Inferring Data Currency and Consistency for Conflict Resolution

Wenfei Fan[1] Floris Geerts[2] Nan Tang[3] Wenyuan Yu[1]

[1]*University of Edinburgh* [2]*University of Antwerp* [3]*QCRI, Qatar Foundation*

wenfei@inf.ed.ac.uk floris.geerts@ua.ac.be ntang@qf.org.qa wenyuan.yu@ed.ac.uk

***Abstract*— This paper introduces a new approach for conflict resolution: given a set of tuples pertaining to the same entity, it is to identify a single tuple in which each attribute has *the latest and consistent* value in the set. This problem is important in data integration, data cleaning and query answering. It is, however, challenging since in practice, reliable timestamps are often absent, among other things. We propose a model for conflict resolution, by specifying data currency in terms of partial currency orders and currency constraints, and by enforcing data consistency with constant conditional functional dependencies. We show that identifying data currency orders helps us repair inconsistent data, and vice versa. We investigate a number of fundamental problems associated with conflict resolution, and establish their complexity. In addition, we introduce a framework and develop algorithms for conflict resolution, by integrating data currency and consistency inferences into a single process, and by interacting with users. We experimentally verify the accuracy and efficiency of our methods using real-life and synthetic data.**

I. INTRODUCTION

Conflict resolution is the process that, given a set I_t of tuples pertaining to the same entity, fuses the tuples into a single tuple and resolves conflicts among the tuples of I_t [10]. Traditional work resolves conflicts typically by taking, *e.g.,* the max, min, avg, any of attribute values (see [4] for a survey).

We study a new approach for conflict resolution, by highlighting both data currency and data consistency. Given I_t, it is to identify a single tuple in which each attribute has *consistent* and *the most current value* taken from I_t, referred to as the *true values* of the entity *relative to* I_t. The need for studying this problem is evident in data integration, where conflicts often emerge from values from different sources. It is also common to find multiple values of the same entity residing in a database. While these values were *once correct*, *i.e.,* they were the true values of the entity at some time, some of them may have become *stale* and thus *inconsistent*. Indeed, it is estimated that in a customer database, about 50% of the records may become obsolete within two years [11]. With these comes the need for resolving conflicts for, *e.g.,* data fusion [4], [10], data cleaning [1] and query answering with current values [15].

No matter how important, the problem is rather challenging. Indeed, it is already highly nontrivial to find consistent values for an entity [1], [7]. Moreover, it is hard to identify the most current entity values [15] since in the real world, reliable timestamps are often absent [29]. Add to this the complication that when resolving conflicts one has to find the entity values that are *both* consistent *and* most current.

Example 1: The photo in Fig. 1 is known as "V-J Day in Times Square". The nurse and sailor in the photo have been identified as Edith Shain and George Mendonça, respectively, and their information is collected in sets E_1 and E_2 of tuples, respectively, shown in Fig. 2.

We want to find the true values of these entities, *i.e.,* a tuple t_1 for Edith (resp. a tuple t_2 for George) such that the tuple has the most current and consistent attribute values for her (resp. his) status, job, the number of kids, city, AC (area code), zip and county in E_1 (resp. E_2). However, the values in E_1 (E_2) have conflicts, and worse still, they do not carry timestamps. They do not tell us, for instance, whether Edith still lives in *NY*, or even whether she is still alive. □

The situation is bad, but not hopeless. We can often deduce certain currency orders from the semantics of the data. In addition, dependencies such as conditional functional dependencies (CFDs) [13] have proven useful in improving the consistency of the data. Better still, data currency and consistency interact with each other. When they are taken together, we can often infer some true values from inconsistent tuples, even in the absence of timestamps, as illustrated below.

Example 2: From the semantics of the data, we can deduce the *currency constraints* and CFDs shown in Fig. 3.

(1) Currency constraints. We know that for each person, status only changes from *working* to *retired* and from *retired* to *deceased*, but not from *deceased* to *working* or *retired*. These can be expressed as φ_1 and φ_2 given in Fig. 3, referred to as *currency constraints*. Here $t_1 \prec_{\text{status}} t_2$ denotes a partial currency order on the attribute status, indicating that t_2 is *more current* than t_1 *in attribute* status. Similarly, we know that job can only change from *sailor* to *veteran* but not the other way around. We can express this as currency constraint φ_3, shown in Fig. 3. Moreover, the number of kids typically increases monotonically. We can express this as φ_4, assuring that t_2 is more current than t_1 in attribute kids if $t_1[\text{kids}] < t_2[\text{kids}]$.

In addition, we know that for each person, if tuple t_2 is more current than t_1 in attribute status, then t_2 is also more current than t_1 in job, AC and zip. Furthermore, if t_2 is more current than t_1 in attributes city and zip, it also has more current county than t_1. These can be expressed as φ_5–φ_8.

(2) *Constant CFDs*. In the US, if the AC is *213* (resp. *212*), then the city must be *LA* (resp. *NY*). These are expressed as conditional functional dependencies ψ_1 and ψ_2 in Fig. 3.

We can apply these constraints to E_1 in Fig. 2, to improve the currency and consistency of the data. By *interleaving* inferences of data currency and consistency, we can actually identify the true values of entity Edith, as follows:

Fig. 1. V-J Day

		name	status	job	kids	city	AC	zip	county
E_1	r_1:	*Edith Shain*	*working*	*nurse*	*0*	*NY*	*212*	*10036*	*Manhattan*
	r_2:	*Edith Shain*	*retired*	*n/a*	*3*	*SFC*	*415*	*94924*	*Dogtown*
	r_3:	*Edith Shain*	*deceased*	*n/a*	null	*LA*	*213*	*90058*	*Vermont*
E_2	r_4:	*George Mendonça*	*working*	*sailor*	*0*	*Newport*	*401*	*02840*	*Rhode Island*
	r_5:	*George Mendonça*	*retired*	*veteran*	*2*	*NY*	*212*	*12404*	*Accord*
	r_6:	*George Mendonça*	*unemployed*	*n/a*	*2*	*Chicago*	*312*	*60653*	*Bronzeville*

Fig. 2. Instances E_1 for entity Edith and E_2 for George

Currency constraints:	φ_1: $\forall t_1, t_2$ (t_1[status] = "working" $\wedge$ t_2[status] = "retired" $\rightarrow t_1 \prec_{\text{status}} t_2$)
	φ_2: $\forall t_1, t_2$ (t_1[status] = "retired" $\wedge$ t_2[status] = "deceased" $\rightarrow t_1 \prec_{\text{status}} t_2$)
	φ_3: $\forall t_1, t_2$ (t_1[job] = "sailor" $\wedge$ t_2[job] = "veteran" $\rightarrow t_1 \prec_{\text{job}} t_2$)
	φ_4: $\forall t_1, t_2$ ($t_1[\text{kids}] < t_2[\text{kids}] \rightarrow t_1 \prec_{\text{kids}} t_2$) φ_5: $\forall t_1, t_2$ ($t_1 \prec_{\text{status}} t_2 \rightarrow t_1 \prec_{\text{job}} t_2$)
	φ_6: $\forall t_1, t_2$ ($t_1 \prec_{\text{status}} t_2 \rightarrow t_1 \prec_{\text{AC}} t_2$) φ_7: $\forall t_1, t_2$ ($t_1 \prec_{\text{status}} t_2 \rightarrow t_1 \prec_{\text{zip}} t_2$)
	φ_8: $\forall t_1, t_2$ ($t_1 \prec_{\text{city}} t_2 \wedge t_1 \prec_{\text{zip}} t_2 \rightarrow t_1 \prec_{\text{county}} t_2$)
Constant CFDs:	ψ_1 : (AC = *213* $\rightarrow$ city = *LA*); ψ_2 : (AC = *212* $\rightarrow$ city = *NY*);

Fig. 3. Currency constraints and constant CFDs

(a) from the currency constraints φ_1 and φ_2, we can conclude that her latest status is *deceased*;
(b) similarly, by φ_4, we find that her true kids value is *3* (assuming null $< k$ for any number k);
(c) from (a) above and φ_5–φ_7, we know that her latest job, AC and zip are *n/a*, *213* and *90058*, respectively;
(d) after currency inferences (a) and (c), we can apply the CFD ψ_1 and find her latest city as *LA*; and
(e) after the consistency inference (d), from (c) and (d) we get her latest county as *Vermont*, by applying φ_8.

Now we have identified a single tuple t_1 = (Edith Shain, deceased, n/a, 3, LA, 213, 90085, Vermont) as the true values of Edith in E_1 (the address is for her cemetery). □

This example suggests the following. (1) Data currency and consistency should be interleaved when resolving conflicts. Indeed, not only deducing currency orders helps us *improve the consistency* (*e.g.,* from steps $(a), (c)$ to (d)), but data consistency inferences also help us *identify the most current values* (*e.g.,* step (e) is doable only after (d)). (2) Both data currency and consistency can be specified with constraints, and hence, can be processed in a uniform logical framework.

While the need for deducing the consistent and most current values has been advocated for conflict resolution [10], [21], prior work typically assumes the availability of timestamps. Previous work on data quality focuses on either data consistency (*e.g.,* [1], [7], [13], [27]) or data currency (*e.g.,* [15]). However, no models or algorithms are yet in place to combine data consistency and currency for *conflict resolution*.

Contributions. We propose to study conflict resolution by inferring *both* data currency *and* data consistency.

(1) We propose a model for conflict resolution (Section II). We specify data currency in terms of (a) *partial currency orders* denoting available (yet possibly incomplete) temporal information on the data, and (b) simple *currency constraints*, to express currency relationships derived from the semantics of the data. Data consistency is specified in terms of *constant CFDs* [13] on the latest values of the data. Given such a specification S_e on a set E of tuples pertaining to the same entity e, we aim to derive the true values of e from S_e.

(2) We introduce a framework for conflict resolution (Section III). One may find *some* true values of an entity from a specification of an entity, but *not all*, as illustrated below.

Example 3: Consider the set E_2 of tuples for entity George Mendonça (Fig. 2). Along the same lines as Example 2, we find that its true (name, kids) values are (*George Mendonça*, *2*). However, we do not have sufficient information to infer the true values of the other attributes. □

In light of this, our framework *automatically derives* as many true values as possible from a given specification S_e of an entity e, identifies attributes for which the true values of e are not derivable from S_e, and *interacts* with users to solicit additional input for those attributes, so that all the true values of e can be derived from S_e and users' input.

(3) We study problems fundamental to conflict resolution (Section IV). Given a specification S_e, we determine whether partial currency orders, currency constraints and CFDs in S_e have conflicts among themselves? Whether some other currency orders are implied by S_e? Whether true values of an entity can be derived from S_e? If not, what additional minimum currency information has to be provided so that the true values are derivable? We establish their complexity bounds, ranging from NP-complete and coNP-complete to Σ_2^p-complete. These results reveal the complexity *inherent to* conflict resolution.

(4) We develop several practical algorithms (Section V). We propose methods for finding (a) whether a specification S_e has conflicts, (b) what true values can be derived from S_e, and (c) a minimum set of attributes that require users' input to find their true values. All these problems are *intractable*; in particular, the last problem is Σ_2^p-complete. Nevertheless, we provide efficient heuristic algorithms, by integrating inferences of data consistency and currency into a single process.

(5) We evaluate the accuracy and efficiency of our method using real-life and synthetic data (Section VI). We find that *unifying* currency and consistency *substantially improves* the accuracy of traditional methods, by 201% (F-measure).

We contend that this work provides fundamental results for conflict resolution, and proposes a practical solution via data currency and consistency in the *absence* of timestamps.

Related work. Conflict resolution has been studied for decades, started from [8]. It aims to combine data from different sources into a single representation (see [4], [10] for surveys). In that context, inconsistencies are typically resolved by selecting the max, min, avg, any value [4]. While the need for current values was also observed there [10], [21], they are identified only by using *timestamps*. This work differs from the traditional work in the following. (1) We revise the conflict resolution problem to identify values of entities that are both *consistent* and most *current*. (2) We *do not* assume the availability of timestamps, which are often missing in practice [29]. (3) We resolve conflicts by using currency constraints and CFDs [1], [7], [13], instead of picking max, min, avg or any value. (4) We employ *automated reasoning* to identify true values by unifying the inferences of currency and consistency.

There has been work on truth discovery from data sources [9], [17], [28]. Their approaches include (1) vote counting and probabilistic computation based on the trustworthiness of data sources [17], [28]; (2) source dependencies to find copy relationships and reliable sources [9]; and (3) employing lineage information and probabilities [26]. In contrast, we assume no information about the accuracy of data sources, but derive true values based on data currency and consistency. In addition, we adopt a logical approach via automated reasoning about constraints, as opposed to probabilistic computation. This work is complementary to the previous work.

This work extends [13], [15]. A data currency model was presented in [15] with partial currency orders and denial constraints [1]. CFDs were studied for specifying data consistency [13]. This work differs from [13], [15] in the following. (1) We propose a conflict resolution model that combines data currency and consistency. In contrast, [15] only studies data currency, while [13] only considers data consistency. (2) We *interleave* inferences of data currency and consistency, which is far more intriguing than handling currency and consistency separately, and requires new techniques to capture the interaction between the two. (3) We use currency constraints, which are simpler than denial constraints, to strike a balance between the complexity of inferring true values and the expressivity needed for specifying currency (Section IV). (4) *No practical algorithms* were given in [15] for deriving current values.

Previous work on data consistency [1], [7], [13], [19], [27] has been focusing on consistent query answering and data repairing [2], topics different from conflict resolution. The study of preferred repairs [19] also advocates partial orders. It differs from the currency orders we study here in that they use PTIME functions to rank different repairs over the entire database, whereas we derive the currency orders by automated *reasoning about both* available partial temporal information and currency constraints. Preferred repairs are implemented by [7] via a cost metric, and by [27] based on a decision theory, which can be incorporated into our framework.

There has also been a large body of work on temporal databases (see [6], [24] for surveys). In contrast to that line of work, we do not assume the availability of timestamps.

It has recently been shown that temporal information helps record linkage identify records that refer to the same entity [20]. Here we show that data currency also helps conflict resolution, a different process that takes place *after* record linkage has identified tuples pertaining to the same entity. While [20] is based on timestamps, we do not assume it here.

II. A Conflict Resolution Model

We now introduce our conflict resolution model. We start with currency (Section II-A) and consistency (Section II-B) specifications. We then present the model (Section II-C).

A. Data Currency

We specify the currency of data by means of (a) partial currency orders, and (b) currency constraints.

Data with partial currency orders. Consider a relation schema $R = (A_1, \ldots, A_n)$, where each attribute A_i has a domain $\mathsf{dom}(A_i)$. In this work we focus on *entity instances* I_e of R, which are sets of tuples of R *all* pertaining to the *same* real-world entity e, and are typically much smaller than a database instance. Such entity instances can be identified by *e.g.,* record linkage techniques (see [12] for a survey).

For an attribute $A_i \in R$ and an entity instance I_e of R, we denote by $\mathsf{adom}(I_e.A_i)$ the set of A_i-attribute values that occur in I_e, referred to as *the active domain of* A_i *in* I_e.

For example, two entity instances are given in Fig. 2: $E_1 = \{r_1, r_2, r_3\}$ for entity "Edith", and $E_2 = \{r_4, r_5, r_6\}$ for "George"; and $\mathsf{adom}(E_1.\mathsf{city})$ = {*NY, SFC, LA*}.

A *temporal instance* I_t of I_e is given as $(I_e, \preccurlyeq_{A_1}, \ldots, \preccurlyeq_{A_n})$, where each $\preccurlyeq_{A_i}$ is a partial order on I_e, referred to as the *currency order for attribute* A_i for the entity represented by I_e. For $t_1, t_2 \in I_e$, $t_1 \preccurlyeq_{A_i} t_2$ if and only if (iff) either t_1 and t_2 share the same A_i-attribute value (*i.e.,* $t_1[A_i] = t_2[A_i]$), or that $t_2[A_i]$ is more current than $t_1[A_i]$ (denoted by $t_1 \prec_{A_i} t_2$).

Intuitively, currency orders represent *available* temporal information about the data. Observe that $\preccurlyeq_{A_i}$ is a *partial order*, possibly empty. For example, for E_1 above, we only know that $r_3 \preccurlyeq_{\mathsf{kids}} r_1$ and $r_3 \preccurlyeq_{\mathsf{kids}} r_2$ since $r_3[\mathsf{kids}]$ is null, which are in the currency order $\preccurlyeq_{\mathsf{kids}}$, while the currency orders for other attributes are empty, excluding the case when tuples carry the same attribute value. Similarly for E_2. In particular, $t_1 \preccurlyeq_{A_i} t_2$ if $t_1[A_i]$ is null, *i.e.,* an attribute with value missing is ranked the lowest in the currency order.

Current instances. Currency orders are often incomplete. Hence we consider possible completions of currency orders.

A *completion* I_t^c of I_t is a temporal instance $I_t^c = (I_e, \preccurlyeq^c_{A_1}, \ldots, \preccurlyeq^c_{A_n})$, such that for each $i \in [1, n]$, (1) $\preccurlyeq_{A_i} \subseteq \preccurlyeq^c_{A_i}$, and (2) for all tuples $t_1, t_2 \in I_e$, either $t_1 \preccurlyeq^c_{A_i} t_2$ or $t_2 \preccurlyeq^c_{A_i} t_1$. That is, $\preccurlyeq^c_{A_i}$ induces a *total order* on tuples in $I + e$.

That is, I_t^c totally sorts the attribute values in I_e such that the most current value of each attribute is the last in the order.

We define *the most current* A_i*-attribute value of* I_t^c to be $t[A_i]$ that comes last in the total order $\preccurlyeq^c_{A_i}$. The *current tuple* of I_t^c, denoted by $\mathsf{LST}(I_t^c)$ (*i.e.,* last), is the tuple t_l such that for each attribute A_i, $t_l[A_i]$ is the most current A_i-value of I_t^c, *i.e.,* t_l contains the most current values from I_t^c.

Currency constraints. One can derive additional currency information from the semantics of the data, which is modeled as *currency constraints*. A currency constraint φ is of the form

$$\forall t_1, t_2\ (\omega \rightarrow t_1 \prec_{A_r} t_2),$$

where ω is a conjunction of predicates of the form: (1) $t_1 \prec_{A_l} t_2$, *i.e.,* t_2 is more current than t_1 in attribute A_l; (2)

$t_1[A_l]$ op $t_2[A_l]$, where op is one of $=, \neq, >, <, \leq, \geq$; and (3) $t_i[A_l]$ op c for $i \in \{1, 2\}$, where c is a constant.

In contrast to denial constraints in the model of [15], currency constraints are defined on two tuples, like functional dependencies. Such constraints suffice to specify currency information commonly found in practice (see, *e.g.,* Example 2).

Currency constraints are interpreted over completions I_t^c of I_t. We say that I_t^c *satisfies* φ, denoted by $I_t^c \models \varphi$, if for any two tuples t_1, t_2 in I_e, if these tuples and related order information in I_t^c satisfy the predicates in ω, following the standard semantics of first-order logic, then $t_1 \prec^c_{A_r} t_2$.

We say that I_t^c satisfies a set Σ of currency constraints, denoted by $I_t^c \models \Sigma$, if $I_t^c \models \varphi$ for all $\varphi \in \Sigma$.

Example 4: Recall the entity instances E_1 and E_2 given in Fig. 2. Currency constraints on these instances include φ_1–φ_8 as specified in Fig. 3 and interpreted in Example 2.

It is readily verified that for any completion E_1^c of E_1, if it satisfies these constraints, it yields $\mathsf{LST}(E_1^c)$ of the form (*Edith*, *deceased*, *n/a*, *3*, x_{city}, *213*, *90058*, x_{county}) for Edith, in which the most current values for attributes name, status, job, kids, AC and zip are deduced from the constraints and remain unchanged, while x_{city} and x_{county} are values determined by the total currency order given in E_1^c. Observe that the values of the current tuple are taken from *different tuples* in E_1, *e.g.,* kids = *3* from r_2 and AC = *213* from r_3.

Similarly, for any completion of E_2, its current tuple has the form (*George*, $x_{\mathsf{status}}, x_{\mathsf{job}}$, *2*, $x_{\mathsf{city}}, x_{\mathsf{AC}}, x_{\mathsf{zip}}, x_{\mathsf{county}}$), if they satisfy all constraints. Hence, currency constraints help us find some but not *all* of the most current values of entities. $\square$

B. Data Consistency

To specify the consistency of data, we use a simple class of conditional functional dependencies (CFDs) [13] as follows.

A *constant* CFD [13] ψ on a relation schema R is of the form $t_p[X] \rightarrow t_p[B]$, where (1) $X \subseteq R$, $B \in R$; and (2) t_p is the *pattern tuple* of ψ with attributes in X and B, where for each A in $X \cup \{B\}$, $t_p[A]$ is a *constant* in $\mathsf{dom}(A)$ of A.

For example, ψ_1 and ψ_2 in Table 3 are constant CFDs on the relation of Table 2, as interpreted in Example 2.

Such CFDs are defined on *the current tuple* of a completion. Consider a completion I_t^c of I_t and let $t_l = \mathsf{LST}(I_t^c)$ be the current tuple of I_t^c. We say that the completion I_t^c *satisfies* a constant CFD $\psi = t_p[X] \rightarrow t_p[B]$, denoted by $I_t^c \vDash \psi$, iff when $t_l[X] = t_p[X]$ then $t_l[B] = t_p[B]$.

Intuitively, this assures that if $t_l[X] = t_p[X]$ and if $t_l[X]$ contains the most current X-attribute values, then $t_l[B]$ can be *repaired* by taking the value $t_p[B]$ in the pattern, and moreover, $t_l[B]$ is the most current value in attribute B.

We say that I_t^c *satisfies* a set Γ of constant CFDs, denoted as $I_t^c \vDash \Gamma$, iff $I_t^c \vDash \psi$ for each $\psi \in \Gamma$.

Observe that a constant CFD is defined on a *single tuple* $\mathsf{LST}(I_t^c)$. In light of this, we do not need general CFDs of [13] here, which are typically defined on *two tuples*.

Example 5: Recall the current tuples for E_1 in Example 4. Then all completions of E_1 that satisfy ψ_1 in Fig. 3 have the form (*Edith*, *deceased*, *n/a*, *3*, *LA*, *213*, *90058*, *Vermont*), in which x_{city} is instantiated as *LA* by ψ_1, and as a result, x_{county} becomes *Vermont* by the currency constraint φ_8. $\square$

C. Conflict Resolution

We are ready to bring currency and consistency together.

Specifications. A *specification* $S_e = (I_t, \Sigma, \Gamma)$ *of an entity* consists of (1) a temporal instance $I_t = (I_e, \preccurlyeq_{A_1}, \ldots, \preccurlyeq_{A_n})$; (2) a set Σ of currency constraints; and (3) a set Γ of constant CFDs. A completion $I_t^c = (I_e, \preccurlyeq^c_{A_1}, \ldots, \preccurlyeq^c_{A_n})$ of I_t is a *valid completion* of S_e if I_t^c satisfies both Σ and Γ. We say that S_e is *valid* if there exists a valid completion I_t^c of S_e, *e.g.,* the specification of E_1 (or E_2) and the constraints in Fig. 3 is valid.

True values. There may be many valid completions I_t^c, each leading to a possibly different current tuple $\mathsf{LST}(I_t^c)$. When two current tuples differ in some attribute, there is *a conflict*. We aim to resolve such conflicts. If all such current tuples agree on *all* attributes, then the specification is conflict-free, and a *unique* current tuple exists for the entity e specified by S_e. In this case, we say that this tuple is the true value of e.

More formally, the *true value* of S_e, denoted by $\mathsf{T}(S_e)$, is the *single* tuple t_c such that for *all valid* completions I^c of S_e, $t_c = \mathsf{LST}(S_e)$, if it exists. For each attribute A_i of R, we call $t_c[A_i]$ the *true value* of A_i in S_e.

The conflict resolution problem. Consider a specification $S_e = (I_t, \Sigma, \Gamma)$, where $I_t = (I_e, \preccurlyeq_{A_1}, \ldots, \preccurlyeq_{A_n})$. Given S_e, conflict resolution is to find the minimum amount of additional currency information such that the true value exists.

The additional currency information is specified in terms of a *partial temporal order* $O_t = (I, \preccurlyeq'_{A_1}, \ldots, \preccurlyeq'_{A_n})$. We use $S_e \oplus O_t$ to denote the extension $S'_e = (I'_t, \Sigma, \Gamma)$ of S_e by enriching I_t with O_t, where $I'_t = (I_e \cup I, \preccurlyeq_{A_1} \cup \preccurlyeq'_{A_1}, \ldots, \preccurlyeq_{A_n} \cup \preccurlyeq'_{A_n})$. We only consider partial temporal orders O_t such that $\preccurlyeq_{A_i} \cup \preccurlyeq'_{A_i}$ is a partial order for all $i \in [1, n]$.

We use $|O_t|$ to denote $\Sigma_{i \in [1,n]} |\preccurlyeq'_{A_i}|$, *i.e.,* the sum of the sizes of all the partial orders in O_t.

Given a valid specification $S_e = (I_t, \Sigma, \Gamma)$ of an entity, the *conflict resolution problem* is to find a partial temporal order O_t such that (a) $\mathsf{T}(S_e \oplus O_t)$ exists and (b) $|O_t|$ is minimum.

Example 6: Recall from Example 4 the current tuples for George. Except for name and kids, we do not have a unique current value for the other attributes. Nonetheless, if a partial temporal order O_t with, *e.g.,* $r_6 \prec_{\mathsf{status}} r_5$ is provided by the users (*i.e.,* status changes from *unemployed* to *retired*), then the true value of George in E_2 can be derived as (*George*, *retired*, *veteran*, *2*, *NY*, *212*, *12404*, *Accord*) from the currency constraints and CFDs of Fig. 3. $\square$

III. A Conflict Resolution Framework

We propose a framework for conflict resolution. As depicted in Fig. 4, given a specification $S_e = (I_t, \Sigma, \Gamma)$ of an entity e, the framework is to find the true value $\mathsf{T}(S_e)$ of e by reasoning about data currency and consistency, and by interacting with the users to solicit additional data currency information.

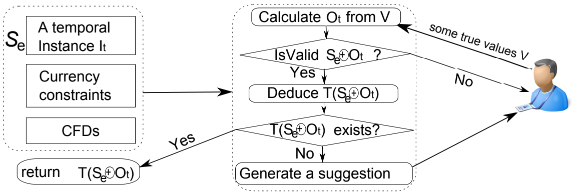

Fig. 4. Framework overview

The framework provides the users with suggestions. A *suggestion* is a minimum set $\mathcal{A}$ of attributes of e such that if the true values of these attributes are provided by the users, $\mathsf{T}(S_e)$ is automatically deduced from the users' input, Σ, Γ and I_t. The true values for $\mathcal{A}$ are represented as a temporal order O_t. More specifically, the framework works as follows.

(1) *Validity checking.* It first inspects whether $S_e \oplus O_t$ is valid, via automated reasoning, where O_t is a partial temporal order provided by the users (see step (4) below), *initially empty*. If so, it follows the 'Yes' branch. Otherwise the users need to revise O_t by following the 'No' branch.

(2) *True value deducing.* After $S_e \oplus O_t$ is validated, it derives as many true values as possible, via automated reasoning.

(3) *Finding the true value.* If $\mathsf{T}(S_e \oplus O_t)$ exists, it terminates and returns the true value, by following the 'Yes' branch. Otherwise, it follows the 'No' branch and goes to step (4).

(4) *Generating suggestions.* It computes a suggestion $\mathcal{A}$ along with its candidate values from the active domain of S_e, such that if the users pick and validate the true values for $\mathcal{A}$, then $\mathsf{T}(S_e \oplus O_t)$ is warranted to be found. The users are expected to provide V, the true values of *some attributes* in $\mathcal{A}$, represented as a partial temporal order O_t. Given O_t, $S_e \oplus O_t$ is constructed and the process goes back to step (1).

The process proceeds until $\mathsf{T}(S_e \oplus O_t)$ is found, or when the users opt to settle with true values for a subset of attributes of e. That is, if users do not have sufficient knowledge about the entity, they may let the system derive true values for as many attributes as possible, and revert to the traditional methods to pick the max, min, avg, any values for the rest of the attributes.

Remarks. (1) To specify users' input, let I_t in S_e be $(I_e, \preccurlyeq_{A_1}, \ldots, \preccurlyeq_{A_n})$ and $\mathcal{A} \cup \mathcal{A}' \cup \mathcal{B} = \{A_1, \ldots, A_n\}$, where (i) $\mathcal{A}$ is the set of attributes identified in step (4) for which the true values are unknown; (ii) for $\mathcal{B}$, their true values $\mathsf{V}_\mathcal{B}$ have been deduced (step (2)); and (iii) $\mathcal{A}'$ is the set of attributes whose true values can be deduced from $\mathsf{V}_\mathcal{B}$ and the suggestion for $\mathcal{A}$. Given a suggestion, the user is expected to provide a set V of true values for (a subset of) $\mathcal{A}$. Here V consists of either the candidate values from the suggestion, or some *new* values not in the active domains of S_e that users opt to choose. The users *do not* have to enter values for *all* attributes in $\mathcal{A}$.

From the input V, a partial temporal order O_t is automatically derived, by treating V as the most current values of those attributes involved. Indeed, O_t has the form $(I_e \cup \{t_o\}, \preccurlyeq'_{A_1}, \ldots, \preccurlyeq'_{A_n})$, where t_o is a new tuple such that for all attributes A, $t_o[A] = \mathsf{V}(A)$ if V has a value $\mathsf{V}(A)$ for A, and $t_o[A] = \mathsf{null}$ otherwise, while $t_o[\mathcal{B}] = \mathsf{V}_\mathcal{B}$ remains unchanged. Moreover, $\preccurlyeq'_A$ extends $\preccurlyeq_A$ by including $t[A] \preccurlyeq_A t_o[A]$ if $t_o[A] \neq \mathsf{null}$, for all tuples $t \in I_e$. Then $S_e \oplus O_t$ can be readily defined.

(2) There have been efficient methods for discovering constant CFDs, *e.g.,* [14]. Along the same lines as CFD discovery [5], [14], automated methods can be developed for discovering currency constraints from (possibly dirty) data. With certain quality metric in place [5], the constraints discovered can be as accurate as those manually designed (such as those given in Fig. 3), and can be used by the framework as input.

(3) To simplify the discussion we do not allow users to change constraints in S_e. We defer this issue to Section VII.

IV. Fundamental Problems

We next identify fundamental problems associated with conflict resolution based on both data currency and consistency, and establish their complexity. These results are not only of theoretical interest, but also tell us where the complexity arises, and hence guide us to develop effective (heuristic) algorithms.

Satisfiability. The *satisfiability problem* is to determine, given a specification $S_e = (I_t, \Sigma, \Gamma)$ of an entity, whether S_e is *valid*, *i.e.,* whether there exists a valid completion of S_e.

It is to check whether S_e makes sense, *i.e.,* whether the currency constraints, CFDs and partial orders in S_e, when put together, have conflicts themselves. The analysis is needed by the step (1) of the framework of Fig. 4, among other things.

The problem is important, but is NP-complete. One might think that the absence of currency constraints or CFDs would simplify the analysis. Unfortunately, its intractability is *robust*.

Theorem 1: *The satisfiability problem for entity specifications is NP-complete. It remains NP-hard for valid specifications $S_e = (I_t, \Sigma, \Gamma)$ of an entity when (1) both Σ and Γ are fixed; (2) $\Gamma = \emptyset$,* i.e., *with only currency constraints; or (3) $\Sigma = \emptyset$,* i.e., *when only constant CFDs are present.*

Proof: We show that the satisfiability problem for entity specifications is NP-complete. For the NP upper bound it suffices to observe that the following NP algorithm correctly decides whether a given specification has a valid completion. Given a specification $S_e = (I_t, \Sigma, \Gamma)$, the algorithm simply guesses a completion I_t^c of I_t and then verifies whether (i) $I_t^c \models \Sigma$; and (ii) $I_t^c \models \Gamma$. The latter two checks are in PTIME and if the guessed completion passes these checks, then the algorithm returns "yes". Otherwise, the guessed completion is rejected. Note that a "guess" simply adds currency information to I_t and is bounded by the values that appear as constants in I_t.

The NP-lower bound is established by a reduction from the 3-satisfiability problem. An instance of the 3-satisfiability problem is formula $\varphi = C_1 \wedge \cdots \wedge C_r$ with $C_j = \ell_1^j \vee \ell_2^j \vee \ell_3^j$ and for $k \in \{1,2,3\}$ and $j \in [1, r]$, ℓ_k^j is either a variable or a complement of a variable from $X = \{x_1, \ldots, x_n\}$, and is to determine whether φ is satisfiable. This problem is known to be NP-complete (cf. [22]).

We define a specification $S_e = (I_t, \Sigma, \Gamma)$ such that there exists a valid completion of S_e iff φ is satisfiable. The specification S_e consists of a temporal instance I of schema $R(D, C, P, U, V, W)$ and a fixed set of currency constraints

	satisfiability	implication	true value	minimum coverage
complexity	NP-complete	coNP-complete	coNP-complete	Σ_2^p-complete

Fig. 5. Complexity of reasoning about conflict resolution

Σ. No constant CFDs are defined in S_e. Intuitively, D is to distinguish between tuples that encode truth assignments and tuples that correspond to clauses in φ; C is to identify variables (by x_i) and clauses (by $j \in [1, r]$); P is used to enforce the validity of clauses and finally, U, V and W represent the positions (1, 2 and 3, resp) of variables in each clause.

We first explain how the instance I of R together with the currency constraints in Σ is to encode truth assignments for X and clauses in φ. More specifically, for each variable $x_i \in X$, we use two constants a_i and b_i such that $a_i \preccurlyeq_A b_i$ encodes that x_i is set to true, whereas $b_i \preccurlyeq_A a_i$ encodes that $\bar{x}_i$ is set to true (or, equivalently that x_i is set to false). Here A ranges over attributes U, V and W. More specifically, for each variable $x_i \in X$ we include two tuples in I:

$$(0, x_i, 0, a_i, a_i, a_i) \quad \text{and} \quad (0, x_i, 0, b_i, b_i, b_i).$$

These are to encode truth assignments of X. To ensure that the choice of truth value for variables is consistent, we include the following currency constraints in Σ:

$$\forall t_1, t_2\ R(t_1[D] = 0 \wedge t_2[D] = 0 \wedge t_1[C] = t_2[C] \wedge \\ \wedge\, t_1[A] \prec t_2[A] \rightarrow t_1[B] \prec t_2[B]),$$

where A and B range over distinct pairs taken form $\{U, V, W\}$. These currency constraints enforce that variables x_i are set to true (resp. false) independent of the position at which they appear in clauses (*i.e.,* in attribute U, V or W).

We next consider the clauses in φ. Let $C_j = \ell_1^j \vee \ell_2^j \vee \ell_3^j$ and observe that this can be equivalently written as $\bar{\ell}_1^j \wedge \bar{\ell}_2^j \rightarrow \ell_3^j$. For instance, consider a clause $C = x_1 \vee \bar{x}_2 \vee \bar{x}_3$. This is equivalent to $\bar{x}_1 \wedge x_2 \rightarrow \bar{x}_3$. Given this, we include two tuples in I for each clause:

$$(1, j, 1, v_1, v_2, v_3) \quad \text{and} \quad (1, j, 2, v_1', v_2', v_3'),$$

where $v_i = a_k$ and $v_i' = b_k$ if $\bar{\ell}_i^j = x_k$, and $v_i = b_k$ and $v_i' = a_k$ if $\ell_i^j = x_k$, for $i = 1, 2$, and conversely for $i = 3$. The example clause C is thus encoded by $(1, \_, 1, b_1, a_2, b_3)$ and $(1, \_, 2, a_1, b_2, a_3)$. The link between truth assignments selected by completions and the validity of clauses is established by means of the following currency constraint:

$$\forall t_1, t_2 R(t_1[D] = 1 \wedge t_2[D] = 1 \wedge t_1[C] = t_2[C] \wedge \\ t_1[P] = 1 \wedge t_1[P] = 2 \wedge \\ t_1[U] \prec t_2[U] \wedge t_1[V] \prec t_2[V] \rightarrow t_1[W] \prec t_2[W]).$$

This constraint tells that whenever the truth assignment (represented by a completion) makes $\bar{\ell}_1^j \wedge \bar{\ell}_2^j$ true, then it must also make ℓ_3^j true.

We next show the correctness of the reduction. Suppose that φ is true and let μ_X be a satisfying truth assignment. We define a valid completion of S_e as follows: For attributes D, C and P we order the tuples in I arbitrarily. For attributes U (and consequently also for V and W by the currency constraints) we set $a_i \preccurlyeq_U^c b_i$ if $\mu_X(x_i)$ is true and $b_i \preccurlyeq_U^c a_i$ otherwise. We need to verify that the second currency constraint is satisfied. This follows immediately from the fact that each clause is satisfied by μ_X. Conversely, suppose that we have a valid completion. From this, we define μ_X by simply setting $\mu_X(x_i) = 1$ if $a_i \preccurlyeq_U^c b_i$ and $\mu_X(x_i) = 0$ otherwise. Similarly as above, it is readily verified that μ_X satisfies all clauses. Indeed, this follows from the second currency constraint.

It remains to show that the satisfiability problem for specifications is NP-complete when (a) Σ and Γ are fixed; (b) $\Gamma = \emptyset$; or (c) $\Sigma = \emptyset$. Since we have just shown that the satisfiability problem is in NP, for general Σ and Γ, it suffices to show the lower bounds. Furthermore, observe the proof above uses a fixed set of currency constraints and does not use constant CFDs. In other words, (a) and (b) follow directly from that lower bound proof. It remains to show (c), *i.e.,* that the satisfiability problem for specifications is NP-hard even when only constant CFDs are present. We establish this lower bound by a reduction from the complement of the tautology problem, which is known to coNP-complete. An instance of the tautology problem is a formula $\varphi = C_1 \vee \cdots \vee C_r$ with $C_j = \ell_1^j \wedge \ell_2^j \wedge \ell_3^j$ and each ℓ_k^j is either a variable or a complement of a variable from $X = \{x_1, \ldots, x_n\}$, and is to determine whether φ is true for any truth assignment of X. We define a specification $S_e = (I_t, \Sigma = \emptyset, \Gamma)$ such that S_e has a valid completion ifff φ is not a tautology.

The temporal instance I_t of S_e is an instance over the schema $R(X_1, \ldots, X_n, C)$ and consists of two tuples $(0, 0, \ldots, 0)$ and $(1, 1, \ldots, 1)$. We do not impose any currency order or currency constraints on I_t. Observe that each completion I_t^c results in a current tuple $\mathsf{LST}(I_t^c)$ that encodes a truth assignment μ_X of X in its first n attributes.

The set Γ of constant CFDs is given as follows: For each clause C_j we define $\psi_j = t_p[L_1, L_2, L_3] \rightarrow t_p[C]$ where $L_i = X_k$ if ℓ_i^j or $\bar{\ell}_i^j$ is x_k and the pattern tuple $t_p = (v_1, v_2, v_3, 1)$ is given by $v_i = 1$ if $\ell_i^j = x_k$ and $v_i = 0$ if $\ell_i^j = \bar{x}_k$. Clearly, a completion $I_t^c \models \psi_j$ if the truth assignment μ_X encoded by the current tuple $\mathsf{LST}(I_t^c)$ makes C_j true. We further add $\psi_C = s_p[C] \rightarrow s_p[C]$ with $s_p = (1, 0)$ to Γ, which intuitively prevents any clause to be satisfied. Indeed, observe that a completion I_t^c such that $I_t^c \models \psi_C$ must set the C-attribute of its current tuple to 0. Contrast this with the requirement on the C-attribute of current tuples imposed by the ψ_j's.

We next show the correctness of the reduction. If φ is a tautology then every truth assignment μ_X makes at least one clause C_j true. That is, any valid I_t^c must set the C-attribute of its current to 1 (by ψ_j) and at the same time it must set the C-attribute to 0 (by ψ_C). Hence, no valid completion can exists. Conversely, if there exists a valid completion I_t^c of S_e such that $I_t^c \models \Gamma$ then this implies that its current tuple must have its C-attribute set to 0. In other words, none of the lefthand sides of the ψ_j's can be true and hence μ_X must make all clauses false. In other words, μ_X is a counter example to the validity of φ and hence φ is not a tautology.

$\square$

Implication. Consider a valid specification $S_e = (I_t, \Sigma, \Gamma)$ of an entity and a partial temporal order $O_t = (I_e, \preccurlyeq'_{A_1}, \ldots, \preccurlyeq'_{A_n})$. We say that O_t is *implied by* S_e, denoted by $S_e \models O_t$, iff *for all* valid completions I_t^c of S_e, $O_t \subseteq I_t^c$. Here $O_t \subseteq I_t^c$ if $\preccurlyeq'_{A_i} \subseteq \preccurlyeq^c_{A_i}$ for all $i \in [1, n]$, where $I_t^c = (I_e, \preccurlyeq^c_{A_1}, \ldots, \preccurlyeq^c_{A_n})$.

The *implication problem* is to decide, given a valid specification S_e and a partial temporal order O_t, whether $S_e \models O_t$.

That is, no matter how we complete the temporal instance I_t of S_e, as long as the completion is valid, it includes O_t. The implication analysis is conducted at step (2) of the framework of Fig. 4, for deducing true values of attributes.

Unfortunately, the implication problem is coNP-complete.

Theorem 2: *The implication problem for conflict resolution is coNP-complete. It remains coNP-hard for valid specifications $S_e = (I_t, \Sigma, \Gamma)$ of an entity when (1) both Σ and Γ are fixed; (2) $\Gamma = \emptyset$,* i.e., *with only currency constraints; or (3) $\Sigma = \emptyset$,* i.e., *when only constant CFDs are present.*

Proof: We show that the implication problem for conflict resolution is coNP-complete. The coNP upper bound is verified by providing an NP algorithm for the complement problem. In a nutshell, given a specification $S_e = (I_t, \Sigma, \Gamma)$ and a partial temporal order O_t, the algorithm simply guesses a completion I_t^c of I_t and then verifies whether (i) $I_t^c \models \Sigma$; (ii) $I_t^c \models \Gamma$; and (iii) $O_t \not\subseteq I_t^c$. If I_t^c passes these checks successfully, then the algorithm returns "yes" and hence O_t is not deterministic. Otherwise, the current guess is rejected. This clearly is an NP algorithm for the complement problem and hence the implication problem is in coNP.

For the lower bounds, we need to show that the implication problem is coNP-hard when (1) both Σ and Γ are fixed; (2) $\Gamma = \emptyset$; or (3) $\Sigma = \emptyset$. The lower bounds for (1) and (2) are established by a modification of the proof of Theorem 1. More specifically, the modifications are as follows: First, the relation schema used in that proof is extended with an additional attribute A; Second, each tuple t in the temporal instance has now two copies: one tuple t^a with its A-attribute set to a constant a and one tuple t^b with its A-attribute set to b; Finally, the premise of each currency constraint used in the proof of Theorem 1 carries an additional condition "$t_1[A] = a \wedge t_2[A] = b \wedge t_1[A] \preccurlyeq_A t_2[A]$". These conditions enforce the constraints to have an effect only on completions that make b more current than a in attribute A.

Denote by $S'_e = (I'_t, \Sigma', \Gamma = \emptyset)$ the specification obtained from S_e in the proof of Theorem 1 after the modifications have been made. Let O_t be the partial temporal order $(I'_t, \{t^b \preccurlyeq_A t^a\}, \emptyset, \ldots, \emptyset)$, where t^a and t^b are the two copies of an arbitrary tuple t in I_t .We claim the following: (i) S'_e is valid; and (ii) O_t is deterministic *w.r.t.* S'_e iff the formula φ is not satisfiable. For (i) it suffices to observe that any completion $(I'_t)^c$ which puts $t^b \preccurlyeq_A t^a$ and arbitrarily completes all other attributes, results in a valid completion. Indeed, this is simply because the conditions added to the premise of constraints used in the proof of Theorem 1 are false, and hence the currency constraint vacuously hold. Hence, S'_e is valid.

For (ii), assume first that there exists a truth assignment μ_X which makes φ true. We then define a completion $(I'_t)^c$ of I'_t by setting $t^a \preccurlyeq_A t^b$, where t is the tuple used to define O_t, and further completing the attributes based on μ_X as in the proof of Theorem 1. As a consequence, $O_t \not\subseteq (I'_t)^c$ and hence O_t is not deterministic for S'_e. Conversely, suppose that O_t is not deterministic. This implies the existence of a valid completion $(I'_t)^c$ of I'_t which puts $t^a \preccurlyeq_A t^b$ and further satisfies all currency constraints in Σ'. Similar to the proof of Theorem 1 it is readily verified that a truth assignment μ_X can be constructed from $(I'_t)^c$ which makes φ true. Hence, O_t is deterministic iff φ is not satisfiable. Observe that the proof only uses a fixed set of currency constraints and does not require any constant CFDs.

Similarly, The coNP-lower bound for (3) is established by a similar modification of the specification given in the proof of Theorem 1. More specifically, we extend the schema with an additional attribute A and the corresponding temporal instance I'_t consists of $t_0 = (a, 0, \ldots, 0)$ and $t_1 = (b, 1, \ldots, 1)$. We further extend the constant CFDs $\psi_j = t_p[L_1, L_2, L_3] \rightarrow t_p[C]$ in the proof of Theorem 1 to $\psi'_j = t'_p[A, L_1, L_2, L_3] \rightarrow t'_p[C]$ where $t'_p = (a, t_p)$. Similarly for ψ_C. That is, these constant CFDs only have an effect when the current tuples has a as its A-attribute value. Denote by $S'_e = (I'_t, \Sigma = \emptyset, \Gamma')$ the specification obtained in this way. Clearly, S'_e is consistent since we just need to enforce $t_0 \preccurlyeq^c_A t_1$ in a completion to assure that the corresponding current tuple vacuously satisfies the CFDs in Γ'. Consider $O_t = (I'_t, \{t_0 \preccurlyeq^c_A t_1\}, \emptyset, \ldots, \emptyset)$. Then, similar to the argument given above, O_t will be deterministic *w.r.t.* S'_e iff φ $\square$

True value deduction. The *true value problem* is to decide, given a valid specification S_e for an entity, whether $\mathsf{T}(S_e)$ exists. That is, there exists a tuple t_c such that for all valid completions I_t^c of S_e, $\mathsf{LST}(I_t^c) = t_c$.

This analysis is needed by step (3) of the framework (Fig. 4) to decide whether S_e has enough information to deduce $\mathsf{T}(S_e)$.

However, this problem is also nontrivial: it is intractable.

Theorem 3: *The true value problem for conflict resolution is coNP-complete. It remains coNP-hard for valid specifications $S_e = (I_t, \Sigma, \Gamma)$ for an entity even when (1) both Σ and Γ are fixed; (2) $\Gamma = \emptyset$,* i.e., *with only currency constraints; or (3) $\Sigma = \emptyset$,* i.e., *when only constant CFDs are present.*

Proof: We show that the true value problem for conflict resolution is coNP-complete. The coNP upper bound is verified by providing an NP algorithm for the complement problem. In a nutshell, given a specification $S_e = (I_t, \Sigma, \Gamma)$, the algorithm simply guesses *two* completions I_t^c and $(I_t^c)'$ of I_t and then verifies whether both completions are valid and generate *different* current tuples. If the guessed completions satisfy these conditions then the algorithm returns "yes" and no true value exists. Otherwise, the current guesses are rejected. This clearly is an NP algorithm for the complement problem and hence the true value problem is in coNP.

For the lower bounds, we need to show that the true value problem is coNP-hard when (1) both Σ and Γ are fixed; (2) $\Gamma = \emptyset$; or (3) $\Sigma = \emptyset$.

The lower bounds for (1) and (2) are established by a modification of the proof of Theorem 2. Indeed, it suffices to add two tuples $t^a_{\#} = (a, \#, \ldots, \#)$ and $t^b_{\#} = (b, \#, \ldots, \#)$ to the temporal instance, together with additional currency constraints that enforce $\#$ to come after any other constant in all attributes of the schema except for A (which does not carry $\#$). As a consequence, any completion can only result in current tuples $t^a_{\#}$ or $t^b_{\#}$. Denote by $S''_e = (I''_t, \Sigma'', \Gamma = \emptyset)$ the specification obtained in this way from S'_e in the proof of Theorem 2.

As argued there, S''_e is valid since one only has to consider a completion which puts $t^b_{\#} \preccurlyeq_A t^a_{\#}$. Furthermore, we next show that a true value exists iff φ is not satisfiable. Indeed, suppose that φ is not satisfiable then any valid completion $(I''_t)^c \models \Sigma''$ has to set $t^b_{\#} \preccurlyeq_A t^a_{\#}$. Indeed, otherwise the currency constraints will be triggered and the completion would generate a satisfying truth assignment for φ, which by assumption does not exist. Hence, the true value will be the tuple $t^a_{\#}$. Conversely, suppose that no true value exists. This implies that there exists two completions of I''_t, one which leads to current tuple $t^a_{\#}$, and one which leads to current tuple $t^b_{\#}$. In the second case, $t^a_{\#} \preccurlyeq_A t^b_{\#}$ and hence, as argued in the proof of Theorem 2 one can construct a satisfying truth assignment for φ from the completions. Hence, if no true value exists, then φ must be satisfiable.

The coNP-lower bound for (3) is established by a modification of the specification given in the proof for constant CFDs of Theorem 1. The modifications are as follows: We introduce a third tuple to the temporal instance $t_b = (b, b, \ldots, b)$ and extend the set Γ' with the following constant CFDs: $\psi^i_{a \neg b} = t_p[AX_i] \to t_p[A]$ with $t_p = (a, b, b)$, for $i \in [1, n]$. These constant CFDs prevent the current tuple t in completions to have $t[A] = a$ and $t[X_i] = b$ for $i \in [1, n]$. In addition we add $\psi^i_{bb} = s_p[A] \to s_p[X_i]$ with $s_p = (b, b)$ and $\psi_{bb} = s_p[A] \to s_p[C]$ with $s_p = (b, b)$, enforcing current tuples t with $t[B] = b$ to have the constant b in all of its attributes.

Denote by $S''_e = (I''_t, \Sigma = \emptyset, \Gamma'')$ the specification obtained from S'_e in the proof of Theorem 1. A completion which results in current tuple t_b is clearly a valid completion and hence S''_e is valid itself. Moreover, it is readily verified that a true value exists iff φ is a tautology. Indeed, observe first that completions either result in the current tuple t_b or a tuple of the form $(a, \mu_X, 0)$, where μ_X is a truth assignment for X. Whereas t_b can always be witnessed by a valid completions (as mentioned above), $(a, \mu_X, 0)$ can only be witnessed provided that μ_X makes φ false (using the argument given in the proof ofTheorem 1. Hence t_b is the true value iff φ is a tautology. $\square$

Coverage analysis. The *minimum coverage problem* is to determine, given a valid specification $S_e = (I_t, \Sigma, \Gamma)$ and a positive integer k, whether there exists a partial temporal order O_t such that (1) $\mathsf{T}(S_e \oplus O_t)$ exists, and (2) $|O_t| \leq k$.

Intuitively, this is to check whether one can add a partial temporal order O_t of a *bounded* size to a specification such that the enriched specification has sufficient information to deduce all the true values of an entity. The analysis of *minimum* O_t is required by step (4) of the framework of Fig. 4.

This problem is, unfortunately, Σ^p_2-complete ($\mathsf{NP}^{\mathsf{NP}}$).

Theorem 4: *The minimum coverage problem is Σ^p_2-complete. It remains Σ^p_2-hard for valid specifications $S_e = (I_t, \Sigma, \Gamma)$ for an entity even when (1) both Σ and Γ are fixed; (2) $\Gamma = \emptyset$,* i.e., *with only currency constraints; or (3) $\Sigma = \emptyset$,* i.e., *when only constant CFDs are present.*

Proof: We show that the minimal coverage problem is Σ^p_2-complete. For the Σ^p_2 upper bound it suffices to observe that the following $\mathrm{NP}^{\mathrm{coNP}}$ algorithm correctly decides whether there exists a partial temporal order O_t of size $|O_t| \leq k$ such that $\mathsf{T}(S_e \oplus O_t)$ exists. Given specification $S_e = (I_t, \Sigma, \Gamma)$, the algorithm first guesses a partial temporal order O_t and then checks whether $|O_t| \leq k$ and whether $\mathsf{T}(S_e \oplus O_t)$ exists. The latter can be done in coNP (see Theorem 3). If the guessed partial temporal order passes these checks, then the algorithm returns "yes". Otherwise, the guessed order is rejected.

We need to show that the minimal coverage problem is Σ^p_2-hard when (1) Σ and Γ are fixed; (2) $\Gamma = \emptyset$; or (3) $\Sigma = \emptyset$. For (a) and (b) we establish the Σ^p_2-lower bound is established by reduction from the $\exists^*\forall^*\mathsf{DNF}$ problem, which is known to be Σ^p_2-complete [25]. An instance of the $\exists^*\forall^*\mathsf{DNF}$ problem is a formula of the form $\varphi = \exists X \forall Y \psi$ where $X = \{x_1, \ldots, x_n\}$ and $Y = \{y_1, \ldots, y_m\}$, $\psi = C_1 \vee \cdots \vee C_r$ and for $j \in [1, r]$, $C_j = \ell^j_1 \wedge \ell^j_2 \wedge \ell^j_3$ and for $k = 1, 2, 3$, the literal ℓ^j_k is either a variable or the complement of a variable in $X \cup Y$; It is to determine whether φ is true.

We define a specification $S_e = (I_t, \Sigma, \Gamma)$ and a constant k such that the minimal coverage problem for S_e and k has a solution iff φ is true. In particular, in S_e we have a fixed set of currency constraints and no constant CFDs. Hence, the reduction shows (1) and (2). We deal with case (3) later.

Consider the relation schema $R(A, D, C, P, U, V, W)$ used in the proof of Theorem 2. We populate the corresponding temporal instance $I_t = (I, \preccurlyeq_A, \preccurlyeq_D, \preccurlyeq_C, \preccurlyeq_P, \preccurlyeq_U, \preccurlyeq_V, \preccurlyeq_W)$ as follows: We assume the presence of $2(n+m)$ distinct constants a_i and b_i for $i \in [1, n]$ and c_i and d_i for $i \in [1, m]$. As in the proof of Theorem 2, truth values for variables in X will be encoded by means of two tuples:

$$(a, 0, x_i, 0, a_i, a_i, a_i) \quad \text{and} \quad (a, 0, x_i, 0, b_i, b_i, b_i).$$

with their A-attribute set to a, and two tuples and

$$(b, 0, x_i, 0, a_i, a_i, a_i) \quad \text{and} \quad (b, 0, x_i, 0, b_i, b_i, b_i).$$

with their A-attribute set to b. Similarly, truth values for variables in Y are encoded by the following four tuples:

$$(a, 0, y_i, 0, c_i, c_i, c_i) \quad \text{and} \quad (a, 0, y_i, 0, c_i, c_i, c_i).$$

and

$$(b, 0, y_i, 0, d_i, d_i, d_i) \quad \text{and} \quad (b, 0, y_i, 0, d_i, d_i, d_i).$$

In addition we add a currency constraint to Σ for every pair of attributes (L, L') taken from $\{U, V, W\}$:

$$\forall t_1, t_2 R(t_1[D] = 0 \wedge t_2[D] = 0 \wedge t_1[C] = t_2[C] \wedge \\ t_1[A] = a \wedge t_2[A] = b \wedge t_1[A] \prec_A t_2[A] \wedge \\ t_1[L] \prec t_2[L] \rightarrow t_1[L'] \prec t_2[L'].$$

These currency constraints ensure that whenever $a \prec_A b$, the order between a_i and b_i, and c_i and d_i is consistent in all attributes U, V and W. As before, $a_i \prec_U b_i$ indicates that x_i is set to true, whereas $b_i \prec_U a_i$ indicates that x_i is false. Similarly for variables in Y but using the constants c_i and d_i instead. In other words, with every completion of I_t in which $a \prec_A b$, we can associate truth assignments μ_X and μ_Y of X and Y, respectively.

We next encode the clauses in φ in a similar way as in the proof of Theorem 2. More specifically, given $C_1 \vee \cdots \vee C_r$ we encode its negation $\bar{C}_1 \wedge \cdots \wedge \bar{C}_r$ with $\bar{C}_j = \bar{\ell}_1^j \vee \bar{\ell}_2^j \vee \bar{\ell}_3^j$. Such clauses can be equivalently written as $\ell_1^j \wedge \ell_2^j \rightarrow \bar{\ell}_3^j$ by means of the tuples

$$(a, 1, j, 1, v_1, v_2, v_3) \quad \text{and} \quad (a, 1, j, 2, v_1', v_2', v_3'),$$

and their b-variants

$$(b, 1, j, 1, v_1, v_2, v_3) \quad \text{and} \quad (a, 1, j, 2, v_1', v_2', v_3').$$

Here, $v_i = a_k$ and $v_i' = b_k$ if $\ell_i^j = x_k$, and $v_i = b_k$ and $v_i' = a_k$ if $\bar{\ell}_i^j = x_k$, for $i = 1, 2$, and defining v_i and v_i' the other way around for $i = 3$. Similarly, for variables in Y but then using constants c_i and d_i instead. For example, consider the clause $C = x_1 \wedge \bar{x}_2 \wedge \bar{y}_3$ whose complement is $\bar{C} = \bar{x}_1 \vee x_2 \vee y_3$. Equivalently, we write $\bar{C}$ as $x_1 \wedge \bar{x}_2 \rightarrow y_3$. Hence, we encode $\bar{C}$ by $(a, 1, \_, 1, a_1, b_2, c_3)$ and $(a, 1, \_, 2, b_1, a_2, d_3)$, together with their b-counterparts $(b, 1, \_, 1, a_1, b_2, c_3)$ and $(b, 1, \_, 2, b_1, a_2, d_3)$.

The link between truth assignments selected by completions and the validity of (complemented) clauses is established by means of the following currency constraint:

$$\forall t_1, t_2 R(t_1[D] = 1 \wedge t_2[D] = 1 \wedge t_1[C] = t_2[C] \wedge \\ t_1[P] = 1 \wedge t_1[P] = 2 \wedge \\ t_1[A] = a \wedge t_2[A] = b \wedge t_1[A] \prec_A t_2[A] \wedge \\ t_1[U] \prec t_2[U] \wedge t_1[V] \prec t_2[V] \rightarrow t_1[W] \prec t_2[W].$$

This constraint tells that whenever the truth assignment (represented by a completion) makes $\ell_1^j \wedge \ell_2^j$ true, then it must also make $\bar{\ell}_3^j$ true, provided that $a \prec_A b$.

We also include two tuples $t_\#^a = (a, \#, \ldots, \#)$ and $t_\#^b = (b, \#, \ldots, \#)$, which will serve as potential true values of the entity represented by S_e. We enforce the symbol # to come after any other constant by means of currency constraints (one for each attribute in R). Clearly, in valid completions: when $a \prec_A b$ then $t_\#^b$ is the current tuple; when $b \prec_A a$, $t_\#^a$ is the current tuple.

Finally, we ensure that the partial temporal order O_t can only add currency information related to the values a_i and b_i in the instance; hence O_t can only affect the choice of truth values for variables in X. To achieve this, observe that given we instance I constructed so far, $|O_t|$ can reach the maximum of $7|I|^2$, where 7 is simply the number of attributes in R. We let $k = 7|I|^2$. Next, for each constant v different from the a_i's and b_i' we add $p > k$ tuples of the form $(v_{\text{id}}, v, \ldots, v)$, where v_{id} is a unique identifier for each of these tuples. Let I' denote the temporal instance obtained in this way and let $S_e = (I', \Sigma, \Gamma = \emptyset)$. Clearly, any O_t which relates tuples in an attribute that concern values different from a_i and b_i will cause the addition of more than k tuples. Indeed, let B be an arbitrary attribute. Then the addition of $t \prec_B t'$ implies that $s \prec_B s'$ for all tuples s and s' that share the same B-attribute value with t and t', respectively. By the choice of k and the addition of $p > k$ tuples for each constant, any O_t of size $\leq k$ can only related tuples that contain a_i or b_i values in one of its attributes.

Observe that the specification S_e defined above is valid. Indeed, any completion which makes a more current than b in the A-attribute vacuously satisfies the currency constraints in Σ. As a consequence $t_\#^a$ will always be one of the possible current tuples. We claim that the minimal coverage problem has a solution iff φ is true. Suppose first that φ is false. In other words, for every μ_X of X, there exists a μ_Y of Y which makes $C_1 \vee \cdots \vee C_r$ false. Consider a partial temporal order O_t with $|O_t| \leq k$. By construction O_t can only add temporal information between tuples that concern variables in X. In other words, the impact of O_t is that it restricts the set of truth assignments of X that can be obtained by means of valid completions. However, since φ is false, even for each μ_X in this restricted set, there exists a μ_Y which makes the $C_1 \vee \cdots \vee C_r$ false. This in turn implies that $t_\#^b$ can be a current tuple in a completion that sets $a \prec_A b$. Indeed, simply consider the completion which (i) sets $a \prec_A b$; (ii) selects a μ_X that belongs to the restricted set; (iii) selects μ_Y such that the clauses are false; and (iv) arbitrarily complete the other attributes. It is easily verified that this completion indeed satisfies all currency constraints since it satisfies the constraints related to truth assignments and all constraints corresponding to the negated clauses (recall that μ_X and μ_Y make all $\bar{C}_j$ true.) Hence, when φ is false, both $t_\#^a$ and $t_\#^b$ are current tuples and no true value can exist, irregardless of the choice of O_t.

Conversely, suppose that φ is true. In other words, there exists a μ_X of X such that for all μ_Y of Y, $C_1 \vee \cdots \vee C_r$ is true. We let O_t be the partial temporal order that restricts the choices for truth assignments for X to be μ_X. By construction, this can be done using $\leq k$ added pairs. It is impossible that $t_\#^b$ becomes a current tuple. Indeed, for this to happen we need a completion which sets $a \prec Ab$ and in addition satisfies all constraints in Σ. This, however, would imply the existence of a μ_Y which together with μ_X, makes $C_1 \vee \cdots \vee C_r$ false. This in contrast to the assumption that φ is holds for μ_X. As a consequence, $\mathsf{T}(S_e \oplus O_t)$ exists and is equal to $t_\#^a$.

For case (3), when $\Sigma = \emptyset$ we show the Σ_2^p-hard again by a reduction from the $\exists^*\forall^*$DNF problem, but this time using constant CFDs only. The idea behind the reduction is similar to that of the reduction given for cases (a) and (b).

We use relational schema $R(A, X_1, \ldots, X_n, Y_1, \ldots, Y_m, C)$ and to start with, we instantiate the corresponding temporal instance I_t with two tuples $t_0 = (a, 0, 0, \ldots, 0)$ and $t_1 = (b, 1, 1, \ldots, 1)$. Completions thus lead to current tuples ranging over all possible truth assignments for X and Y. We further introduce tuple $t_b = (b, b, \ldots, b)$ which will correspond to the true value of the entity if it exists. Finally, let $k = n$ and add $p > n$ tuples of the form $(c_i, \ldots, c_i, 0, 0, 0, \ldots)$ and $(c_i, \ldots, c_i, 1, \ldots, 1)$ for $i \in [1, p]$. Here, the c_i's belong the the attributes A and $X_1, \ldots, X_n$. We further assume that initial temporal orders are available which indicate that the c_i's come before a, b, 0 and 1. Intuitively, the addition of these p tuples will cause any additional temporal information in the Y-attributes (and A-attribute) to have more than k effects, *e.g.*,, when $t_0 \prec_Y t_1$ then this information needs to be imposed on all p tuples as well since these contain the same values in their Y-attributes as t_0 and t_1. As a consequence, any partial temporal order O_t of size $\leq k$ can only insert information on the X-attributes. In other words, adding O_t will cause the selection of a truth assignment for X.

Next we define the constant CFDs for the specification. We use the same constant CFDs as in the proof of Theorem 3 and let $S_e = (I_t, \Sigma = \emptyset, \Gamma)$ the resulting specification. As argued in the proof of Theorem 3, S_e is valid because any completion which puts $a \prec_A b$ vacuously satisfies the CFDs in Γ. Recall also that t_b will be the current tuple in this case. We next show that the minimal coverage problem has a solution for S_e and k iff φ is true. Indeed, suppose that φ is false. Then for all μ_X there exists a μ_Y, such that $C_1 \vee \cdots \vee C_r$ is false. Let O_t be any partial temporal order of size $\leq k$. As argued above, the addition of O_t causes the selection of a subset of truth assignments of X. For any such μ_X we have a μ_Y which makes the clauses false. In other words, a completion exists which puts (i) $b \prec_A a$; (ii) selects a μ_X; and (iii) μ_Y which falsifies φ. By the definition of the CFDs, this implies that a current tuple of the form $(a, \ldots)$ exists and hence there is no true value for the entity (since t_b is also a current tuple). Conversely, if φ is true, we simple take O_t to be such that it selects the satisfying truth assignment (observe that such O_t can be taken of size $\leq k$) and hence for any μ_Y we have that the clauses are satisfied. In other words, completions with $b \prec_A a$ cannot exist by the definition of the constant CFDs. Hence, t_b will be the only possible current tuple and a true value exists. □

Remark. The complexity results are summarized in Fig. 5. From the results we find the following.

(1) The main conclusion is that these problems are hard. In fact as we have shown that all the lower bounds remain intact for valid specifications $S_e = (I_t, \Sigma, \Gamma)$ of an entity when (1) both Σ and Γ are fixed; (2) $\Gamma = \emptyset$, *i.e.,* when constant CFDs are absent; or (3) $\Sigma = \emptyset$, *i.e.,* when currency constraints are absent. Hence unless P = NP, efficient algorithms for solving these problems are necessarily *heuristic*.

(2) The results not only reveal the complexity of conflict resolution, but also advance our understanding of data currency and consistency. Indeed, while the minimum coverage problem is particular for conflict resolution and has *not* been studied before, the other problems are also of interest to the study of data currency. Theorems 1, 2 and 3 show that currency constraints make our lives easier as opposed to denial constraints: they reduce the complexity of inferring data currency reported in [15], from Σ_2^p-complete, Π_2^p-complete (coNP$^{\mathsf{NP}}$) and Π_2^p-complete down to NP-complete, coNP-complete and coNP-complete, respectively, When it comes to data consistency, it is known that the satisfiability and implication problems for general CFDs are NP-complete and coNP-complete, respectively [13]. Theorems 1 and 2 give a *stronger* result: these lower bounds already hold for constant CFDs.

V. Algorithms for Conflict Resolution

We next provide algorithms underlying the framework depicted in Fig. 4. We first present an algorithm for checking whether a specification is valid (step (1) of the framework; Section V-A). We then study how to deduce true attribute values from a valid specification (step (2); Section V-B). Finally, we show how to generate suggestions (step (4); Section V-C).

A. Validity Checking

We start with algorithm IsValid that, given a specification $S_e = (I_t, \Sigma, \Gamma)$, returns true if S_e is valid, and false otherwise. As depicted in Fig. 4, IsValid is invoked for an initial specification S_e and its extensions $S_e \oplus O_t$ with users' input.

Theorem 1 tells us that it is NP-complete to determine whether S_e is valid. Hence IsValid is necessarily heuristic if it is to be efficient. We approach this by reducing the problem to SAT, one of the most studied NP-complete problem, which is to decide whether a Boolean formula is satisfiable (see, *e.g.,* [3]). Several high-performance tools for SAT (SAT-solvers) are already in place [3], which have proved effective in software verification, AI and operations research, among others. For instance, MiniSAT [18] can effectively solve a formula with $4,500$ variables and $100K$ clauses in 1 second.

Algorithm. Using a SAT-solver, We outline IsValid as follows. (1) Instantiation(S_e): It expresses S_e as a set $\Omega(S_e)$ of predicate formulas. (2) ConvertToCNF($\Omega(S_e)$): It then converts $\Omega(S_e)$ into a CNF $\Phi(S_e)$ (the conjunctive normal form) such that S_e is valid iff $\Phi(S_e)$ satisfiable. (3) Finally, it applies an SAT-solver to $\Phi(S_e)$, and returns true iff $\Phi(S_e)$ is true.

We next present the details of procedures Instantiation and ConvertToCNF. We denote also by R the set $\{A_i \mid i \in [1, n]\}$ of attributes of R. We define a strict partial order $\prec_{A_i}^v$ on the values in the union of adom$(I_e.A_i)$ and all the constants that appear in attribute A_i of some constant CFDs in Γ.

Instantiation. We express the currency orders, currency constraints and CFDs of S_e in a uniform set $\Omega(S_e)$ of constraints, referred to as *instance constraints*. This is done by instantiating variables in S_e with data in active domains as follows.

(1) *Currency orders*. To encode currency orders in I_t, for each $A_i \in R$, we include the following constraints in $\Omega(S_e)$.

(a) Partial orders in I_t: ($true \rightarrow t_1[A_i] \prec^v_{A_i} t_2[A_i]$) for each $t_1 \preccurlyeq_{A_i} t_2$ in I_t, as long as $t_1[A_i] \neq t_2[A_i]$.

(b) Transitivity of $\prec_{A_i}$: $(a_1 \prec^v_A a_2 \wedge a_2 \prec^v_A a_3 \rightarrow a_1 \prec^v_A a_3)$ for all distinct values a_1, a_2, a_3 in $\mathsf{adom}(I_e.A_i)$.

(c) Asymmetry: $(a \prec^v_{A_i} b) \rightarrow \neg(b \prec^v_{A_i} a)$ for $a, b \in \mathsf{adom}(I_e.A_i)$.

Intuitively, these assure that each $\prec_{A_i}$ is a *strict partial order* (via (b) and (c)), and express available temporal information in I_t as predicate formulas (via (a)).

(2) *Currency constraints*. For each currency constraint $\varphi = \forall t_1, t_2\ (\omega \rightarrow t_1 \prec_{A_r} t_2)$ in Σ and for all distinct tuples $s_1, s_2 \in I_e$, we include the following constraint in $\Omega(S_e)$:

$$\mathsf{ins}(\omega, s_1, s_2) \ \rightarrow\ s_1[A_r] \prec^v_{A_r} s_2[A_r],$$

where $\mathsf{ins}(\omega, s_1, s_2)$ is obtained from ω by (a) substituting $s_i[A_j]$ for t_i and $\prec^v_{A_j}$ for $\prec_{A_j}$ in each predicate $t_1 \prec_{A_j} t_2$, for $i \in [1, 2]$; and (b) evaluating each conjunct of ω defined with a comparison operator to its truth value *w.r.t.* s_1 and s_2. Intuitively, $\mathsf{ins}(\omega, s_1, s_2)$ "instantiates" ω with s_1 and s_2.

Example 7: For currency constraint φ_1 in Fig. 3, and tuples r_1 and r_2 in Fig. 2 for Edith, its instance constraint is (*true*$\rightarrow$ *working* $\prec^v_{\mathsf{status}}$ *retired*). Note that the precondition of φ_1 is evaluated true on these two particular tuples.

For φ_6 and r_1, r_2, we get (*working* $\prec^v_{\mathsf{status}}$ *retired* $\rightarrow$ *212* $\prec^v_{\mathsf{AC}}$ *415*), by replacing $\prec_{\mathsf{status}}$ with $\prec^v_{\mathsf{status}}$, and by replacing tuples with their corresponding attribute values. □

(3) *Constant CFDs*. For each constant CFD $t_p[X] \rightarrow t_p[B]$ in Γ and each $b \in \mathsf{adom}(I_e.B) \setminus \{t_p[B]\}$, $\Omega(S_e)$ includes

$$\psi = (\omega_X \ \rightarrow\ b \prec^v_B t_p[B]),$$

where ω_X is a conjunction of all formulas of the form $a \prec^v_{A_j} t_p[A_j]$ for each $a \in \mathsf{adom}(I_e.A_j) \setminus \{t_p[A_j]\}$ and each $A_j \in X$.

Intuitively, constraint ψ asserts that if $t_p[X]$ is true in attributes X, then $t_p[B]$ is the true value of B.

Example 8: Recall constant CFD ψ_1 (Fig. 3). For E_1 (Edith), it is encoded by two instance constraints below, in Ω_{E_1}:

$$212 \prec^v_{\mathsf{AC}} 213 \wedge 415 \prec^v_{\mathsf{AC}} 213 \rightarrow NY \prec^v_{\mathsf{city}} LA,$$
$$212 \prec^v_{\mathsf{AC}} 213 \wedge 415 \prec^v_{\mathsf{AC}} 213 \rightarrow SFC \prec^v_{\mathsf{city}} LA,$$

i.e.,LA is her true city value if her true AC value is *213*. □

ConvertToCNF. We convert $\Omega(S_e)$ into a CNF $\Phi(S_e)$ as follows. We substitute a Boolean variable $x^{A_i}_{a_1 a_2}$ for each predicate $a_1 \prec^v_{A_i} a_2$ in $\Omega(S_e)$, and write each formula of the form $(x_1 \wedge \cdots \wedge x_k \rightarrow x_{k+1})$ as $(\neg x_1 \vee \cdots \vee \neg x_k \vee x_{k+1})$. Then $\Phi(S_e)$ is a CNF with the conjunction of all formulas in $\Omega(S_e)$.

One can readily verify the following (by contradiction), which justifies the reduction from the validity of S_e to SAT.

Lemma 5: *Specification S_e is valid iff its converted* CNF *$\Phi(S_e)$ is satisfiable.*

Complexity: Observe the following. (a) The size $|\Omega(S_e)|$ of $\Omega(S_e)$ is bounded by $O((|\Sigma|+|\Gamma|)|I_t|^2+|I_t|^3)$, since encoding currency orders, currency constraints and constant CFDs is in time $O(|I_t|^3)$, $O(|\Sigma||I_t|^2)$ and $O(|\Gamma||I_t|^2)$, respectively. (b) It takes $O(|\Omega(S_e)|)$ time to convert $\Omega(S_e)$ into $\Phi(S_e)$. Hence the size of the CNF $\Phi(S_e)$ is bounded by $O((|\Sigma|+|\Gamma|)|I_t|^2+|I_t|^3)$. In practice, an entity instance I_t is typically *much smaller than* a database, and the sets Σ and Γ of constraints are also *small*. As will be seen in Section VI, SAT-solvers can efficiently process CNFs of this size.

Algorithm DeduceOrder

Input: A valid specification $S_e = (I_t, \Sigma, \Gamma)$ of an entity.
Output: A partial temporal order O_d such that $S_e \models O_d$.

1. $\Omega(S_e) := \mathsf{Instantiation}(S_e)$; $\Phi(S_e) := \mathsf{ConvertToCNF}(\Omega(S_e))$;
2. $O_d := (I_e, \emptyset, \ldots, \emptyset)$;
3. **while** there exists a one-literal clause C in $\Phi(S_e)$ **do**
   /* $x^A_{a_1 a_2}$ in C is the variable denoting $a_1 \prec^v_A a_2$ */
4. **if** C is a one-literal clause ($x^A_{a_1 a_2}$) **then**
5. add $a_1 \prec^v_A a_2$ to O_d; $C_\neg := \neg x^A_{a_1 a_2}$;
6. **if** C is a one-literal clause ($\neg x^A_{a_1 a_2}$) **then**
7. add $a_2 \prec^v_A a_1$ to O_d; $C_\neg := x^A_{a_1 a_2}$;
8. Reduce $\Phi(S_e)$ by using C and $C\neg$; /* see details below */
9. **return** O_d.

Fig. 6. Algorithm DeduceOrder

B. Deducing True Values

We now develop an algorithm that, given a valid specification $S_e = (I_t, \Sigma, \Gamma)$ of an entity e, deduces *true values* for as many attributes of e as possible. It finds a maximum partial order O_d such that $S_e \models O_d$, *i.e.,* (a) for all valid completions I^c_t of S_e, $O_d \subseteq I^c_t$ (Section IV), and (b) for tuples $t_1, t_2 \in I_e$ and $A_i \in R$, if $S_e \models t_1 \prec_{A_i} t_2$ then $t_1 \prec_{A_i} t_2$ is in O_d.

As an immediate corollary of Theorem 2, one can show that this problem is also coNP-complete, even when either Σ or Γ is fixed or absent. Thus we give a heuristics to strike a balance between its complexity and accuracy. The algorithm is based on the following lemma, which is easy to verify.

Lemma 6: *For* CNF *$\Phi(S_e)$ converted from a valid specification S_e, and for tuples t_1, t_2 in S_e with $t_1[A_i] = a_1$ and $t_2[A_i] = a_2$, $S_e \models t_1 \prec_{A_i} t_2$ iff $\Phi(S_e) \rightarrow x^{A_i}_{a_1 a_2}$ is a tautology, where $x^{A_i}_{a_1 a_2}$ is the variable denoting $a_1 \prec^v_{A_i} a_2$ in $\Phi(S_e)$.*

Here $\Phi(S_e) \rightarrow x^{A_i}_{a_1 a_2}$ indicates that for any truth assignment μ, if μ satisfies $\Phi(S_e)$, then $\mu(x^{A_i}_{a_1 a_2})$ is *true*, *i.e.,* the one-literal clause $x^{A_i}_{a_1 a_2}$ is implied by $\Phi(S_e)$, which in turn encodes S_e. Based on this, our algorithm checks one-literal clauses in $\Phi(S_e)$ one by one, and enriches O_d accordingly.

Algorithm. The algorithm, referred to as DeduceOrder, is given in Fig. 6. It first converts specification S_e to CNF $\Phi(S_e)$ (line 1; see Section V-A). For each literal C of the form $x^{A_i}_{a_1 a_2}$ or $\neg x^{A_i}_{a_1 a_2}$, it checks whether C is a clause in (implied by) $\Phi(S_e)$ (line 3), and if so, adds it to O_d (lines 4-7). It then reduces $\Phi(S_e)$ by using C and its negation $\neg C$ (line 8). That is, for each clause C' that contains C, the entire C' is removed since C' is *true* if C has to be satisfied (*i.e., true*). Similarly, for each clause C'' that contains $\neg C$, $\neg C$ is removed from C'', as $\neg C$ has to be *false*. The O_d is then returned (line 9).

Example 9: Consider E_2 in Fig. 2 and the constraints of Fig. 3, DeduceOrder finds O_d including: (1) *0* $\prec^v_{\mathsf{kids}}$ *2* by φ_4, (2) *working* $\prec^v_{\mathsf{status}}$ *retired* by φ_1, (3) *sailor* $\prec^v_{\mathsf{job}}$ *veteran*, *401*

$\prec^v_{\mathsf{AC}}$ *212* and *02840* $\prec^v_{\mathsf{zip}}$ *12404*, by (2) and φ_5, φ_6 and φ_7, respectively. A current tuple of George is then of the form (*George*, $x_{\mathsf{status}}, x_{\mathsf{job}}$, *2*, $x_{\mathsf{city}}, x_{\mathsf{AC}}, x_{\mathsf{zip}}, x_{\mathsf{county}}$), with variables.

Assume that the users assure that the true value of the attribute status is *retired*. Then the algorithm can deduce the following from the extended specification:

(a) x_{job}, x_{AC} and x_{zip} as *n/a*, *212* and *12404*, from tuple r_5 via currency constraints φ_5, φ_6 and φ_7, respectively;

(b) $x_{\mathsf{city}} = NY$, from the true value of AC (*i.e., 212* deduced in step (a) above) and the constant CFD ψ_2;

(c) x_{county} as *Accord*, from constraint φ_8 and the true values of city and zip deduced in steps (b) and (a), respectively.

The automated deduction tells us that the true value for George is t_2 = (George, retired, n/a, 2, NY, 212, 12404, Accord). This shows that *currency constraints* help *consistency* (from step (a) to (b)), and *vice versa* (*e.g.,* from (b) to (c)). □

Complexity. (1) It takes $O((|\Sigma| + |\Gamma|)|I_t|^2 + |I_t|^3)$ time to convert S_e into $\Phi(S_e)$ (line 1; see Section V-A). (2) The *total time* taken by the **while** loop (lines 3-8) is in $O((|\Sigma| + |\Gamma|)|I_t|^2 + |I_t|^3)$. Indeed, we maintain a hash-based index for literals C, in which the key is C and its value is the list of clauses in $\Phi(S_e)$ that contain C or $\neg C$. In the process, $\Phi(S_e)$ decreases monotonically. Hence in total it takes at most $O(|\Phi(S_e)|)$ time to reduce $\Phi(S_e)$ for *all literals*, where $|\Phi(S_e)|$ is bounded by $O((|\Sigma|+|\Gamma|)|I_t|^2+|I_t|^3)$. Taken together, the algorithm is in $O((|\Sigma| + |\Gamma|)|I_t|^2 + |I_t|^3)$ time.

By Lemma 6, one might want to compute a temporal order O'_d consisting of all such variables $x^{A_i}_{a_1a_2}$ that $\Phi(S_e) \wedge \neg x^{A_i}_{a_1a_2}$ is *not* satisfiable. That is, for each variable $x^{A_i}_{a_1a_2}$, we inspect $\Phi(S_e) \wedge \neg x^{A_i}_{a_1a_2}$ by invoking a SAT-solver. However, this approach, referred to as NaiveDeduce, calls the SAT-solver $|I_t|^2$ times. As will be seen in Section VI, DeduceOrder finds O_d with its accuracy comparable to O'_d, without incurring the cost of repeatedly calling a SAT-solver.

True value deduction. Using O_d found by DeduceOrder, one can deduce true attributes values as follows: a value a_1 is the true value of attribute A_i if for all values $a_2 \in \mathsf{adom}(I_e.A_i) \setminus \{a_1\}$, the currency order $a_2 \prec^v_A a_1$ is in O_d.

C. Generating Suggestions

True value deduction given above finds us the true values $\mathsf{V}_{\mathcal{B}}$ for a set of attributes $\mathcal{B} \subseteq R$. To identify the true value of the entity e specified by $S_e = (I_t, \Sigma, \Gamma)$, we compute a suggestion for a set of attributes $\mathcal{A} \subseteq R$ such that if the true values for $\mathcal{A}$ are validated, the true value of *the entire* e can be determined, even for attributes in $R \setminus (\mathcal{B} \cup \mathcal{A})$ (see Fig. 4). Below we first define suggestions and a notion of derivation rules. We then provide an algorithm for computing suggestions.

C.1. Suggestions and Derivation rules

For an attribute $A_i \in R \setminus \mathcal{B}$, we denote by $\mathsf{V}(A_i)$ the candidate true values for A_i, *i.e.,* for any $a_1 \in \mathsf{V}(A_i)$, there exists no $a_2 \in \mathsf{adom}(I_e.A_i) \setminus \{a_1\}$ such that $a_1 \prec^v_A a_2$ is in O_d. For a set X of attributes, we write $\mathsf{V}(X) = \{\mathsf{V}(A_i) \mid A_i \in X\}$.

Suggestion. A *suggestion* for S_e is a pair $(\mathcal{A}, \mathsf{V}(\mathcal{A}))$, where $\mathcal{A}=(A_1,\ldots,A_m)$ is a set of attributes of R such that $\mathcal{A} \cap \mathcal{B} = \emptyset$ and (1) there exist values $(a_1,\ldots,a_m)$ such that if $(a_1,\ldots,a_m)$ are validated as the true values of $\mathcal{A}$, then the true value $\mathsf{T}(S_e)$ of S_e exists; and (2) for all possible values $(a'_1,\ldots,a'_m)$ that satisfy condition (1), a'_i is in $\mathsf{V}(A_i)$ for $i \in [1, m]$.

Intuitively, condition 1 says that when the true values of $\mathcal{A}$ are validated, so is $\mathsf{T}(S_e)$. That is, the true values of attributes in $\mathcal{A}' = R \setminus (\mathcal{B} \cup \mathcal{A})$ can be deduced from $\mathsf{V}_{\mathcal{B}}$ and the true values of $\mathcal{A}$. Condition 2 says that $\mathsf{V}(\mathcal{A})$ gives "complete" candidates for the true values of $\mathcal{A}$ in their active domains.

One naturally wants a suggestion to be as "small" as possible, so that it takes minimal efforts to validate the true values of $\mathcal{A}$. This motivates us to study the *minimum suggestion problem*, which is to find a suggestion $(\mathcal{A}, \mathsf{V}(\mathcal{A}))$ with the minimum number $|\mathcal{A}|$ of attributes. Unfortunately, this problem is Σ^p_2-complete ($\mathsf{NP}^{\mathsf{NP}}$), which can be verified by reduction from the minimum coverage problem (Theorem 4).

Corollary 7: *The minimum suggestion problem for conflict resolution is* Σ^p_2*-complete.*

In light of the high complexity, we develop an effective heuristics to compute suggestions. To do this, we examine how true values are inferred via currency constraints and CFDs, by expressing them as a uniform set of rules.

Derivation rules. A *true-value derivation rule* for S_e has the form $(X, P[X]) \rightarrow (B, b)$, where (1) X is a set of attributes, B is a single attribute, and (2) b is a value that is either in $\mathsf{adom}(I_e.B)$ or in attribute B of some constant CFD; and (3) for each $A_i \in X$, $P[A_i]$ is drawn from $\mathsf{adom}(I_e.A_i)$. It assures if $P[X]$ is the true value of X, then b is the true value of B.

Derivation rules are computed from instance constraints $\Omega(S_e)$ of S_e, as shown below (to be elaborated shortly).

Example 10: Sample rules for George in Fig. 2 include:

n_1 : ({status}, {*retired*}) $\rightarrow$ (job, *veteran*)
n_2 : ({status}, {*retired*}) $\rightarrow$ (AC, *212*)
n_3 : ({status}, {*retired*}) $\rightarrow$ (zip, *12404*)
n_4 : ({city, zip}, {*NY*, *12404*}) $\rightarrow$ (county, *Accord*)
n_5 : ({AC}, {*212*}) $\rightarrow$ (city, *NY*)
n_6 : ({status}, {*unemployed*}) $\rightarrow$ (job, *n/a*)
n_7 : ({status}, {*unemployed*}) $\rightarrow$ (AC, *312*)
n_8 : ({status}, {*unemployed*}) $\rightarrow$ (zip, *60653*)
n_9 : ({city, zip}, {*Chicago*, *60653*}) $\rightarrow$ (county, *Bronzeville*)

Here rule n_5 is derived from CFD ψ_2, which states that if his true AC is *212*, then his true city must be *NY*. Rule n_1 is from tuple r_5 and constraint φ_5 (Fig. 3), which states that if his true status is *retired*, then his true job is *veteran*. Note that in n_1, status is instantiated with *retired*. Similarly, n_6 is derived from r_6 and φ_5; n_2 and n_3 (resp. n_7 and n_8) are derived from tuple r_5 (resp. r_6) and constraints φ_6 and φ_7, respectively; and n_4 (resp. n_9) is derived from r_5 (resp. r_6) and φ_8. □

To find a suggestion, we want to find a set $\mathcal{A}$ of attributes so that a maximum number of derivation rules can be applied to them at the same time, and hence, the true values of as

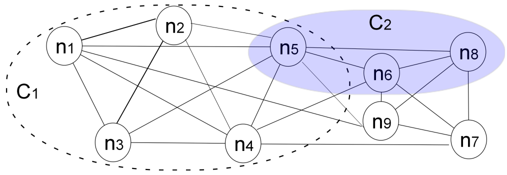

Fig. 7. Sample compatibility graph

many other attributes as possible can be derived from these rules. To capture this, we use the following notion.

Compatibility graphs. Consider a set Π of derivation rules. The *compatibility graph* $G(N, E)$ of Π is an *undirected* graph, where (1) each node x in N is a rule $(X_x, P_x[X_x]) \rightarrow (B_x, b_x)$ in Π, and (2) an edge (x, y) is in E iff $B_x \neq B_y$ and $P_x[X_{xy}] = P_y[X_{xy}]$, where $X_{xy} = (X_x \cup B_x) \cap (X_y \cup B_y)$.

Intuitively, two nodes are connected (*i.e.,* compatible) if their associated derivation rules derive different attributes (*i.e.,* $B_x \neq B_y$), and they agree on the values of their common attributes (*i.e.,* $P_x[X_{xy}] = P_y[X_{xy}]$). Hence these rules have no conflict and can be applied at the same time.

Example 11: The compatibility graph of the rules given in Example 10 is shown in Fig. 7. There is an edge (n_1, n_2) since their common attribute status has the same value *retired*; similarly for the other edges. In contrast, there is no edge between n_5 and n_7 since the values of their common attribute AC are different: *212* for n_5 and *312* for n_7. □

Observe that each clique $\mathcal{C}$ in the compatibility graph indicates a set of derivation rules that can be applied together. Let $\mathcal{A}'$ be the set of attributes whose true values can be derived from the rules in $\mathcal{C}$, if $\mathcal{C}$ and S_e have no conflicts (will be discussed shortly). To find a suggestion, we compute a maximum clique $\mathcal{C}$ from the graph, and define a suggestion as $(\mathcal{A}, \mathsf{V}(\mathcal{A}))$, where $\mathcal{A}$ consists of attributes in $R \setminus (\mathcal{A}' \cup \mathcal{B})$, and $\mathsf{V}(\mathcal{A})$ is the set of candidate true values for $\mathcal{A}$.

Example 12: Example 6 shows that for George (E_2), only the true values of name and kids are known, *i.e.,* $\mathcal{B} = \{\mathsf{name}, \mathsf{kids}\}$ and $\mathsf{V}_{\mathcal{B}} = $ (*George, 2*). To find a suggestion for George, we identify a clique $\mathcal{C}_1$ with five nodes n_1–n_5 in the compatibility graph of Fig. 7. Observe the following. (a) The values of job, AC and zip depend on the value of status by rules n_1, n_2 and n_3, respectively. (b) The AC in turn decides city by n_5. (c) From city and zip one can derive county by n_4. Hence, the set of attributes that can be derived from clique $\mathcal{C}_1$ is $\mathcal{A}' = \{\mathsf{job}, \mathsf{AC}, \mathsf{zip}, \mathsf{city}, \mathsf{county}\}$. This yields a suggestion $(\mathcal{A}, \mathsf{V}(\mathsf{status}))$, where $\mathcal{A} = R \setminus (\mathcal{A}' \cup \mathcal{B}) = \{\mathsf{status}\}$, and $\mathsf{V}(\mathsf{status}) = \{$*retired*, *unemployed*$\}$. As long as users identify the true value of status, the true value of George exists, and can be automatically deduced as described in Example 9. □

However, $\mathcal{C}$ and S_e may have conflicts, as illustrated below.

Example 13: Consider the clique $\mathcal{C}_2$ of Fig. 7 with three nodes n_5, n_6 and n_8. Observe the following: (a) n_5 indicates that *312* $\prec^v_{\mathsf{AC}}$ *212*, since *212* is *assumed* the latest AC value; whereas (b) n_6, n_8 and constraint φ_6 in Fig. 3 state that *312* is the latest AC value, *i.e., 212* $\prec^v_{\mathsf{AC}}$ *312*. These tell us that the values embedded in clique $\mathcal{C}_2$ may not lead to a valid completion for E_2, *i.e.,* $\mathcal{C}_2$ and S_e have conflicts. □

Algorithm Suggest

Input: A specification $S_e = (I_t, \Sigma, \Gamma)$, order O_d ($S_e \models O_d$), and $\mathsf{V}_{\mathcal{B}}$.
Output: A suggestion $(\mathcal{A}, \mathsf{V}(\mathcal{A}))$.

1. $\mathsf{V}(R) := \mathsf{DeriveVR}(I_t, O_d)$; $\quad\Omega(S_e) := \mathsf{Instantiation}(S_e)$;
2. $\Pi := \mathsf{TrueDer}(\Omega(S_e), \mathsf{V}(R))$; $\quad G := \mathsf{CompGraph}(\Pi, S_e)$;
3. $\mathcal{C} := \mathsf{MaxClique}(G)$; $\quad\mathcal{A} := \mathsf{GetSug}(S_e, \mathcal{C}, \mathsf{V}_{\mathcal{B}})$;
4. **return** $(\mathcal{A}, \mathsf{V}(\mathcal{A}))$;

Fig. 8. Algorithm Suggest

To handle conflicts between $\mathcal{C}$ and S_e, we use MaxSat to find a maximum subgraph $\mathcal{C}'$ of $\mathcal{C}$ that has no conflicts with S_e (MaxSat is to find a maximum set of satisfiable clauses in a Boolean formula; see *e.g.,* [23]). For instance, for clique $\mathcal{C}_2$ of Example 13, we use a MaxSat-solver [23] to identify clique $\mathcal{C}_2'$ with nodes n_6 and n_8, which has no conflicts with the specification for George. We then derive $\mathcal{A}' = \{\mathsf{job}, \mathsf{zip}\}$ from $\mathcal{C}_2'$. Since $\mathcal{B}$ is $\{\mathsf{name}, \mathsf{kids}\}$ (Example 12), we find $\mathcal{A} = R \setminus (\mathcal{A}' \cup \mathcal{B}) = \{\mathsf{status}, \mathsf{city}, \mathsf{AC}, \mathsf{county}\}$ for suggestion.

C.2. Computing Suggestions

We now present the algorithm for computing suggestions, referred to as Suggest and shown in Fig. 8. It takes as input a specification S_e of e, partial orders O_d deduced from S_e ($S_e \models O_d$, by Algorithm DeduceOrder), and the set $\mathsf{V}_{\mathcal{B}}$ of validated true values. It finds and returns a suggestion $(\mathcal{A}, \mathsf{V}(\mathcal{A}))$.

Algorithm Suggest first computes candidate true values for all attributes whose true values are yet unknown (line 1). It then deduces a set of derivation rules from instance constraints $\Omega(S_e)$ (line 1) of S_e (line 2; as illustrated in Example 10). Based on these derivation rules, it builds a compatibility graph (line 2; see Example 11) and identifies a maximum clique $\mathcal{C}$ in the graph (line 3). Finally, it generates a suggestion using the clique (line 3; see Examples 12 and 13).

We next present the procedures used in the algorithm.

<u>DeriveVR</u>: For each $A \in R$ not in $\mathsf{V}_{\mathcal{B}}$, it computes $\mathsf{V}(A)$. Initially $\mathsf{V}(A)$ takes the active domain $\mathsf{adom}(I_e.A)$. It then removes all $a_1 \in \mathsf{adom}(I_e.A)$ from $\mathsf{V}(A)$ if there exists $a_2 \in \mathsf{adom}(I_e.A) \setminus \{a_1\}$ such that $a_1 \prec^v_A a_2$ is in the deduced O_d, as a_2 is more current than a_1 in A. It takes $O(|I_t|^2)$ time with an index, since it checks at most $|O_d|$ orders, and $|O_d| \leq |I_t|^2$.

<u>TrueDer</u>: Given $\Omega(S_e)$, it deduces a set Π of derivation rules.
(1) From a constant CFD $(t_p[X_\varphi] \rightarrow t_p[B_\varphi])$. We add $(X_\varphi, t_p[X_\varphi]) \rightarrow (B_\varphi, t_p[B_\varphi])$ to Π, provided that $t_p[A] \in \mathsf{V}[A]$ for each $A \in X_\varphi \cap \mathcal{B}$, *i.e.,* when the values of the CFD have no conflict with those validated true values.
(2) From those instance constraints in $\Omega(S_e)$ that represent currency constraints and currency orders in S_e. It deduces derivation rules of the form $(X, P(X)) \rightarrow (B, b)$, for each attribute B whose true value is unknown and for each $b \in \mathsf{V}(B)$, if such a rule exists. While it is prohibitively expensive to enumerate all these rules, we use a heuristics to find a set of derivation rules in $O(|\Omega(S_e)|)$ time as follows:
(i) for each B and $b \in \mathsf{V}(B)$, let $U_{(B,b)} = \{b_i \prec^v_B b \mid b_i \in \mathsf{V}(B) \setminus \{b\}\}$, *i.e.,* b is assumed the true value of B;
(ii) it partitions $\Omega(S_e)$ based on $U_{(B,b)}$: let $\Omega_{(B,b)}$ consist of $\phi \in \Omega(S_e)$, where ϕ is of the form $\omega \rightarrow b_i \prec^v_B b$; note

that each ϕ appears in at most one of the partitions;

(iii) for each $b_i \in U_{(B,b)}$, it picks $\phi = \omega \to b_i \prec^v_B b$ from $\Omega_{(B,b)}$ if it exists; it includes those attributes of ω in X and their instantiations in $P(X)$, until all b_i's in $U_{(B,b)}$ are covered by such a ϕ (see Example 10 for how $P(X)$ is populated). Note that $|X| \leq |R|$.

The procedure is in $O((|\Sigma| + |\Gamma|)|I_t|^2 + |I_t|^3)$ time. Indeed, for (1), it is bounded by $O(|\Gamma|)$; and for (2), since $U_{(B,b)}$'s are disjoint, $\Omega_{(B,b)}$'s partition $\Omega(S_e)$, and each ϕ in $\Omega(S_e)$ is used at most once, the cost is in $O((|\Sigma| + |\Gamma|)|I_t|^2 + |I_t|^3)$.

CompGraph: Given rules Ω, it generates their compatibility graph $G(N, E)$ (see Example 11). The procedure takes at most $O(|\Pi|^2)$ time, where $|\Pi|$ is no larger than $|R||I_t|$.

MaxClique: It computes a maximum clique $\mathcal{C}$ of $G(N, E)$ (an NP-complete problem). Several tools have been developed for computing maximum cliques, with a good approximation bound (*e.g.,* [16]). We use one of these tools as MaxClique.

GetSug: Given clique $\mathcal{C}$, it computes a suggestion. It first finds the maximal subgraph $\mathcal{C}'$ of $\mathcal{C}$ that has no conflicts with S_e, by using an efficient MaxSat-solver [23] (see Example 13). It then derives a set $\mathcal{A}'$ of attributes from $\mathcal{C}'$ (see Example 12). Finally, it returns $(\mathcal{A}, \mathsf{V}(A))$, where $\mathcal{A} = R \setminus (\mathcal{A}' \cup \mathcal{B})$, and $\mathcal{B}$ is the set of attributes with validated true values $\mathsf{V}_{\mathcal{B}}$. Note that the input to the MaxSat-solver is no larger than $|R|^2|I_t|^2$.

Correctness. Algorithm Suggest guarantees to generate a suggestion $(\mathcal{A}, \mathsf{V}(\mathcal{A}))$. Indeed, (1) the clique $\mathcal{C}'$ revised by MaxSat has no conflicts with S_e, and thus $\mathcal{C}'$ and S_e warrant to have a valid completion I^c_t. Let $t_c = \mathsf{LST}(I^c_t)$. If $\mathsf{V}(\mathcal{A})$ are validated for $\mathcal{A}$, then t_c must be the *true value* $\mathsf{T}(S_e)$ of S_e, since $t_c[\mathcal{B}] = \mathsf{V}_{\mathcal{B}}$ remains unchanged for all valid completions of S_e, and $t_c[\mathcal{A}']$ is uniquely determined by $t_c[\mathcal{A}]$ and $\mathsf{V}_{\mathcal{B}}$ by the construction. (2) All possible true values for $\mathcal{A}$ from their active domains are already included in $\mathsf{V}(\mathcal{A})$.

VI. Experimental Study

We conducted experiments with both real-life and synthetic data. We evaluated the accuracy and scalability of (1) IsValid for validating a specification, (2) DeduceOrder for deducing true values, (3) Suggest for computing suggestions, and (4) the overall performance of conflict resolution supporting (1-3).

Experimental data. We used two real-life datasets (NBA and CAREER) and synthetic data (Person). Constraints were discovered using profiling algorithms [5], [14], and examined manually. Timestamps for the datasets were either missing (for CAREER and Person) or incomplete (NBA). We assumed *empty currency orders* in all the experiments even when partial timestamps were given. The available (incomplete) timestamps were *only* used when we verified the derived true entity values.

NBA *player statistics*. This dataset was retrieved from (1) *http://databasebasketball.com/*, (2) *http://www.infochimps.com/marketplace*, and (3) *http://en.wikipedia.org/wiki/List_of_National_Basketball_Association_arenas*. It consists of three tables: (a) Player (from sources 1 and 3) contains information about players, identified by player id (pid). (b) Stat (from 1) includes the statistics of these players from 2005/2006 to the 2010/2011 season. (c) Arenas (from 3) records the historical team names and arenas of each team. We created a table, referred to as NBA, by first joining Player and Stat via *equi-join* on the pid attribute, and then joining Arenas via *equi-join* on the team attribute. The NBA table consists of 19573 tuples for 760 entities (*i.e.,* players). Its schema is (pid, name, true name, team, league, tname, points, poss, allpoints, min, arena, opened, capacity, city). When producing the NBA table we took care of the attributes containing multiple values for a player, *e.g.,* multiple teams for the same player, and multiple teams for one arena. We ensure that only one attribute value (*e.g.,* team) appears in any tuple. Only data from (1) and (3) carries (partial) timestamps. Therefore, the true values of entities in the NBA table *cannot* be directly derived.

The number of tuples pertaining to an entity ranges from 2 to 136, about 27 in average. We consider *entity instances*, *i.e.,* tuples referring to the same entity, which are *much smaller* than a database. We found 54 currency constraints: 15 for team names (tname) as shown by φ_1 below; 32 for arena, similar to φ_2; and 4 (resp. 3) for attribute allpoints that were scored since 2005 (resp. arena), similar to φ_3 (resp. φ_4), where B ranges over points, poss, min and tname (resp. opened, capacity and years). We deduced 58 constant CFDs, *e.g.,* the ψ_1 below. Note that some rules are derived automatically, while the others are designed manually based on the semantics of the data.

φ_1: $\forall t_1, t_2$ (t_1[tname] = "New Orleans Jazz" $\wedge$ t_2[tname] = "Utah Jazz" $\to t_1 \prec_{\mathsf{tname}} t_2$);
φ_2: $\forall t_1, t_2$ (t_1[arena] = "Long Beach Arena" $\wedge$ t_2[arena] = "Staples Center" $\to t_1 \prec_{\mathsf{arena}} t_2$);
φ_3: $\forall t_1, t_2$ (t_1[allpoints] $<$ t_2[allpoints] $\wedge$ $t_1[B] \neq t_2[B] \to t_1 \prec_{\mathsf{B}} t_2$)
φ_4: $\forall t_1, t_2$ ($t_1 \prec_{\mathsf{arena}} t_2 \wedge t_1[B] \neq t_2[B] \to t_1 \prec_{\mathsf{B}} t_2$)
ψ_1: (arena = "United Center" $\to$ city = "Chicago, Illinois")

(2) CAREER. The data was retrieved *as is* from the link *http://www.cs.purdue.edu/commugrate/data/citeseer*. Its schema is (first name, last name, affiliation, city, country). We chose 65 persons from the dataset, and for each person, we collected all of his/her publications, one tuple for each. *No reliable timestamps* were available for this dataset.

The number of tuples pertaining to an entity ranges from 2 to 175, about 32 in average. We derived 503 currency constraints: if two papers A and B are by the same person and A cites B, then the affiliation and address (city and country) used in paper A are more current than those used in paper B. We also deduced a single CFD of the form: (affiliation $\to$ city, country), but with 347 patterns with different constants.

The constraints for each dataset (NBA and CAREER) have essentially *the same form*, and *only differ* in their constants, *i.e.,* the number of constraints with different forms is *small*.

(3) Person *data*. The synthetic data adheres to the schema given in Table 2. We found 983 currency constraints (of *the same form* but with distinct constant values for status, job and kid) and a single CFD AC $\to$ city with 1000 patterns (counted as distinct constant CFDs), similar to those in Table 3. The data generator used two parameters: n denotes the number of

entities, and s is the size of *entity instances* (the number of tuples pertaining to an entity). For each entity, it first generated a true value t_c, and then produced a set E of tuples that have conflicts but do not violate the currency constraints; we treated $E \setminus \{t_c\}$ as the entity instance. We generated n = 10k entities, with s from 1 to 10k. We used *empty* currency orders here.

Algorithms. We implemented the following algorithms in *C++*: (a) IsValid (Section V-A): it calls MiniSat [18] as the SAT-solver; (b) DeduceOrder and NaiveDeduce: NaiveDeduce repeatedly invokes MiniSat [18], as described in Section V-B; and (c) Suggest: it uses MaxClique [16] to find a maximal clique, and MaxSat-solver [23] to derive a suggestion (Section V-C). We simulated user interactions by providing true values for *a subset of* suggested attributes, some with new values, *i.e.,* values not in the active domain. We also implemented (d) Pick, a traditional method that randomly takes a value [4]; to favor Pick, we picked a value from those that are not less current than any other values, based on currency constraints $\forall t_1, t_2(\omega \rightarrow t_1 \prec_A t_2)$ in which ω is a conjunction of comparison predicates only, *e.g.,* φ_1-φ_3 above.

Accuracy. To measure the quality of suggestions, we used F-measure (*http://en.wikipedia.org/wiki/F-measure*):

$$\text{F-measure} = 2 \cdot (\text{recall} \cdot \text{precision})/(\text{recall} + \text{precision}).$$

Here precision is the ratio of the number of values correctly deduced to the total number of values deduced; and recall is the ratio of the number of values correctly deduced to the total number of attributes with conflicts or stale values.

All experiments were conducted on a Linux machine with a 3.0GHz Intel CPU and 4GB of Memory. Each experiment was repeated 5 times, and the average is reported here.

Experimental results. We next present our findings. Due to the small size of the CAREER data for each entity, experiments conducted on it took typically less than 10 milliseconds (ms). Hence we do not report its result in the efficiency study.

Exp-1: Validity checking. We first evaluated the scalability of IsValid. The average time taken by entity instances of various sizes is reported in Fig. 9(a), where the lower x-axis shows the sizes of NBA, and the upper x-axis is for Person data. The results show that IsValid suffices to validate specifications of a reasonably large size. For example, it took 220 ms for NBA entity instances of 109-135 tuples and 112 constraints, with 14 attributes in each tuple. For Person, it took an average of 4.7 seconds on entities of 8k-10k tuples and 1983 constraints.

We also find IsValid accurate (not shown for the lack of space): specifications reported (in)valid are indeed (in)valid.

Exp-2: Deducing true values. We next evaluated the performance of algorithms DeduceOrder and NaiveDeduce. The results on both NBA and Person data are reported in Fig. 9(b), which tell us the following: (a) DeduceOrder scales well with the size of entity instances, and (b) DeduceOrder substantially outperforms NaiveDeduce on both datasets, for reasons given in Section V-B. Indeed, DeduceOrder took 51 ms on NBA entity instances with 109-135 tuples, and 914 ms on Person entities of 8k-10k tuples; in contrast, NaiveDeduce spent 13585 ms and over 20 minutes (hence not shown in Fig. 9(b)) on the same datasets, respectively.

We also find that DeduceOrder derived as many true values as NaiveDeduce on both datasets (not shown). This tells us that DeduceOrder can efficiently deduce true values on large entity instances without compromising the accuracy.

Exp-3: Suggestions for user interactions. We evaluated the accuracy of suggestions generated from currency constraints Σ and CFDs Γ put together. The results on NBA, CAREER and Person are given in Figures 9(e), 9(i) and 9(m), respectively, where the x-axis indicates the rounds of interactions, and the y-axis is the percentage of true attribute values deduced.

These results tell us the following. (a) Few rounds of interactions are needed to find all the true attribute values for an entity: at most 2, 2 and 3 rounds for NBA, CAREER and Person data, respectively. (b) A large part of true values can be *automatically deduced* by means of currency and consistency inferences: 35%, 78% and 22% of true values are identified from Σ + Γ *without user interaction*, as indicated by the 0-interaction in Figures 9(e), 9(i) and 9(m), respectively.

Impact of $|\Sigma|$ and $|\Gamma|$. To be more precise when evaluating the accuracy, we use F-measure, which combines precision and recall, and take the cases of using $|\Gamma|$ only or $|\Sigma|$ only into consideration. Figures 9(f)–9(h), 9(j)–9(l) and 9(n)–9(p) show the results for NBA, CAREER and Person, respectively, when varying both $|\Sigma|$ and $|\Gamma|$, $|\Sigma|$ only, and varying $|\Gamma|$ alone, respectively. The x-axis shows the percentage of Σ or Γ used, and the y-axis shows the corresponding F-measure values.

These results tell us the following. (a) As shown in Figures 9(f), 9(j) and 9(n), our method substantially outperforms the traditional method Pick, by 201% in average on all datasets, even when we favor Pick by allowing it to capitalize on currency orders. This verifies that data currency and consistency can significantly improve the accuracy of conflict resolution. (b) When Σ and Γ are taken together, the F-measure value is up to 0.930 for NBA (Fig. 9(f), the top right point), 0.958 for CAREER (Fig. 9(j)), and 0.903 for Person (Fig. 9(n)), in contrast to 0.830 (=in Fig. 9(g), 0.907 in Fig. 9(k), and 0.826 in Fig. 9(o), respectively, when Σ is used alone, and as opposed to 0.210 in Fig. 9(h), 0.741 in Fig. 9(l), and 0.234 in Fig. 9(p), respectively, with Γ only. These further verify that the inferences of data currency and consistency should be *unified* instead of taking separately. (c) The more currency constraints and/or CFDs are available, the higher the F-measure is, as expected. (d) The two curves for the 2- and 1-interaction overlap in Figures 9(f)–9(h) for NBA, 2- and 1-interaction in Figures 9(j)–9(l) for CAREER, and 3- and 2-interaction in Figures 9(n)–9(p) for Person. These indicate that the users must provide true values for those attributes that we do not have enough information to deduce their true values.

Exp-4: Efficiency. The overall performance for resolving conflicts in the NBA (resp. Person) data is reported in Fig. 9(c) (resp. Fig. 9(d)). Each bar is divided into the elapsed time taken by (a) validity checking, (b) true value deducing, and (c) suggestion generating, including computing the maximal

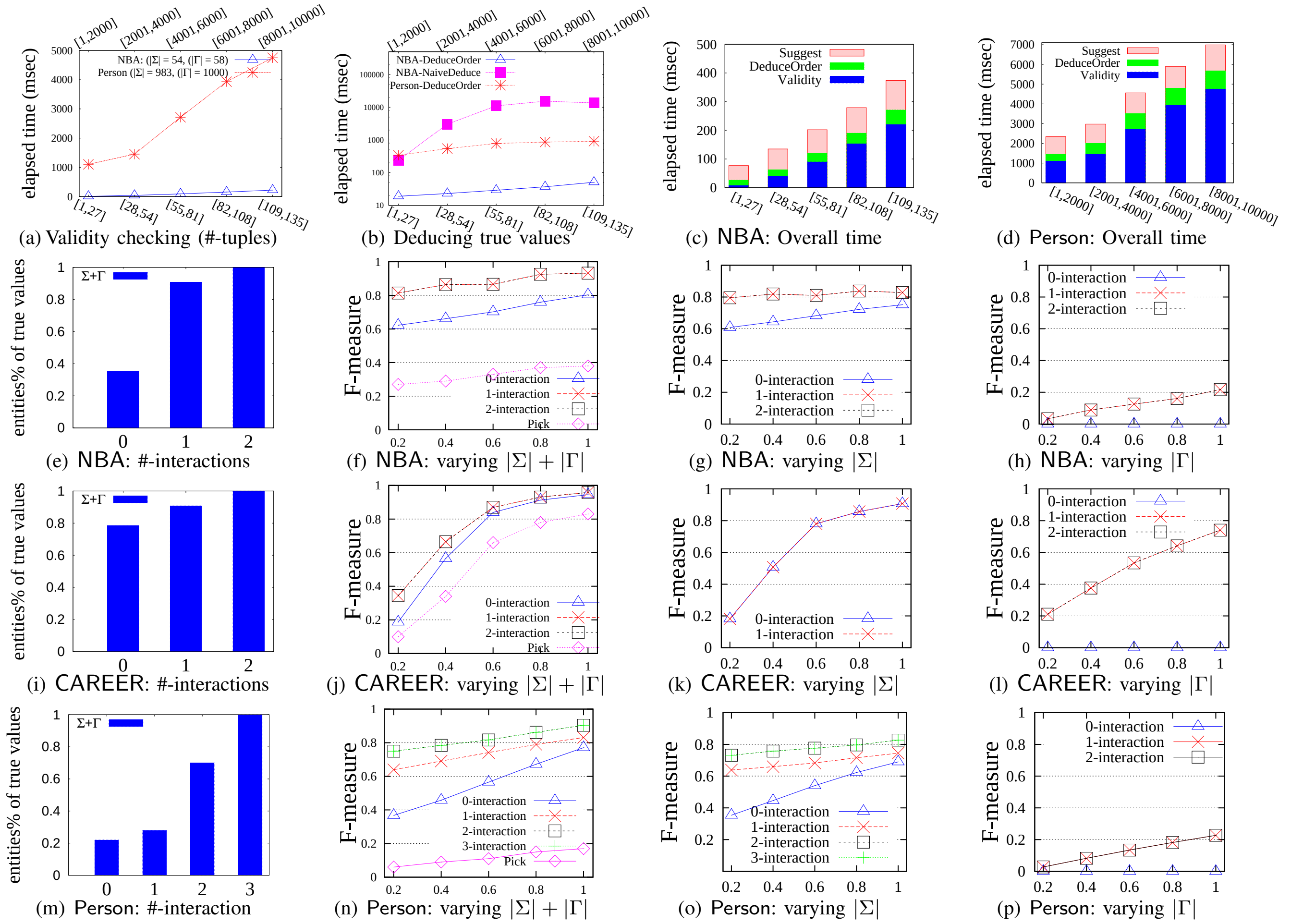

Fig. 9. Experimental results

clique and running MaxSat. The result shows that conflict resolution can be conducted efficiently in practice, *e.g.,* each round of interactions for NBA took 380 ms. Here validating specifications takes most time, dominated by the cost of SAT-solver, while deducing true values takes the least time.

Summary. We find the following. (a) Conflict resolution with data currency and consistency substantially outperforms the traditional method Pick, by 201%. (b) It is more effective to unify the inferences of data currency and consistency than treating them independently. Indeed, when Σ and Γ are taken together, the F-measure improves over Σ only and Γ only by 11% and 236%, respectively. (c) Our conflict resolution method is efficient: it takes less than 0.5 second on the real-life datasets even with interactions. (d) Our method scales well with the size of entities and the number of constraints. Indeed, it takes an average of 7 seconds to resolve conflicts in Person entity instances of 8k-10k tuples, with 1983 constraints. (e) At most 2-3 rounds of interactions are needed for all datasets.

VII. Conclusion

We have proposed a model for resolving conflicts in entity instances, based on both data currency and data consistency. We have also identified several problems fundamental to conflict resolution, and established their complexity. Despite the inherent complexity of these problems, we have introduced a framework for conflict resolution, along with practical algorithms supporting the framework. Our experiments have verified that our methods are effective and efficient.

We are now exploring more efficient algorithms for generating suggestions, and testing them with data in various domains. We are also extending our framework by allowing users to edit constraints, and by improving the accuracy when users do not have sufficient currency knowledge about their data.

References

[1] M. Arenas, L. E. Bertossi, and J. Chomicki. Consistent query answers in inconsistent databases. In *PODS*, 1999.

[2] L. Bertossi. *Database Repairing and Consistent Query Answering*. Morgan & Claypool Publishers, 2011.

[3] A. Biere, M. Heule, H. van Maaren, and T. Walsh, editors. *Handbook of Satisfiability*, volume 185 of *Frontiers in Artificial Intelligence and Applications*. IOS Press, 2009.

[4] J. Bleiholder and F. Naumann. Data fusion. *ACM Comput. Surv.*, 41(1), 2008.

[5] F. Chiang and R. Miller. Discovering data quality rules. In *VLDB*, 2008.

[6] J. Chomicki and D. Toman. Time in database systems. In *Handbook of Temporal Reasoning in Artificial Intelligence*. Elsevier, 2005.

[7] G. Cong, W. Fan, F. Geerts, X. Jia, and S. Ma. Improving data quality: Consistency and accuracy. In *VLDB*, 2007.

[8] U. Dayal. Processing queries over generalization hierarchies in a multidatabase system. In *VLDB*, 1983.

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

[10] X. Dong and F. Naumann. Data fusion - resolving data conflicts for integration. In *VLDB*, 2009.

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

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

[13] W. Fan, F. Geerts, X. Jia, and A. Kementsietsidis. Conditional functional dependencies for capturing data inconsistencies. *TODS*, 33(1), 2008.

[14] W. Fan, F. Geerts, J. Li, and M. Xiong. Discovering conditional functional dependencies. *TKDE*, 23(5):683–698, 2011.

[15] W. Fan, F. Geerts, and J. Wijsen. Determining the currency of data. In *PODS*, 2011.

[16] U. Feige. Approximating maximum clique by removing subgraphs. *SIAM J. Discret. Math.*, 18, February 2005.

[17] A. Galland, S. Abiteboul, A. Marian, and P. Senellart. Corroborating information from disagreeing views. In *WSDM*, 2010.

[18] E. Giunchiglia and A. Tacchella, editors. *Theory and Applications of Satisfiability Testing*, SAT, 2004.

[19] S. Greco, C. Sirangelo, I. Trubitsyna, and E. Zumpano. Preferred repairs for inconsistent databases. In *IDEAS*, 2003.

[20] P. Li, X. Dong, A. Mauricio, and D. Srivastava. Linking temporal records. *PVLDB*, 2011.

[21] A. Motro and P. Anokhin. Fusionplex: resolution of data inconsistencies in the integration of heterogeneous information sources. *Information Fusion*, 7(2), 2006.

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

[23] B. Selman and H. Kautz. Walksat home page, 2004. *http://www.cs.washington.edu/homes/kautz/walksat/*.

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

[25] L. J. Stockmeyer. The polynomial-time hierarchy. *Theore. Comput. Sci*, 3(1), 1976.

[26] J. Widom. Trio: A system for integrated management of data, accuracy, and lineage. In *CIDR*, 2005.

[27] M. Yakout, A. K. Elmagarmid, J. Neville, and M. Ouzzani. GDR: a system for guided data repair. In *SIGMOD*, 2010.

[28] X. Yin, J. Han, and P. S. Yu. Truth discovery with multiple conflicting information providers on the web. *TKDE*, 20(6), 2008.

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