management of Data, 2010
8 Fan W F, Geerts F, Jia X B, et al. Conditional functional dependencies for capturing data inconsistencies. ACM Trans Database Syst, 2008, 33: 1–48
9 Golab L, Karloff H, Korn F, et al. On generating near-optimal tableaux for conditional functional dependencies. In: Proceedings of the VLDB Endowment, 2008
10 Fan W F, Geerts F, Tang N, et al. Conflict resolution with data currency and consistency. J Data Inf Qual, 2014, 5: 1–37
11 Arenas M, Bertossi L, Chomicki J. Consistent query answers in inconsistent databases. In: Proceedings of Symposium on Principles of Database Systems, 1999
12 Chu X, Ilyas I F, Papotti P. Holistic data cleaning: putting violations into context. In: Proceedings of IEEE International Conference on Data Engineering, 2013
13 Chiticariu L, Li Y Y, Reiss F R. Rule-based information extraction is dead! Long live rule-based information extraction systems! In: Proceedings of Empirical Methods in Natural Language Processing, 2013
14 Fan W F, Li J Z, Ma S, et al. Interaction between record matching and data repairing. In: Proceedings of International Conference on Management of Data, 2011
15 Dong X, Halevy A, Madhavan J. Reference reconciliation in complex information spaces. In: Proceedings of International Conference on Management of Data, 2005
16 Whang S E, Benjelloun O, Garcia-Molina H. Generic entity resolution with negative rules. VLDB J, 2009, 18: 1261–1277
17 Sadri F, Ullman J D. The interaction between functional dependencies and template dependencies. In: Proceedings of International Conference on Management of Data, 1980
18 Bahmani Z, Bertossi L, Vasiloglou N. ERBlox: combining matching dependencies with machine learning for entity resolution. Int J Approx Reason, 2017, 83: 118–141
19 Whang S E, Garcia-Molina H. Joint entity resolution on multiple datasets. VLDB J, 2013, 22: 773–795
20 Verroios V, Garcia-Molina H, Papakonstantinou Y. Waldo: an adaptive human interface for crowd entity resolution. In: Proceedings of International Conference on Management of Data, 2017
21 Firmani D, Saha B, Srivastava D. Online entity resolution using an Oracle. Proc VLDB Endow, 2016, 9: 384–395
22 Ebraheem M, Thirumuruganathan S, Joty S, et al. Distributed representations of tuples for entity resolution. In: Proceedings of Very Large Data Bases, 2018
23 Qian K, Popa L, Sen P. Active learning for large-scale entity resolution. In: Proceedings of Conference on Information and Knowledge Management, 2017
24 Zhang D X, Guo L, He X N, et al. A graph-theoretic fusion framework for unsupervised entity resolution. In: Proceedings of the 34th International Conference on Data Engineering, 2018
25 Yakout M, Elmagarmid A K, Neville J, et al. Guided data repair. In: Proceedings of Very Large Data Bases, 2011
26 He J, Veltri E, Santoro D, et al. Interactive and deterministic data cleaning. In: Proceedings of International Conference on Management of Data, 2016
27 Assadi A, Milo T, Novgorodov S. Dance: data cleaning with constraints and experts. In: Proceedings of International Conference on Data Engineering, 2017
28 Guo S T, Dong X L, Srivastava D, et al. Record linkage with uniqueness constraints and erroneous values. In: Proceedings of Very Large Data Bases, 2010
29 Fan W F, Li J Z, Ma S, et al. Towards certain fixes with editing rules and master data. VLDB J, 2012, 21: 213–238
30 Fan W F, Lu P, Tian C, et al. Deducing certain fixes to graphs. Proc VLDB Endow, 2019, 12: 752–765
31 Yakout M, Berti-Équille L, Elmagarmid A K. Don't be scared: use scalable automatic repairing with maximal likelihood and bounded changes. In: Proceedings of International Conference on Management of Data, 2013. 553–564
32 Abiteboul S, Hull R, Vianu V. Foundations of Databases. Reading: Addison-Wesley, 1995
33 Aires J P, Meneguzzi F. Norm conflict identification using deep learning. In: Proceedings of International Conference on Autonomous Agents and Multiagent Systems, 2017. 194–207
34 Sycara K P. Machine learning for intelligent support of conflict resolution. Decision Support Syst, 1993, 10: 121–136

35 Loshin D. Master Data Management. San Francisco: Knowledge Integrity Inc., 2009
36 Chandra A K, Merlin P M. Optimal implementation of conjunctive queries in relational data bases. In: Proceedings of Symposium on the Theory of Computing, 1977
37 Aggarwal C C. Data Classification: Algorithms and Applications. Boca Raton: CRC Press, 2014
38 Fan W F, Geerts F. Foundations of Data Quality Management. San Rafael: Morgan & Claypool Publishers, 2012
39 Klug A. On conjunctive queries containing inequalities. J ACM, 1988, 35: 146–160
40 Baudinet M, Chomicki J, Wolper P. Constraint-generating dependencies. J Comput Syst Sci, 1999, 59: 94–115
41 Beeri C, Bernstein P A. Computational problems related to the design of normal form relational schemas. ACM Trans Database Syst, 1979, 4: 30–59
42 Rutenburg V. Complexity of generalized graph coloring. In: Proceedings of International Symposium on Mathematical Foundations of Computer Science, 1986
43 Schaefer M, Umans C. Completeness in the polynomial-time hierarchy: a compendium. 2002. http://ovid.cs.depaul.edu/documents/phcom.pdf

Profile of Wenfei FAN

Professor Wenfei Fan is the chair of web data management at the University of Edinburgh, UK, the chief scientist of Shenzhen Institute of Computing Science, and a chief scientist of Beijing Advanced Innovation Center for Big Data and Brain Computing, China. He received his Ph.D. from the University of Pennsylvania (USA), and his MS.c. and BS.c. from Peking University (China). He joined the University of Edinburgh in 2004; prior to that, he was a member of technical staff at Bell Laboratories in Murray Hill, NJ, USA.

He is a foreign member of Chinese Academy of Science, a fellow of the Royal Society (FRS), a fellow of the Royal Society of Edinburgh (FRSE), a member of the Academy of Europe (MAE), and an ACM Fellow (FACM). He is a recipient of Royal Society Wolfson Research Merit Award in 2018, ERC Advanced Fellowship in 2015, the Roger Needham Award in 2008 (UK), Yangtze River Scholar in 2007 (China), the Outstanding Overseas Young Scholar Award in 2003 (China), the Career Award in 2001 (USA), and several Test-of-Time and Best Paper Awards (Alberto O. Mendelzon Test-of-Time Award of ACM PODS 2015 and 2010, Best Paper Awards for SIGMOD 2017, VLDB 2010, ICDE 2007, and Computer Networks 2002).

Prof. Fan "has made fundamental contributions to both theory and practice of data management. He has both formalized the problems of querying big data and has developed radically new techniques that overcome the limits associated with conventional database systems. In addition, he has made seminal contributions to data quality, in which he devised new techniques for data cleaning that have found wide commercial adoption. He has also contributed to our understanding of semi-structured data" (cf. the Royal Society, UK). His current research interests include database theory and systems, in particular big data, data quality, data sharing, distributed computation, query languages, and social media marketing.

Querying big data. Dealing with massive data collections introduces fundamental challenges to database systems. Prof. Fan initiated formal approaches to querying big data.

(1) BD-tractability. Prof. Fan was one of the first to observe that querying big data requires a departure from classical computational complexity theory and the conventional query evaluation paradigms. That is, polynomial-time problems, which are called "tractable" in classical complexity theory, can no longer be considered tractable on big data. He revised the theory of tractability and proposed BD-tractability, a class of tractable queries in big data.

(2) A new parallel model. Graphs are a major source of big data, such as social networks and knowledge bases. To query big graphs, parallel computation is often a must. However, parallel algorithms are hard to write, debug and analyze.

Prof. Fan proposed a parallel model for graph computations based on simultaneous fixpoint computation with partial and incremental evaluation. It allows one to re-use existing serial graph algorithms as a whole, and it automatically parallelizes the computation. Under a monotone condition, the parallelized computation is guaranteed to converge at a correct answers as long as the serial algorithms are correct. Better yet, the ease of programming often comes with performance improvement. The work received three awards from SIGMOD 2017, VLDB 2017, and SIGMOD 2018. GRAPE, a parallel graph system based on the model, is being used at Alibaba Group and has proven effective in various applications.

(3) A new query paradigm. Parallel processing of big data requires necessary hardware that few organizations can afford.

Prof. Fan approached the problem by proposing a theory of bounded evaluation. The idea is to reduce queries on big data to computations on small data. It was shown that with the right auxiliary structures, many queries are boundedly evaluable, i.e., they can be answered using a fixed fraction of the total data. To handle queries that are not bounded, he introduced a data-driven approximation scheme, which uses the available resources to compute a set of approximate answers such that each approximate answer is within a fixed distance of an exact answer and conversely, each exact answer is within a fixed distance of an approximate answer. Putting these together, Prof. Fan proposed a new paradigm for querying big data under limited resources. This promises to provide small companies with the capability of big data analytics. The work received Royal Society Wolfson Research Merit Award in 2018.

As proof of concept, a world-class telecommunication company found that 90% of their queries are boundedly evaluable, and that bounded evaluation reduces big datasets from petabytes to gigabytes in many cases, improving performance by orders of magnitude.

Data quality. Together with volume, quality (veracity) is often considered the most important problem in data management. The cost of bad data is huge; some estimates put the annual cost to the USA alone into the trillions[1)]. While the study of data quality goes back to 1960s, it was Prof. Fan's work starting in 2005 that reshaped the field. There are five central issues: consistency, accuracy, completeness, timeliness, and entity resolution. Prof. Fan's work had substantial impact on each of these, from theory to practice.

(1) Theory. Prof. Fan provided models, characterizations and complexity for each of the five issues. For example, to catch semantic inconsistencies, he defined what is

1) Harvard Business Review. https://hbr.org/2016/09/bad-data-costs-the-u-s-3-trillion-per-year.

now called the "standard class" of conditional dependencies that extends classical dependency theory in relational databases. To cope with missing data, he introduced a notion of relative information completeness, which revises the conventional closed world and open world assumptions. He also proposed a uniform logical framework to reason about the interaction of the five issues of data quality.

(2) Practical techniques. Prof. Fan's team transferred their theoretical work to technology, by developing key algorithms for discovering data quality rules, detecting errors by employing the rules, and repairing the data by fixing the errors with certainty. These yield a package of techniques for cleaning real-life data. The combination of theory and practice received best paper awards from two of the leading database systems conferences (VLDB and ICDE) and the British Computer Society's Roger Needham Award, as well as awards from industry (IBM, Google, and Yahoo).

Semi-structured data. Prof. Fan was also recognized for opening up the field of integrity constraints for semi-structured data, and making contributions to Web data management.

(1) Constraints. Earlier in his career, Prof. Fan developed the field of integrity constraints for XML, now a mature area well represented in the full spectrum of database research, from theory to practice and standards. This work won him his first test-of-time award for PODS 2010. More recently Prof. Fan studied constraints for graphs, as a combination of topological constraints and value dependencies. Prof. Fan developed axiom systems, complexity and algorithms for reasoning about such constraints. These find applications in knowledge acquisition, knowledge base enrichment, and fraud detection.

(2) Web data management. Beyond constraints, Prof. Fan has also worked on a variety of topics for Web data management, including (a) query languages, (b) transformations between Web data and relations, (c) integration of Web data, (d) securing XML queries, and (e) graph association rules for social media marketing. Prof. Fan made contributions to each and every of these topics, from theory to practice. His work on query languages earned him his second test-of-time award from PODS and is the standard reference work on this topic.

Selected publications. Following conventions in the subject, authors are usually listed alphabetically.

• Fan W F, Lu P. Dependencies for graphs. ACM Trans Database Syst, 2019, 44: 5 (invited extension of a PODS 2017 paper)

• Fan W F, Yu W Y, Xu J B, et al. Parallelizing sequential graph computations. ACM Trans Database Syst, 2018, 43: 18 (invited extension of an SIGMOD paper, which received the Best Paper Award for SIGMOD 2017)

• Fan W F, Cao Y, Xu J B, et al. From think parallel to think sequential. In: Proceedings of International Conference on Management of Data, 2018. 47: 15–22 (SIGMOD Research Highlight Award)

• Deng T, Fan W F, Geerts F. On the complexity of package recommendation problems. SIAM J Comput, 2013, 42: 1940–1986

• Fan W F, Geerts F, Neven F. Making queries tractable on big data with preprocessing. In: Proceedings of the 39th International Conference on Very Large Data Bases (VLDB), 2013. 6: 47–481

• Fan W F, Geerts F. Foundations of Data Quality Management. San Rafael: Morgan & Claypool Publishers, 2012

• Fan W F, Geerts F, Wijsen J. Determining the currency of data. ACM Trans Database Syst, 2012, 37: 25 (invited extension of a PODS 2011 paper)

• Fan W F, Li J Z, Ma S, et al. Towards certain fixes with editing rules and master data. VLDB J, 2012, 21: 213–238 (invited extension of a VLDB paper, which received the Best Paper Award for VLDB 2010)

• Fan W F, Geerts F. Relative information completeness. ACM Trans Database Syst, 2010, 35: 27 (invited extension of a PODS 2009 paper)

• Fan W F, Geerts F, Neven F. Expressiveness and complexity of XML publishing transducers. ACM Trans Database Syst, 2010, 33: 25 (invited extension of a PODS 2008 paper)

• Fan W F, Geerts F, Ji X B, et al. Conditional functional dependencies for capturing data inconsistencies. ACM Trans Database Syst, 2008, 33: 6 (its conference version "Conditional functional dependencies for data cleaning" received the Best Paper Award for ICDE 2007)

• Fan W F. Dependencies revisited for improving data quality. In: Proceedings of Symposium on Principles of Database Systems, 2008. 159–170 (invited paper)

• Benedikt M, Fan W F, Geerts F. XPath satisfiability in the presence of DTDs. J ACM, 2008, 55: 8: (invited extension of a PODS 2005 paper, which received the PODS Test-of-Time Award in 2015)

• Fan W F, Siméon J. Integrity constraints for XML. J Comput Syst Sci, 2003, 66: 254–291 (invited extension from a PODS 2000 paper, which received the PODS Test-of-Time Award in 2010)

• Buneman P, Davidson S, Fan W F, et al. Keys for XML. Comput Netw, 2002, 39: 473–487 (invited extension of a WWW 2001 paper; the Best Paper of the Year Award from Computer Networks in 2002)

• Fan W F, Libkin L. On XML integrity constraints in the presence of DTDs. J ACM, 2002, 49: 368–406