Temporal Data Exchange

Ladan Golshanara¹, Jan Chomicki¹, and Wang-Chiew Tan²

¹ State University of New York at Buffalo, NY, USA
ladangol@buffalo.edu, chomicki@buffalo.edu

² Recruit Institute of Technology and UC Santa Cruz, CA, USA
tan@cs.ucsc.edu

Abstract

Data exchange is the problem of transforming data that is structured under the source schema into data structured under another schema, called the target schema, so that both source and target data satisfy the relationship between the schemas. Many applications such as planning, scheduling, medical and fraud detection systems, require data exchange in the context of temporal data. Even though the formal framework of data exchange for relational database systems is well-established, it does not immediately carry over to the setting of temporal data, which necessitates reasoning over unbounded periods of time.

In this work, we study data exchange for temporal data. We first motivate the need for two views of temporal data: the concrete view, which depicts how temporal data is compactly represented and on which implementations are based, and the abstract view, which defines the semantics of temporal data. We show how the framework of data exchange can be systematically extended to temporal data. The core of our framework consists of two new chase algorithms: the abstract chase over an abstract temporal instance and the concrete chase over a concrete temporal instance. We show that although the two chase procedures operate over fundamentally different views of temporal data, the result of the concrete chase is semantically aligned with the result of the abstract chase. To obtain the semantic alignment, the nulls (which are introduced by data exchange and model incompleteness) in both the concrete view and the abstract view are annotated with temporal information. Furthermore, we show that the result of the concrete chase provides a foundation for query answering. We define naïve evaluation on the result of the concrete chase and show it produces certain answers.

Keywords and phrases Data Exchange, Temporal Database, Chase, Incomplete information, Abstract view, Concrete view

Introduction

Temporal data refers to historical data or data that is dated. Temporal data is needed by many organizations and individuals to support audit trails. With temporal data one can represent when a fact is true and for how long [6]. The temporality of facts is also critical in diverse domains, from medical diagnosis to assessing the changing business conditions of companies [17]. In fact, temporal database features, which were recently added to the SQL-2011 standard [12], have already been adopted by major database management systems such as DB2, Oracle, and Teradata.

Temporal database is a relational database where there is at least one temporal attribute associated with each relation schema. The need to store information that span over long periods of time necessitates that temporal attribute values be intervals so that multiple (sometimes infinitely many) time points can be compactly represented. For example, the fact that Ada worked in IBM between 2010 and 2013 is usually represented as (Ada, IBM,
(2010, 2014) where (2010, 2014) is a clopen interval that denotes the years 2010, 2011, 2012, and 2013. The fact that Ada has worked in Intel since then can be represented as (Ada, Intel, [2014, ∞)). An infinite interval, such as [2014, ∞), is a useful abstraction when the endpoint is not provided. However, it is simpler and more natural to view information at every snapshot, i.e., at discrete time points. For these reasons, prior work on temporal databases [5, 10, 19] has provided two views of temporal data: the concrete temporal view (or concrete view in short) and the abstract temporal view (or abstract view in short). In the concrete view the same data over multiple time points is compactly stored using time intervals, such as the examples with Ada above. This view is an extension of the relational model where the temporal attribute takes clopen intervals as values. On the other hand, in the abstract temporal view (or abstract view in short) there is a fact representing the data at every time point when it is true. For example, there are four abstract tuples (Ada, IBM, 2010), (Ada, IBM, 2011), (Ada, IBM, 2012), (Ada, IBM, 2013) corresponding to the concrete tuple (Ada, IBM, [2010, 2014]), and infinitely many abstract tuples (Ada, Intel, 2014), (Ada, Intel, 2015), ... that correspond to the concrete tuple (Ada, Intel, [2014, ∞)). The abstract view is based on the relational model with infinite relations. The semantic alignment between concrete and abstract views is formally defined as the semantic mapping in Section 3. Note that SQL:2011 [12] implicitly supports both views. It provides features for defining tables with interval endpoint attributes, while the semantics of integrity constraints is defined with respect to time points.

Data exchange [7] refers to the problem of translating data that conforms to one schema (called the source schema RS) into data that conforms to another schema (called the target schema RT) given a specification of the relationship between the two schemas. This relationship is specified by means of a schema mapping consisting of a set of source-to-target tuple generating dependencies (s-t tgds) and a set of tuple generating dependencies (tgds) and equality generating dependencies (egds) on the target schema. Given a schema mapping and a source instance I, the goal of data exchange is to materialize a target instance J that satisfies the specification (i.e., (I, J) satisfies s-t tgds and J satisfies tgds and egds). Such an instance J is called a solution for I w.r.t the given schema mapping. For a given source instance, there may be no solution since there may not exist a target instance that satisfies the specification. On the other hand, there may be many solutions. It was shown in [7] that among all solutions of a given source instance, the universal solutions are the preferred solutions because they are the most general. In [7], the chase procedure is used to find a universal solution. Universal solutions can be used to determine certain answers to the unions of conjunctive queries posed over a target schema. Certain answers to a query Q consist of all tuples that will be in the answer of Q in any arbitrary solution for a source instance w.r.t a schema mapping.

In this paper, we study the challenges that arise when we consider temporal data in the framework of data exchange. We examine the most basic case where we assume that every relation in a source instance has exactly one temporal attribute and source instances are complete (i.e., have no unknown values). In our framework, the temporal schema mapping \( \mathcal{M} \) is a quadruple \( \mathcal{M} = (R_S, R_T, \Sigma_{st}, \Sigma_k) \), where \( \Sigma_{st} \) is a set of source-to-target temporal tuple generating dependencies (s-t ttgds) between the source and target schema, and \( \Sigma_k \) is a set of temporal key constraints (tkcs) [4] over the target schema. Temporal key constraints are fundamental for temporal applications. We introduce two chase algorithms: abstract chase for the abstract view and concrete chase for the concrete view. The chase procedure in [7] is a sequential process and assumes finite relations. This assumption holds for the concrete view (modulo modifications to the chase procedure to manage time intervals) but not for
the abstract view which might contain infinite relations. To cope with infinite abstract
relations, chase steps are applied in parallel in the abstract chase algorithm. Both proposed
chase algorithms have two rounds: An s-t ttgd round followed by a tkc round. The s-t ttgd
round aggregates all s-t ttgd chase steps that can be applied on its input while the tkc round
aggregates all tkc chase steps. If the tkc round is successful, the result of the abstract chase
algorithm (resp. concrete chase algorithm) satisfies the given temporal schema mapping \( M \).
Otherwise, both chase algorithms fail. Note that s-t ttgd and tkc chase steps are extensions
of the chase steps in \[7\] for temporal databases.

The concrete chase handles clopen intervals by normalizing the source instance. Intuitively,
normalization makes the clopen intervals behave as time points. Normalization was previously
used in the context of query answering in temporal databases \[13, 20\] on tuples with the same
schema that agree on non-temporal attribute values. The proposed normalization operation
in this paper does not have these restrictions. As a result of data exchange, unknown values
may occur in target instances. More precisely, at each s-t ttgd chase step in the abstract chase
and concrete chase algorithms fresh nulls are generated. As mentioned before, a concrete fact
is a compact representation of a set of abstract facts; therefore, a fresh null generated in a
concrete view should represent a set of distinct fresh nulls in the corresponding abstract view.
In order to keep the correspondence between the concrete and abstract view of temporal data
in the presence of unknown values generated by data exchange, the nulls in a concrete target
instance are annotated with the clopen interval of the tuple they occur in, e.g. \( N_s^{[2008,∞)} \) is an
interval-annotated null in a concrete tuple, while the nulls in an abstract target instance are
annotated with the time point of the tuple they occur in, e.g. \( N^t_0 \) is a point-annotated null
in an abstract tuple.

For example, consider the relational schema Emp(Name, Position, Company, Year).
The concrete tuple \((Ada, N^{[2008,∞)}, IBM, [2008,∞])\) contains an annotated null \( N^{[2008,∞)} \) and
the tuple denotes that the position of Ada at IBM is unknown and the position is
possibly different every year since 2008. This tuple corresponds to the abstract tuples
\((Ada, N^{[2008,∞)}, IBM, 2008), (Ada, N^{[2008,∞]}, IBM, 2009), \ldots\). Labeled nulls which are used in rela-
tional data exchange to model incompleteness are not sufficient to show the connection
between the concrete and abstract views after data exchange. This is shown in the proof of
Theorems \[6\] and \[8\].

\textbf{Example 1.} Suppose we have the following relations in the source schema that rep-
resents the employment histories of persons: \( \text{Employee1}(Name, Company, Time) \) and \( \text{Employee2}(Name, Position, Dept, Time) \).

A concrete source instance is shown in Figure \[1\] while the corresponding abstract instance
is shown in Figure \[2\] Only the last two digits of years are shown. We want to move the
data to a bigger schema with two relations: \( \text{Emp}(Name, Position, Company, Time) \), and \( \text{Sal}(Name, Position, Salary, Time) \).
Suppose we have the following s-t ttgds:

\[ \forall n \forall c \forall t \text{Employee1}(n, c, t) \rightarrow \exists p \exists s \text{Emp}(n, p, c, t) \land \text{Sal}(n, p, s, t) \]

\[ \forall n \forall p \forall d \forall t \text{Employee2}(n, p, d, t) \rightarrow \exists c \exists s \text{Emp}(n, p, c, t) \land \text{Sal}(n, p, s, t) \]

and the tkcs:

\[ \forall n \forall p_1 \forall p_2 \forall c_1 \forall c_2 \forall t \text{Emp}(n, p_1, c_1, t) \land \text{Emp}(n, p_2, c_2, t) \rightarrow p_1 = p_2 \land c_1 = c_2 \]

\[ \forall n \forall p_1 \forall p_2 \forall s_1 \forall s_2 \forall t \text{Sal}(n, p_1, s_1, t) \land \text{Sal}(n, p_2, s_2, t) \rightarrow p_1 = p_2 \land s_1 = s_2 \]
Figure 1 Concrete source instance

<table>
<thead>
<tr>
<th>Name</th>
<th>Company</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ada</td>
<td>IBM</td>
<td>(08, 11)</td>
</tr>
<tr>
<td>Ada</td>
<td>Intel</td>
<td>(11, 13)</td>
</tr>
</tbody>
</table>

Employee2

<table>
<thead>
<tr>
<th>Name</th>
<th>Position</th>
<th>Dept</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ada</td>
<td>Developer</td>
<td>Computer</td>
<td>(08, 10)</td>
</tr>
<tr>
<td>Ada</td>
<td>DBA</td>
<td>Computer</td>
<td>(10, 11)</td>
</tr>
</tbody>
</table>

Figure 2 Abstract source instance

<table>
<thead>
<tr>
<th>Name</th>
<th>Company</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ada</td>
<td>IBM</td>
<td>08</td>
</tr>
<tr>
<td>Ada</td>
<td>IBM</td>
<td>09</td>
</tr>
<tr>
<td>Ada</td>
<td>IBM</td>
<td>10</td>
</tr>
<tr>
<td>Ada</td>
<td>Intel</td>
<td>11</td>
</tr>
<tr>
<td>Ada</td>
<td>Intel</td>
<td>12</td>
</tr>
</tbody>
</table>

Figure 3 and Figure 5 respectively show the solutions for the concrete and abstract source instances w.r.t the above temporal schema mapping. These solutions can be obtained using the concrete and abstract chase algorithms, respectively. Note that both solutions satisfy the given temporal schema mapping. Moreover, if two annotated nulls are syntactically the same in an abstract or a concrete instance (e.g. \( N^{[11,13)} \) in tables \( \text{Emp} \) and \( \text{Sal} \) in Figure 3 or \( J^{11} \) and \( K^{12} \) in Figure 5), they are generated by applying an s-t ttgd with an existentially quantified variable shared between the atoms on its right hand side. Figure 4 is the abstract view of the concrete solution in Figure 3. Observe that Figure 4 and Figure 5 are the same modulo renaming the point-annotated nulls. This is formally proved in Section 6. Specifically, corresponding point-annotated nulls in Figure 4 and Figure 5 carry the same time points (i.e. their temporal attribute value).

Note that even when the source instance is concrete, the s-t ttgds and tkcs are viewed in the abstract way; that is, by interpreting the abstract view of the source instance and applying the standard semantics of s-t ttgds and tkcs. However, for practical purposes, we store the target instance in its concrete representation since this representation is generally more compact than its abstract counterpart.

Contributions Our main contribution of this paper is the formalization and study of the framework of data exchange on temporal data. We define two chase algorithms, one for each view (abstract vs. concrete) of temporal data: the **abstract chase algorithm** and the **concrete chase algorithm**. We show the correctness of the concrete chase with respect to the abstract chase. This result is important because it enables one to implement data exchange on concrete temporal data with semantics corresponding to the abstract case. We also show that the result of the abstract chase over an abstract source instance is a universal solution and, therefore, the corresponding concrete solution provides a foundation for answering unions.
of conjunctive queries. Finally, we define naïve evaluation on the abstract and concrete instances obtained by the chase algorithms and show it produces certain answers.

Additional definitions and the proofs of all the theorems and lemmas in this paper are provided in the Appendix.

2 Related Work

Here we overview the previous relevant work on data exchange and temporal databases. The formal foundations of data exchange were developed by Fagin et al. in [7]. The authors showed that the chase algorithm, previously used for checking implication of data dependencies, can be used to produce a universal solution for instances of the data exchange problem. Universal solutions map homomorphically to other solutions for the source instance. This property makes them the preferred solutions to query. Universal solutions and, in general, solutions of instances of data exchange problem can contain incomplete information. Representing incomplete information and evaluating queries over them are more complex than in the complete case [1, 9]. The gap between the theoretical work on incomplete information and what has been used in practice is discussed in [8, 15]. The chase algorithm proposed in [7] produces labeled nulls for incomplete values. Relations with labeled nulls in them are called naïve tables [1, 9]. In data exchange the semantics of answering queries are defined in terms of certain answers [3, 7]. Certain answers [9] are tuples that belong to the answer of the posed query no matter which solution is used. It is shown in [7] that whenever a universal solution can be computed in polynomial time (for the class of dependencies identified in [7]), certain answers to unions of conjunctive queries can also be computed in polynomial time. Computing certain answers for queries that have inequalities, however, is a coNP-complete problem [7]. Data exchange and incomplete information and other possible semantics for query answering are discussed in detail in [14].

The original chase, which we call the standard chase throughout the paper, assumes finite relational instances and is a sequential process. Further extensions to the standard chase were surveyed in [16]. In [2] a chase step, called core chase step, was proposed that consists of applying all tgds in parallel; however, the details in the case of egds were missing.
Motivated by [2], the proposed chase algorithms in this paper aggregate chase steps in two rounds to cope with possibly infinite abstract relations. The work is different from previous work on data exchange in two aspects: the abstract view requires infinite sets of tuples and the concrete view is an extension of relational model with intervals as the temporal attribute values.

The formal foundations of temporal data models and query languages were studied in [3][5]. Abstract versus concrete temporal views were first developed in the context of the semantics of temporal query languages [19]. These two level views of temporal data later were used in program debugging and dynamic program analysis [13]. The papers [4][5] did not discuss incomplete temporal information and its possible semantics. Koubarakis proposed a unified framework for both finite and infinite, definite and indefinite temporal data [11][10]. His suggested framework extends conditional tables (a.k.a. c-tables) [4] and can be used to store facts such as roomA is booked from 2 to sometime between 5 to 8. He used global conditions to define the constraints on the start point or end point of a time interval. In his framework, the temporal attribute values can be unknown. C-tables are a generalization of naive tables where a table is associated with global and local conditions specified by logical formulas. In temporal data exchange, the proposed chase algorithms produce point-annotated and interval-annotated nulls for unknown values. Since in our framework, the value of temporal attribute is known in source instances and there is no condition on the temporal attribute or non-temporal attributes as a result of data exchange, naive tables are sufficient for representing incomplete information.

3 Preliminaries and Definitions

Time points and time intervals. Let $T_P = \mathbb{N} \cup \{\infty\}$, where $\mathbb{N}$ denotes the set of natural numbers. Let $T_1 = \{(s, e) | s, e \in T_P, s < e\}$. As described in the Introduction, a clopen interval $[s, e)$ denotes the set $\{s, s+1, \ldots, e-1\}$ of time points. A clopen interval is equivalent to two temporal attributes for representing the beginning and the end of the time interval [4]. This construction is also used in the SQL:2011 standard [12].

Domains. There are three domains Const, PNull, and INull. The domain Const is a set of infinitely many uninterpreted constants. The domain PNull is an infinite set of point-annotated nulls. A point-annotated null is a pair $N^s$ where $N$ is a label and $s$ is a time point which shows the temporal context of $N$, i.e. the time point at which $N$ occurs. Similarly, the domain INull is an infinite set of interval-annotated nulls. An interval-annotated null is a pair $N^{s,e}$ where $N$ is a label and $[s,e)$ is a clopen interval which denotes the temporal context of $N$. Throughout this paper, we use annotated nulls to refer to both point-annotated nulls and interval-annotated nulls.

Two annotated nulls are equal if they are syntactically identical. We assume that the sets PNull, INull, Const, $T_P$ and $T_1$ are countably infinite and mutually disjoint.

Abstract and concrete schema and instances, semantic mapping. A schema is a finite sequence $R = <R_1, \ldots, R_k>$ of relation symbols. Each relation symbol $R_i$ is associated with a set $\{A_{j1}, \ldots, A_{j_k}, T\}$ of attributes where the domain of $A_{ji}$ ($1 \leq j \leq k_i$) is Const and the domain of $T$ is either $T_P$ or $T_1$. An abstract instance $I_a$ over the schema $R$ is a function that associates to each relation symbol $R_i$ an abstract relation $I^{R_i}$, where $I^{R_i}$ is a subset of $(\text{Const} \cup \text{PNull})^{k_i} \times T_P$. In other words, except for the temporal attribute, annotated nulls can occur among the values of other attributes in the tuples. A complete abstract instance $I$ is a special case, where $I^{R_i}$ is a subset of Const$^{k_i} \times T_P$.

Similarly, a concrete instance $I_c$ over the schema $R$ is a function that associates to each
relation symbol $R_i$ a relation $I^{R_i}$, where $I^{R_i}$ is a subset of $(\text{Const} \cup \text{Null})^k \times T_i$. Likewise, a \textit{complete concrete instance} is a special case of a concrete instance if each $I^{R_i}$ is a subset of $\text{Const}^k \times T_i$. We will also sometimes write the tuple $(\bar{a}, t)$ in relation $R_i$ as the fact $R_i(\bar{a}, t)$.

We define the relationship between a concrete instance and an abstract instance through the \textit{semantic mapping} function \cite{5}, as follows:

1. If $(a_1, \ldots, a_k, [s, e])$ is a concrete tuple, then $\llbracket (a_1, \ldots, a_k, [s, e]) \rrbracket = \{ (a'_1, \ldots, a'_k, t_0) \mid s \leq t_0 < e \}$ where $a'_i = a_i$ if $a_i$ is a constant, and $a'_i$ is a point-annotated null $N^{0}$ if $a_i$ is an interval-annotated null $N^{(s,e)}$.

2. If $R$ is a concrete relation, then $\llbracket R \rrbracket = \bigcup_{u \in R} [\llbracket u \rrbracket]$.

3. If $I_c$ is a concrete instance that consists of relations $R_1, \ldots, R_k$, then $\llbracket I_c \rrbracket = \llbracket R_1 \rrbracket, \ldots, \llbracket R_k \rrbracket$.

The above definition generalizes the definition in \cite{5} to handle null values in both views. As defined above, under the semantic mapping, each interval-annotated null $N^{(s,e)}$ corresponds to a set of point-annotated nulls $\{N^{s}, \ldots, N^{e-1}\}$ where each $N^{0}$, $t_0 \in [s,e)$, denotes a point-annotated null.

**Temporal schema mapping, solution, conflicting facts.** A \textit{source-to-target temporal tuple generating dependency (s-t ttgd)}, denoted by $\sigma_s$, is a dependency of the following form:

$$\sigma_s = \forall \bar{x} \alpha(\bar{x}, t) \rightarrow \exists \bar{y} \beta(\bar{x}, \bar{y}, t),$$

where $\alpha$ (resp. $\beta$) is a conjunction of relational atoms (i.e. $\alpha = \alpha_1 \land \ldots \land \alpha_m$, $m \geq 1$) over the source schema (respectively, the target schema), and $\bar{x}$ and $\bar{y}$ are vectors of variables and $t$ denotes a variable that represents the temporal attribute values. We call $t$ the \textit{temporal variable} throughout the paper. An example of an s-t ttgd is given in Example 1.

A \textit{temporal key constraint} (tkc), denoted by $\sigma_k$, over a relation schema $R$ is a constraint of the form:

$$\sigma_k = R(\bar{x}_1, \ldots, \bar{x}_m, y_1, \ldots, y_n, t) \land R(\bar{x}_1, \ldots, \bar{x}_m, y'_1, \ldots, y'_n, t) \rightarrow y_1 = y'_1 \land \ldots \land y_n = y'_n,$$

where $m \geq 0$ and $n \geq 1$ and all variables are universally quantified. The set $\{A_1, \ldots, A_m, T\}$ of attributes is the \textit{temporal key} of $R$. A \textit{temporal schema mapping} is a quadruple $M = (R_S, R_T, \Sigma_{st}, \Sigma_k)$, where $R_S$ and $R_T$ are the source and target schemas, respectively; $\Sigma_{st}$ is a set of s-t ttgd and $\Sigma_k$ is a set of tkcs. As in SQL, (temporal) key attributes cannot be null.

We use rhs (resp. lhs) to refer to the right hand side (resp. left hand side) of an s-t ttgd or a tkc. We assume that there is only one temporal variable in s-t ttgds and tkcs. Multiple such variables are essential only if built-in predicates like 'greater than' (i.e. $>$) are available.

When we say an abstract instance \textit{satisfies} an s-t ttgd or a tkc (or a set of them), we refer to first order logic satisfaction and when we say a concrete instance satisfies an s-t ttgd or a tkc, it means the abstract semantics of that concrete instance satisfies the constraint.

Given an abstract source instance $I_a$ and a temporal schema mapping $M$, an \textit{abstract solution} for $I_a$ w.r.t $M$ is an abstract target instance such that $(I_a, J_a)$ satisfies the constraints in $M$. Likewise, a \textit{concrete solution} $I_c$ for a concrete source instance $I_c$ w.r.t $M$ is a target instance such that $(I_c, J_c)$ satisfies constraints in $M$.

Two abstract facts are \textit{conflicting abstract facts}, w.r.t a tkc $\sigma_k$ if they agree on temporal key attribute values, but differ in other attribute values. \textit{Conflicting concrete facts} are defined in the same way.

**Formula homomorphism, abstract homomorphism, universal solution.** In \cite{7}, one type of \textit{homomorphism} is defined between two relational instances containing labeled nulls and constants, and from a conjunctive formula to a relational instance. We distinguish two types of homomorphisms: \textit{formula homomorphism} and \textit{abstract homomorphism} because the mapping between two point-annotated nulls in abstract homomorphisms is treated differently from the mapping of a variable in a formula to an annotated null. More formally, a \textit{formula homomorphism}
To deal with infinitely many chase steps, there are two options: define appropriate notions of fix-point or convergence, or use parallel chase. We show that our parallel chase, like the sequential chase, produces a universal solution (Corollary 16). The abstract chase algorithm proceeds by an s-t ttgd a-chase step, followed by a tkc round, denoted by a-chase$^\Sigma_{st}$. In each round, the a-chase algorithm proceeds by applying possibly infinitely many abstract chase steps in parallel. If the tkc round is successful, the result of the a-chase algorithm is an abstract solution for the given source instance w.r.t $\mathcal{M}$; otherwise, the result is a failure.

4 Abstract Chase

In this section, we define an abstract chase (a-chase) algorithm for the abstract view with a temporal schema mapping $\mathcal{M}$. As mentioned before, an abstract instance is a relational instance with possible infinite relations; therefore, it cannot be chased sequentially as in [7]. To deal with infinitely many chase steps, there are two options: define appropriate notions of fix-point or convergence, or use parallel chase. We show that our parallel chase, like the standard sequential chase, produces a universal solution (Corollary 16). The abstract chase algorithm proceeds by an s-t ttgd a-chase step, denoted by a-chase$^\Sigma_{st}$, followed by a tkc round, denoted by a-chase$^\Sigma_{tkc}$. In each round, the a-chase algorithm proceeds by applying possibly infinitely many abstract chase steps in parallel. If the tkc round is successful, the result of the a-chase algorithm is an abstract solution for the given source instance w.r.t $\mathcal{M}$; otherwise, the result is a failure.

4.1 S-t ttgd Round

In the s-t ttgd a-chase round, the abstract source instance is chased with $\Sigma_{st}$. First, we define an s-t ttgd a-chase step. Then, we define an s-t ttgd a-chase round that aggregates all applicable s-t ttgd a-chase steps. In the definitions below, let $I_a$ be a complete abstract instance.

Definition 2. (S-t ttgd a-chase step): Let $\sigma_a$ be an s-t ttgd in $\Sigma_{st}$. Given a formula homomorphism $h$ from lhs of $\sigma_a$ to $I_a$, we say that an s-t ttgd a-chase step can be applied to $I_a$ with $\sigma_a$ using $h$ to get the instance $K_a$, denoted by $I_a \xrightarrow{\sigma_a,h} K_a$. The instance $K_a$ is obtained by (a) extending $h$ to $h'$ such that each existential variable in the rhs of $\sigma_a$ is
assigned a fresh annotated null, with the temporal context $h(t)$, followed by (b) applying $h'$ on the atoms in the rhs of $\sigma_s$.

\begin{definition}
\textbf{(S-t ttgd a-chase round):} The s-t ttgd a-chase round $(a$-$\text{chase}_{\Sigma_{st}}^{\ast})$ of $I_a$ is the union of results of all a-chase steps in Definition 2 that can be applied on the facts of $I_a$ with all dependencies in $\Sigma_{st}$. Let $J_a$ be the target instance obtained by the s-t ttgd round, defined as:

$$J_a = \bigcup_{\sigma_s \in \Sigma_{st}, h, I_a}^{\sigma_s \vdash a \rightarrow K_a} K_a$$

Note that $J_a$ is an abstract solution for $I_a$ w.r.t $M' = (R_S, R_T, \Sigma_{st}, \emptyset)$ because $(I_a, J_a)$ satisfies $\Sigma_{st}$. Corollary 14 (in Section 7) shows that $J_a$ is a universal abstract solution for $I_a$ w.r.t $M'$.
\end{definition}

\subsection{Tkc Round}

We present here the tkc a-chase round which follows the s-t ttgd a-chase round. The tkc round proceeds in a fashion similar to the s-t ttgd round: aggregating tkc a-chase steps on the result of the s-t ttgd round. However, the final processing is more complex. Unlike the sequential chase with equality generating dependencies (egds) in [7], where a labeled null is replaced with another labeled null or a constant (in each successful egd chase step), we show in the Example 4 that one cannot simply replace an annotated null with another value in parallel application of chase steps.

\begin{example}
Consider the Emp abstract relation in Figure 6 and the following tkc:

$$\forall n \forall p \forall p' \forall c \forall t \ Emp(n, p, c, t) \land Emp(n, p', c, t) \rightarrow p = p'.$$

Notice that the sequential application of chase steps on this instance fails because DBA $\neq$ Manager. Indeed, there is no solution for this instance with the above tkc. However, the parallel application of chase steps will not fail because one of the tkc a-chase steps results in replacing $N^{2008}$ with DBA in the first tuple and the other replaces $N^{2008}$ with Manager in the third tuple, independently of each other.

Instead of replacing point-annotated nulls with values during a tkc a-chase step, a set of equalities is generated that represents the replacements (Definition 5). Then, in the tkc a-chase round, we take the union of all the equalities obtained in the individual chase steps and reason about the equalities that can be deduced by considering their symmetric transitive closure (Definition 6). The abstract chase algorithm fails if an equality between two distinct constants is deduced. Otherwise, an abstract solution for the given source instance w.r.t $M = (R_S, R_T, \Sigma_{st}, \Sigma_k)$ is obtained.

In the following definitions let $J_a$ be the result of the s-t ttgd a-chase round and let $\sigma_k \in \Sigma_k$ be a tkc defined in Section 3.
Definition 5. (Tkc a-chase step): Let \( w_1 \) and \( w_2 \) be two conflicting abstract facts in \( J_a \) w.r.t \( \sigma_k \). If \( h \) is a formula homomorphism from lhs of \( \sigma_k \) to \( J_a \) such that: 
\[
\begin{align*}
w_1 &= R(h(x_1), \ldots, h(x_m), h(y_1), \ldots, h(y_n), h(t)), \\
w_2 &= R(h(x_1), \ldots, h(x_m), h(y'_1), \ldots, h(y'_n), h(t))
\end{align*}
\]
the tkc a-chase step can be applied on \( J_a \) with \( \sigma_k \), denoted by \( a\text{-chase}_{\sigma_k}(w_1, w_2) \), and the result is a set of equalities:
\[
a\text{-chase}_{\sigma_k}(w_1, w_2) = \{ h(y_i) = h(y'_i) | 1 \leq i \leq n \}.
\]

Definition 6. (Tkc a-chase round): Let \( w_1 \) and \( w_2 \) be two conflicting abstract facts in \( J_a \) w.r.t \( \sigma_k \). Let \( E_{J_a, \Sigma_k} \) be defined as follows:
\[
E_{J_a, \Sigma_k} = \bigcup_{\sigma \in \Sigma_k, w_1, w_2 \in J_a} a\text{-chase}_{\sigma_k}(w_1, w_2).
\]
\( E_{J_a, \Sigma_k} \models x = y \), means that the symmetric transitive closure of \( E_{J_a, \Sigma_k} \) contains the equality \( x = y \). We consider three equality cases that can be derived:
- If \( E_{J_a, \Sigma_k} \models c = c' \), where \( c \) and \( c' \) are two distinct constants, then the tkc a-chase round fails and the result of the a-chase algorithm is a failure.
- If \( E_{J_a, \Sigma_k} \models N^{t_1} = c \) for any point-annotated null \( N^{t_1} \), then all occurrences of \( N^{t_1} \) in \( J_a \) will be replaced by \( c \).
- If \( E_{J_a, \Sigma_k} \models N^{t_1} = M^{t_1} \) for point-annotated nulls \( N^{t_1} \) and \( M^{t_1} \), then one of them will be designated to replace all occurrences of the other one in \( J_a \).

Let \( J'_a \) be the instance \( J_a \) after all replacements have been applied. Then \( a\text{-chase}^*_{\Sigma_k}(J_a) = J'_a \).

Note that the order of applying the above replacements does not matter as long as all replacements induced by symmetric transitive closure of \( E_{J_a, \Sigma_k} \) are considered.

If the abstract chase is successful, an abstract solution is obtained for the given source instance w.r.t the given temporal schema mapping. Corollary 16 (in Section 7) shows that this solution is a universal abstract solution w.r.t \( \mathcal{M} \).

5 Concrete Chase

Like the abstract chase algorithm, the concrete chase proceeds by first applying s-t ttgd c-chase steps in an s-t ttgd round, denoted by \( c\text{-chase}_{\Sigma_{st}}^* \), and then tkc c-chase steps in a tkc c-chase round, denoted by \( c\text{-chase}_{\Sigma_{tkc}}^* \). The definitions of c-chase steps and c-chase rounds are in the Appendix. Here, we discuss the important differences between the abstract and concrete chase algorithms.

The main difference in the definition of an s-t ttgd c-chase step from the abstract one is that the temporal attribute values are clopen intervals and interval-annotated nulls are generated for existential variables in the rhs of s-t ttgds. For example, Figure 7 shows the result of \( c\text{-chase}_{\Sigma_{st}}^* \) round on the concrete source instance given in the Example 1. The other important difference is that the concrete source instance is normalized. Normalization is necessary to ensure that all the occurrences of the temporal variable in an s-t ttgd or a tkc map to the same clopen interval in a concrete instance. A set of concrete tuples (that might have different schemas) is normalized if the clopen intervals of the tuples are either equal or disjoint (i.e. no overlap in the intervals of different tuples). Note that given a normalized concrete instance \( I_c \), each time point in \([I_c]\) belongs to a unique interval in \( I_c \).

Without normalization, formula homomorphisms cannot be defined from an s-t ttgd or a tkc to a concrete instance. For example, consider the concrete source instance shown in Figure 1. Assume the following s-t ttgd:
Employee1

<table>
<thead>
<tr>
<th>Name</th>
<th>Company</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ada</td>
<td>IBM</td>
<td>(08, 10)</td>
</tr>
<tr>
<td>Ada</td>
<td>IBM</td>
<td>(10, 11)</td>
</tr>
<tr>
<td>Ada</td>
<td>Intel</td>
<td>(11, 13)</td>
</tr>
</tbody>
</table>

Employee2

<table>
<thead>
<tr>
<th>Name</th>
<th>Position</th>
<th>Dept</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ada</td>
<td>Developer</td>
<td>Computer</td>
<td>(08, 10)</td>
</tr>
<tr>
<td>Ada</td>
<td>DBA</td>
<td>Computer</td>
<td>(10, 11)</td>
</tr>
</tbody>
</table>

σ_s : ∀n∀c∀p∀d∀t Employee_1(n, c, t) ∧ Employee_2(n, p, d, t) → Employee(n, p, c, t).

Note that the intersection of time intervals in concrete relations Employee_1 and Employee_2 is not empty. No formula homomorphism h can be defined from lhs of σ_s to Employee_1 and Employee_2 because the variable t cannot be mapped to a single time interval. However, three formula homomorphisms can be defined from lhs of σ_s to the corresponding abstract instance shown in Figure 2, such as h' : {n → Ada, c → IBM, p → Developer, d → Computer, t → 08}.

The result of normalization of the concrete instance in Figure 1 is shown in Figure 8. Now we can define two formula homomorphisms from σ_s to the normalized instance such as:

h : {n → Ada, c → IBM, p → Developer, d → Computer, t → [08, 10]}.  

6 Implementation of the abstract chase by the concrete chase

In this section, we show that the concrete chase is correct. This means that the result of the concrete chase is homomorphically equivalent to the result of the abstract chase under the semantic mapping. In the theorems of this section, let Σ_st and Σ_a be a set of s-t ttgds and a set of tks, respectively. Also, let ∼ denote the homomorphic equivalence between two abstract instances.

Theorem 7. Let I_c be a normalized complete instance. Then the result of the s-t ttgd c-chase round on I_c w.r.t Σ_st is homomorphically equivalent to the result of s-t ttgd a-chase round on I_a. That is: [c-chase^*_{Σ_st}(I_c)] ∼ a-chase^*_{Σ_a}(I_c).
**Theorem 8.** Let $J_c$ be the result of s-t ttgd c-chase round. Then, $a$-chase$_{\Sigma_k} ([J_c] \sim [c$-chase$_{\Sigma_k} (J_c)])$.

The proof of the theorem makes use of the following lemmas. Lemma 9 shows the correspondence between conflicting concrete and abstract facts. Lemma 10 relates a tkc c-chase step to a set of tkc a-chase steps and finally Lemma 12 shows there is a correspondence between the set of equalities found by the tkc c-chase round and the tkc a-chase round. In these lemmas let $J_c$ be the result of the s-t ttgd c-chase round which is normalized and let $J_a = [J_c]$. Let $\sigma_k \in \Sigma_k$ be a tkc over the schema of $J_c$.

**Lemma 9.** Let $u_1$ and $u_2$ be two distinct concrete facts in $J_c$. The facts $u_1$ and $u_2$ are conflicting w.r.t $\sigma_k$ if and only if all abstract facts $w_1 \in [u_1]$ and $w_2 \in [u_2]$ that have the same time point (i.e. $w_1[T] = w_2[T]$) are conflicting w.r.t $\sigma_k$ in $J_a$.

**Lemma 10.** Let $u_1$ and $u_2$ be two conflicting concrete facts in $J_c$ w.r.t $\sigma_k$. The tkc c-chase step c-chase$_{\sigma_k}(u_1, u_2)$ can be applied to $J_c$ with $\sigma_k$ if and only if for all conflicting abstract facts $w_1 \in [u_1]$ and $w_2 \in [u_2]$ w.r.t $\sigma_k$, the tkc a-chase step a-chase$_{\sigma_k}(w_1, w_2)$ can be applied to $J_a$.

For the next lemma, we first need to define the semantic mapping of the set of equalities generated by tkc c-chase steps.

**Definition 11.** (Semantic mapping of $E_{J_c, \Sigma_k}$): Let $E_{J_c, \Sigma_k}$ be the union of all the equalities found by all c-chase steps on the concrete instance $J_c$ with a set of temporal key constraints $\Sigma_k$. We define the semantic mapping of $E_{J_c, \Sigma_k}$, denoted by $[E_{J_c, \Sigma_k}]$, as follows:
The universality of a tkc a-chase round is shown in Lemma 15. Corollary 16 shows the result. This corollary follows from Lemma 13 which shows there is an abstract homomorphism from \( M \), where \( c \in \text{Const} \), then \( \{N^s = c, ..., N^{e-1} = c\} \) is in \([E_{L}, \Sigma_a]\].

No other equalities are in \([E_{L}, \Sigma_a]\).

\begin{itemize}
  \item If \( N^{(s,e)} = M^{(s,e)} \) is in \( E_{L}, \Sigma_a \), then \( \{N^s = M^s, ..., N^{e-1} = M^{e-1}\} \) is in \([E_{L}, \Sigma_a]\).
  \item If \( N^{(s,e)} = c \) is in \( E_{L}, \Sigma_a \), where \( c \in \text{Const} \), then \( \{N^s = c, ..., N^{e-1} = c\} \) is in \([E_{L}, \Sigma_a]\).
\end{itemize}

\[\textbf{Lemma 12.} \text{Let } E_{L}, \Sigma_a \text{ be the union of all the equalities found by all tkc c-chase steps on the normalized concrete instance } J_c \text{ with a set of tkc } \Sigma_k \text{. Let } E_{L}, \Sigma_k \text{ be union of all the equalities found by all tkc a-chase steps on } J_a = [J_c] \text{ with } \Sigma_k \text{. Then } E_{L}, \Sigma_a \equiv E_{L}, \Sigma_k, \text{ where } \equiv \text{ is the logical equivalence up to renaming point-annotated nulls.}\]

\section{Universal Solutions}

In this section, we show that the result of a successful abstract chase algorithm on \( I_a \) w.r.t \( M \) is a universal abstract solution. As investigated in \cite{[7]}, universal solutions are the preferred solutions to materialize because they can be used to find certain answers to unions of conjunctive queries in polynomial time. To show the universality of an abstract solution \( J_a \) for \( I_a \) w.r.t \( M \), we need to show there is an abstract homomorphism from \( J_a \) to every other abstract solution \( J'_a \) for \( I_a \) w.r.t \( M \). In spite of universality of the solution obtained by an abstract chase on \( I_a \) w.r.t \( M \), it is not always possible to materialize it. Instead, because of Theorems \cite{[7]} and \cite{[8]} the concrete solution generated by the concrete chase algorithm on \( I_c \) w.r.t \( M \), such that \([I_c] = I_a\) is materialized. This solution is called a universal concrete solution.

In this section, let \( M' = (R_S, R_T, \Sigma_{st}, \emptyset) \) and \( M = (R_S, R_T, \Sigma_{st}, \Sigma_k) \) be two temporal schema mappings. Recall that the abstract chase algorithm has two rounds. The universality of the solution obtained by the s-t ttgd a-chase round w.r.t \( M' \) is shown in Corollary 14. This corollary follows from Lemma 12 which shows there is an abstract homomorphism from the result of an s-t ttgd a-chase step on the complete abstract instance \( I_a \) to any abstract solution for \( I_a \) w.r.t \( M' \).

\[\textbf{Lemma 13.} \text{Let } (I_a, \emptyset) \stackrel{\sigma_s \cdot h}{\longrightarrow} (I_a, J_i) \text{ be an s-t ttgd a-chase step on } I_a \text{ with } \sigma_s. \text{ Let } J'_a \text{ be an instance such that } (I_a, J'_a) \text{ satisfies } \sigma_s. \text{ Then there is an abstract homomorphism from } J_i \text{ to } J'_a.\]

\[\textbf{Corollary 14.} \text{Let } J_a \text{ be the result of a-chase}_a \Sigma_{st} \text{ on } I_a \text{ w.r.t } M'. \text{ Let } J'_a \text{ be an instance such that } (I_a, J'_a) \text{ satisfies } \Sigma_{st}. \text{ That is } J'_a \text{ is an abstract solution for } I_a \text{ w.r.t } M'. \text{ Then there is an abstract homomorphism from } J_a \text{ to } J'_a. \text{ Therefore, } J_a \text{ is a universal abstract solution w.r.t } M'.\]

The universality of a tkc a-chase round is shown in Lemma 15. Corollary 16 shows the result of a-chase algorithm is a universal abstract solution.

\[\textbf{Lemma 15.} \text{Let } J_a \text{ be an abstract solution for } I_a \text{ w.r.t } M'. \text{ Let } J'_a \text{ be the result of a successful a-chase}_a \Sigma_{st} \text{ round on } J_a \text{ which is an abstract solution for } I_a \text{ w.r.t } M. \text{ Let } J''_a \text{ be any arbitrary abstract solution for } I_a \text{ w.r.t } M \text{ and such that there exists an abstract homomorphism } h_1 : J_a \mapsto J''_a. \text{ Then } h_1 : J'_a \mapsto J''_a.\]

\[\textbf{Corollary 16.} \text{Let } I_a \text{ be a source instance. Let } J_a \text{ be the result of s-t ttgd a-chase round on } (I_a, \emptyset) \text{ with } \Sigma_{st}. \text{ Let } J'_a \text{ be the result of the tkc a-chase round on } J_a \text{ with } \Sigma_k. \text{ Then, } J'_a \text{ is a universal abstract solution of } I_a \text{ w.r.t } M.\]
### 8 Query Answering

In addition to finding a universal solution for a data exchange problem, another important issue is query answering over the target schema [7]. When queries are posed over the target schema, different answers may be obtained depending on the solution that is considered. A widely used notion is the concept of certain answers, where the answers are the intersection of all the answers to the query on each possible solution. In this section, we first define concrete certain answers and then a method to calculate them.

▶ **Definition 17.** Concrete certain answers: Given a normalized concrete source instance $I_c$, the concrete certain answers of $Q$ w.r.t $I_c$ and $M$, denoted by $\text{certain}_c(Q, I_c, M)$ are:

$$
\text{certain}_c(Q, I_c, M) = \bigcap \{Q(J_c)| J_c \text{ is a concrete solution for } I_c\}
$$

where $Q(J_c)$ is the result of evaluating query $Q$ on $J_c$ treating interval-annotated nulls as distinct constants.

The definition of $\text{certain}_a(Q, I_a, M)$ is similar. Observe that certain answers to queries are complete tuples. Therefore, annotated nulls are used only during data exchange and do not appear in certain answers of queries.

**Conjunctive queries and unions of conjunctive queries** play an important role in relational data exchange because certain answers to this class of queries can be obtained in polynomial time [7]. Conjunctive queries are the $\exists\land$ fragment of relational calculus. In this paper, a conjunctive query $Q$ has the form $\exists\bar{y}\beta(\bar{x}, \bar{y}, t)$. Unions of conjunctive queries are a class of queries of the form $Q_1 \lor \ldots \lor Q_m$, where each $Q_i$ ($1 \leq i \leq m$) is a conjunctive query. Unions of conjunctive queries are preserved under abstract homomorphism [3] meaning that if there is an abstract homomorphism $h: I_a \mapsto I'_a$ and $Q$ is a union of conjunctive queries, then if $w$ is a complete abstract tuple in $Q(I_a)$, it is also in $Q(I'_a)$.

The following theorem claims that the certain answers of a conjunctive query $Q$ on the concrete and abstract view are identical under the semantic mapping. The intuition behind the proof is that certain answers contain only constants and not the annotated nulls; also, the proof uses the properties of unions of conjunctive queries.

▶ **Theorem 18.** Let $M$ be a temporal schema mapping and $Q$ be a union of conjunctive queries over the target schema. Let $I_c$ be a concrete normalized source instance. Then we have:

$$
[\text{certain}_c(Q, I_c, M)] = \text{certain}_a(Q, [I_c], M).
$$

Naïve evaluation [1, 3, 9, 7] is a technique commonly used in the literature to find certain answers for unions of conjunctive queries on naïve tables. It has been shown [3, 7] that naïve evaluation of unions of conjunctive queries on a universal solution $J$ for a relational source instance $I$ gives certain answers. The same proof applies to universal abstract solutions, because the proof needs the definitions of certain answers and universal solutions which are the same for abstract and relational instances, modulo modifications for handling point-annotated nulls. However, for the concrete case, we need to show that naïve evaluation on a universal concrete solution will produce certain answers (Corollary 20).

Given a union of conjunctive queries $Q$ and a concrete solution $J_c$ for a source instance $I_c$ and a temporal schema mapping, the naïve evaluation of $Q$ on $J_c$, denoted by $Q(J_c)_1$, works as follows:

- Each interval-annotated null $N^{[s,e)}$ in $J_c$ is replaced with a fresh constant $cn^{[s,e)}$ everywhere it occurs.
Query $Q$ is evaluated by finding all formula homomorphisms from variables in $Q$ to $J_c$. In particular, the variable $t$ is mapped to a clopen interval. The result of this step is denoted by $Q(J_c)$.

Tuples with fresh constants are dropped from $Q(J_c)$.

Naïve evaluation of $Q$ on an abstract solution $J_a$, denoted by $Q(J_a)$, works in the same way, except that each point-annotated null $N^t_0$ is replaced with a fresh constant $cn^t_0$. The following theorem shows that naïve evaluation on a concrete solution produces the same answers as naïve evaluation on the corresponding abstract solution under the semantic mapping.

\[ Q(J_c) = Q(J_a) \]

\[ Q(J_c) \downarrow = Q(J_a) \downarrow \]

\[ Q(J_c) \downarrow = Q(J_a) \downarrow \]

\[ Q(J_c) \downarrow = Q(J_a) \downarrow \]

\[ Q(J_c) \downarrow = Q(J_a) \downarrow \]

9 Conclusion and Future Work

In this paper, we proposed a framework for data exchange on temporal data which emphasizes on the abstract vs. concrete view of the data. We considered a basic case where source instances have a single temporal dimension, and similarly to the original data exchange framework [7], the source instances are assumed to be complete. The temporal schema mappings consist of a set of s-t tgds and tks. We proposed two chase algorithms: one for concrete temporal instances and one for abstract temporal instances. The abstract chase makes it possible to define the semantics of the concrete chase and to prove that the concrete chase is correct. We defined the abstract chase as the parallel application of chase steps to avoid reasoning over infinite sequences of chase steps. We also discussed that parallel application of chase steps for temporal key constraints requires more care as not to equate one unknown value with two different constants in two chase steps. Then we defined the concrete chase and proved that its result is semantically aligned to the one achieved by the abstract chase. We showed the result of the abstract chase is a universal abstract solution and since one can implement the abstract chase by the concrete chase, the result of the concrete chase is a good candidate to be materialized and used for answering queries.

For future work, we plan to investigate our approach on real use cases. This needs an efficient implementation of normalization operation because normalization might lead to a blow-up in the number of concrete tuples if a large number of clopen intervals in the tuples within the source instance overlap each other. A normalized instance is much more fragmented than an un-normalized one, defeating the whole purpose of using intervals as a more compact representation. Furthermore, it would be nice to identify other tractable temporal constraints on the target schema. For example, egds in general or constraints with a comparison operator on time such as ‘If a Ph.D student graduated at time $t$, then there must be a time $t'$ that they passed the candidacy examination such that $t' < t$’. The challenge in the case of general egds is that the order of chase steps might matter for such dependencies because the application of one chase step might lead to the application of another one. A good starting point for considering comparison operators is the paper [18] which explores arithmetic operations in s-t tgds and tgds.
Acknowledgments

Jan Chomicki is partially supported by NSF grants IIS-1450590 and IIS-1524469. Wang-Chiew Tan is partially supported by NSF grants IIS-1450560 and IIS-1524382.

References

10 Appendix

Definitions of s-t ttgd c-chase step and c-chase round

Definition 21. (s-t ttgd c-chase step): Let $I_c$ be a complete concrete normalized instance and let $\sigma_s$ be an s-t ttgd in $\Sigma_{st}$. Let $h$ be a formula homomorphism from lhs of $\sigma_s$ to $I_c$. The s-t ttgd $\sigma_s$ can be applied to $I_c$ with homomorphism $h$, denoted by $I_c \xrightarrow{\sigma_s, h} K_c$, and the result (i.e., $J_c$), is the set of facts obtained by

- extending $h$ to $h'$ such that each existential variable is assigned a fresh null annotated with the temporal context $h(t)$, followed by
- applying $h'$ to the rhs of $\sigma_s$

We denote the result of applying $\sigma_s$ to $I_c$ by $I_c \xrightarrow{\sigma_s} K_c$.

Note that $h(t)$ is a time interval in the above definition, while it is a time point in definition 2.

Definition 22. (s-t ttgd c-chase round): Let $I_c$ be a normalized concrete instance and let $\Sigma_{st}$ be a set of s-t ttgds. Then the s-t ttgd c-chase round (denoted by $c\text{-chase}_{\Sigma_{st}}$) of $I_c$ is the union of results of all s-t ttgd c-chase steps that can be applied on the facts of $I_c$ with $\Sigma_{st}$. Let $J_c$ be the resulting target instance after the s-t ttgd round. In short:

$$J_c = \bigcup_{\sigma \in \Sigma_{st}, h, I_c \xrightarrow{\sigma, h} K_c} K_c$$

Definitions of tkc c-chase step and c-chase round

Definition 23. (TkC c-chase step): Let $J_c$ be a normalized concrete instance. Let $\sigma_k \in \Sigma_k$ be a tkc over relation schema $R(A_1, ..., A_m, B_1, ..., B_n, T)$ in $J_c$ where $m \geq 0$ and $n \geq 0$ (as defined in Section 3). Let $u_1$ and $u_2$ be two conflicting concrete facts w.r.t $\sigma_k$ in $J_c$ such that:

$$u_1 = R(h(x_1), ..., h(x_m), h(y_1), ..., h(y_n), h(t)) \text{ and } u_2 = R(h(x_1), ..., h(x_m), h(y'_1), ..., h(y'_n), h(t))$$

then $\sigma_k$ can be applied on $u_1$ and $u_2$, denoted by $c\text{-chase}_{\sigma_k}(u_1, u_2)$ and the result is a set of equalities such that:

$$c\text{-chase}_{\sigma_k}(u_1, u_2) = \{h(y_i) = h(y'_i) | 1 \leq i \leq n\}$$

Note that since $J_c$ is normalized, $t$ maps to a unique clopen interval, which is the interval of $u_1$ and $u_2$.

Definition 24. (TkC c-chase round): Let $J_c$ be a normalized concrete instance and let $\Sigma_k$ be a set of tkcs. Let $u_1$ and $u_2$ be two conflicting concrete facts. Let $E_{J_c, \Sigma_k}$ be defined as follows:

$$E_{J_c, \Sigma_k} = \bigcup_{\sigma_k \in \Sigma_k, u_1, u_2 \in J_c} c\text{-chase}_{\sigma_k}(u_1, u_2)$$

Here we have the same cases as Definition 4 except that we deal with interval-annotated nulls. Therefore, if $E_{J_c, \Sigma_k} \models \rho = \rho'$, the result of the tkc c-chase round is a failure (case 1). Otherwise, we have a replacement of an interval-annotated null with a constant or another interval-annotated null based on the equalities implied by $E_{J_c, \Sigma_k}$. Let $J'_c$ be the instance $J_c$ after all replacements have been applied, then $c\text{-chase}_{\Sigma_k}(J_c) = J'_c$. 
Note that we might have infinitely many equalities in $E_{J_c \Sigma}$, but only finitely many equalities in $E_{J_a \Sigma}$.

**Proof of Theorem 7**

**Proof.** Let $J_c = c\text{-chase}_{\Sigma_m}^c(I_c)$ and $I_a = \llbracket I_a \rrbracket$. Let $J_a = a\text{-chase}_{\Sigma_m}^a(I_a)$. First, we show that $[J_a] \Rightarrow J_a$. Let $w = \beta_j(v_1', ..., v_k', [s, e])$ be an abstract fact in $[J_a]$, where $v_1, ..., v_k$ are either constants or point-annotated nulls. So there must be a concrete fact $u = \beta_j(v_1, ..., v_k, [s, e])$ in $J_c$ such that $w \in [u]$ and $t_0 \in [s, e]$. Based on the definition of the semantic mapping, $v'_j = v_j$ $(1 \leq j \leq k)$ if $v_j \in w$ is a constant, and if $v_j$ is a point-annotated null $N^{t_0}$, then $v'_j$ is an interval-annotated null with the interval $N^{[s, e]}$. Note that since $J_c$ is normalized, the interval $[s, e]$ is a unique interval in $J_c$ that the time point $t_0$ belongs to. The concrete fact $u$ is in the result of the s-t ttgd c-chase round on $I_c$. Let

$$\sigma_s : \forall \bar{x} \alpha(\bar{x}, t) \rightarrow \exists \bar{y} \beta(\bar{x}, \bar{y}, t),$$

where $\alpha = \alpha_1 \land ... \land \alpha_m$ and $\beta = \beta_1 \land ... \land \beta_n$, be the s-t ttgd in $\Sigma_{st}$ that is used in a c-chase step on $I_c$ with a formula homomorphism $h_1$. Then there must be $m$ concrete facts $u_1, ..., u_m$ in $I_c$ such that $u_i = \alpha_i(h_1(x), h_1(t))$, $1 \leq i \leq m$. Obviously, $h_1(t) = [s, e]$, which is the clopen interval of the facts $u_1, ..., u_m$. By extending $h_1$ to $h_1'$ (as explained in Definition 21), we have $u = \beta_j(h_1'(\bar{x}), h_1'(\bar{y}), h_1'(t))$, where $h_1(\bar{x}) = h_1'(\bar{x}), h_1(t) = h_1'(t)$ and for each $y \in \bar{y}$, a fresh interval-annotated null with the interval $[s, e]$ is generated.

Let $q = \bigcup_{1 \leq i \leq m} [u_i]$. Clearly $q \subseteq I_a$. We define a formula homomorphism $h_2$, such that $h_2(\bar{x}) = h_1(\bar{x})$, and $h_2(t) = t_0$. Now we have to show $h_2$ is a formula homomorphism from the lhs of $\sigma_s$ to $q$. By applying $h_2$ to $\alpha_i$, $1 \leq i \leq m$, we have $w_i = \alpha_i(h_2(\bar{x}), h_2(t))$, $w_i \in \llbracket w_i \rrbracket$, by construction of $h_2$; therefore, $h_2$ is a formula homomorphism from the lhs of $\sigma_s$ to $q$. By extending $h_2$ to $h_2'$ (as explained in Definition 2), for each existential variable in $\bar{y}$ a fresh null annotated with the time point $h_2(t) = t_0$ is generated. By applying $h_2'$ to the rhs of $\sigma_s$, we have:

$$w' = \beta_j(h_2'(\bar{x}), h_2'(\bar{y}), h_2'(t)) = \beta_j(v''_1, ..., v''_k, t_0),$$

where $v''_j$ $(1 \leq j \leq k)$ is either a constant or a point-annotated null with the time point $t_0$. Observe that, if $v_j$ in $w$ is a constant, $v''_j = v_j$ because $h_1(\bar{x}) = h_1'(\bar{x}), h_2(\bar{x}) = h_2'(\bar{x})$ and if $v_j$ in $w$ is an annotated null such as $N^{t_0}$, then by construction explained above $v''_j$ is a point-annotated null (with the time point $t_0$). Hence, there is an abstract homomorphism from $w$ to $w'$. We have proved that there is an abstract homomorphism from each fact in $[J_a]$ to a fact in $J_a$. We define an abstract homomorphism from $[J_a]$ to $J_a$ as the union of all the abstract homomorphisms at facts level described above. There might be different occurrences of the same point-annotated null in $[J_a]$ because an s-t ttgd c-chase step generates the same interval-annotated null for each occurrence of an existentially quantified variable in the rhs of a $\sigma_s$ and the semantic mapping produces the same set of point-annotated nulls for the same interval-annotated null. Therefore, we need to make sure that each occurrence of a point-annotated null in $[J_a]$ maps to the same point-annotated null in $J_a$. In other words, the union of abstract homomorphisms at facts level is well defined. Since an s-t ttgd a-chase step, by definition, generates the same point-annotated null for each occurrence of an existentially quantified variable in the $\sigma_s$ it is guaranteed all occurrences maps to the same time point. Also, note that abstract homomorphisms are identity on constants and time points, so there is no clash regarding constants and time points.

For the other direction, we need to show $J_a \Rightarrow \llbracket c\text{-chase}_{\Sigma_m}^c(I_c) \rrbracket$. Let $w = \beta_j(a_1, ..., a_k, t_0) \in a\text{-chase}_{\Sigma_m}^a(I_a)$, where $a_1, ..., a_k$ are either nulls annotated with $t_0$ or constants. The fact $w$ is the result of a-chase on $I_a$. Let $\sigma(\forall \bar{x} \alpha(\bar{x}, t) \rightarrow \exists \bar{y} \beta(\bar{x}, \bar{y}, t))$ be the s-t ttgd that was
used in an a-chase step to produce \( w \), where \( \alpha_1 \land ... \land \alpha_m = \alpha \) and \( \beta_1 \land ... \land \beta_m = \beta \). This means that there is a formula homomorphism \( h_1 \) from lhs of \( \alpha \) to \( \alpha \) complete abstract facts \( w_1, ..., w_m \in I_a \) such that \( w_i = \alpha_i(h_1(x), h_1(t)) \), \( 1 \leq i \leq m \). Note that the facts \( w_1, ..., w_m \) have the same time point \( t_0 \), that is \( h_1(t) = t_0 \). As the result of the a-chase step with \( \sigma_s \) and \( h_1 \), we have \( w = \beta_j(h_1'(x), h_1'(y), h_1'(t)) \), where \( h_1' \) is the extension of \( h_1 \) as explained in definition 2 and \( \beta_j \) is an atom in \( \beta \). The facts \( w_1, ..., w_m \) are the result of the semantic mapping on \( I_c \). So we have \( m \) concrete tuples in \( I_c \) such that \( w_i \in \llbracket u_i \rrbracket \), \( 1 \leq i \leq m \). Since the instance \( I_c \) is normalized and \( t_0 \) belongs to all the clopen intervals in the tuples \( u_1, ..., u_m \), we have \( u_i[T] = [s,e] \), \( 1 \leq i \leq m \), where \( [s,e] \) is a unique interval in \( I_c \) such that \( t_0 \in [s,e] \).

We build a formula homomorphism \( h_2 \) such that \( h_2(x) = h_1(x) \) and \( h_2(t) = [s,e] \). Now we have to show, \( h_2 \) is a formula homomorphism from the lhs of \( \alpha \) to the tuples \( u_1, ..., u_m \). By applying \( h_2 \) on \( \alpha \), we have \( \alpha_i(h_2(x), h_2(t)) \). Observe that \( u_i = \llbracket h_2(x), h_2(t) \rrbracket \).

As the result of a c-chase step with \( \sigma_s \) and \( h_2 \), a fact \( u = \beta_j(h_2(x), h_2(y), h_2(t)) \) will be produced where \( h_2 \) is an extension of \( h_2 \) as described in Definition 21. The fact \( u \) has the form \( \beta_j(a_1', ..., a_k', [s,e]) \) where \( a_1', ..., a_k' \) are either constants or annotated nulls with interval \( [s,e] \).

Based on the definition of the semantic mapping a fact \( w' = \beta_j(a_1'', ..., a_k'', t_0) \) is in \( \llbracket a \rrbracket \) where \( a_j'' = a_j' \), \( 1 \leq j \leq k \) if \( a_j' \) is a constant. Or, \( a_j'' \) is a point-annotated null such as \( N^{t_0} \) whenever \( a_j' \) is an interval-annotated null \( N^{[s,e]} \). Observe that \( a_j'' = a_j' \) if \( a_j' \) is a constant because \( h_2(x) = h_2'(x) = h_1(x) = h_1'(x) \). Furthermore, if \( a_j \in w \) is a point-annotated null, then \( a_j'' \) is also a point-annotated null with the time point \( t_0 \) by construction of \( h_2' \). Hence, there is an abstract homomorphism from \( w \) to \( w' \). We have proved that there is an abstract homomorphism from each fact in \( J_a \) to a fact in \( \llbracket J_c \rrbracket \). We define an abstract homomorphism from \( J_a \) to \( \llbracket J_c \rrbracket \) to be the union of all abstract homomorphisms at facts level described above. There might be different occurrences of the same point-annotated null in \( J_a \) because an s-t ttgd a-chase step produces the same point-annotated null for each occurrence of an existentially quantified variable in the rhs of a \( \sigma_s \). We need to make sure each occurrence of a point-annotated null in \( J_a \) maps to the same point-annotated null in \( \llbracket J_c \rrbracket \) in the union of abstract homomorphisms at facts level. This is guaranteed by construction because an s-t ttgd c-chase step produces the same interval-annotated null in \( J_c \) for each occurrence of an existentially quantified variable in the rhs of the \( \sigma_s \) and the semantic mapping generates the same set of point-annotated nulls for each occurrence of an interval-annotated null.

**Proof of Lemma 9**

**Proof.** Assume \( u_1 \) and \( u_2 \) are conflicting concrete facts w.r.t \( \sigma_b \). Thus, they agree on temporal key attributes (i.e. \( u_1[A_1, ..., A_m, T] = u_2[A_1, ..., A_m, T] \)) but differ in some other attribute values (i.e. \( \exists i, 1 \leq i \leq n \text{ s.t. } u_1[B_i] \neq u_2[B_i] \)). Let \( [s,e] \) be the time interval of \( u_1 \) and \( u_2 \) in \( I_c \). Let \( w_1 \) and \( w_2 \) be two arbitrary facts in \( \llbracket u_1 \rrbracket \) and \( \llbracket u_2 \rrbracket \), respectively, such that \( w_1[T] = w_2[T] = t_0 \). Clearly, \( t_0 \in [s,e] \) because of the definition of the semantic mapping. Note that since \( J_c \) is normalized, clopen interval \( [s,e] \) is a unique interval \( t_0 \) belongs to. Abstract facts \( w_1 \) and \( w_2 \) also agree on temporal key attributes. Considering that the temporal key attributes cannot be null, \( u_1[A_j] = u_2[A_j] \) (\( 1 \leq j \leq m \)) is a constant such as \( c \). Therefore, \( w_1[A_j] = w_2[A_j] = c \) based on the semantic mapping definition. Now we show \( w_1 \) and \( w_2 \) are conflicting. We distinguish all concrete conflicting cases between \( u_1 \) and \( u_2 \):

- \( u_1[B_c] = c_1 \) and \( u_2[B_c] = c_2 \), where \( c_1, c_2 \in \text{Const} \) and \( c_1 \neq c_2 \), then we have \( w_1[B_c] = c_1 \) and \( w_2[B_c] = c_2 \).
- \( u_1[B_c] = N^{[s,e]} \) and \( u_2[B_c] = c \) (or vice versa), then we have \( w_1[B_c] = N^{t_0} \) and \( w_2[B_c] = c \).
- \( u_1[B_c] = N^{[s,e]} \) and \( u_2[B_c] = M^{(s,e)} \), then we have \( w_1[B_c] = N^{t_0} \) and \( w_2[B_c] = M^{t_0} \).
All of these cases are valid based on the definition of the semantic mapping and show that \( w_1 \) and \( w_2 \) are conflicting abstract facts. For the other direction, each pair of facts \( w_1 \in [u_1] \) and \( w_2 \in [u_2] \), with the same temporal attribute \( t_0 \in [s, e] \), agree on temporal key attributes \( w_1[A_1, ..., A_m, T] = w_2[A_1, ..., A_m, T] \) but not on some other attribute values (i.e. \( \exists i \text{ s.t. } w_1[B_i] \neq w_2[B_i] \)). The cases where at least one of the \( w_1[B_i] \) or \( w_2[B_i] \) is a constant (and the other one is a point-annotated null) are straightforward. So we just discuss the case where there are different point-annotated nulls, that is \( \exists w \in [u_1] \) and \( w' \in [u_2] \) such that:

\[
\begin{align*}
&u_1 = R(h(x_1), ..., h(x_m), h(y_1), ..., h(y_n), h(t)) \\
&w_2 = R(h(x_1), ..., h(x_m), h(y_1), ..., h(y_n'), h(t))
\end{align*}
\]

Based on Lemma \([9]\) all abstract facts \( w_1 \in [u_1] \) and \( w_2 \in [u_2] \) that agree on the temporal attribute are conflicting abstract facts. Let \( t_0 \) be an arbitrary time point in \([s, e]\), we build formula homomorphism \( h_{t_0} \) as follows:

\[
h_{t_0}(z) = \begin{cases} 
  h(z) & \text{if } h(z) \text{ is a constant and } z \text{ is a non-temporal variable} \\
  N^{t_0} & \text{if } h(z) \text{ is an interval-annotated null } N^{[s, e]} \\
  t_0 & \text{if } h(z) = [s, e]
\end{cases}
\]

Now we need to show \( h_{t_0} \) is a formula homomorphism from \( \sigma_k \) to \( w_1 \in [u_1] \) and \( w_2 \in [u_2] \), such that \( w_1[T] = w_2[T] = t_0 \). Wlog, we just consider \( w_1 \). Applying \( h \) to \( \sigma_k \), the following fact is obtained

\[
R(h(x_1), ..., h(x_m), h(y_1), ..., h(y_n), h(t)) = R(c_1, ..., c_m, v_1, ..., v_n, [s, e])
\]

where \( c_1, ..., c_m \) are constants, \( v_j (1 \leq j \leq n) \) is either constant or an interval-annotated null. If we apply \( h_{t_0} \) to \( \sigma_k \), the result, based on definition of \( h_{t_0} \), is

\[
R(h_{t_0}(x_1), ..., h_{t_0}(x_m), h_{t_0}(y_1), ..., h_{t_0}(y_n), h_{t_0}(t)) = R(c_1, ..., c_m, v'_1, ..., v'_n, t_0)
\]

where \( v'_j, 1 \leq j \leq n \) is either a constant or a point-annotated null. Observe that \( v'_j = v_j \) if \( v_j \) is constant and \( v'_j = N^{t_0} \) if \( v_j = N^{[s, e]} \). Therefore, \( w_1 = R(c_1, ..., c_m, v'_1, ..., v'_n, t_0) \) and \( \sigma_k \) can be applied on \( w_1 \) and \( w_2 \) (i.e. a-chase \( \sigma_k(w_1, w_2) \)).

In the other direction (i.e. only if), for each pair of conflicting abstract facts \( w_1 \in [u_1] \) and \( w_2 \in [u_2] \), with the same time point \( t_0 \in [s, e] \), there is an a-chase step (i.e. a-chase \( \sigma_k(w_1, w_2) \)). Let \( h_{t_0} \) be the homomorphism used in a-chase \( \sigma_k(w_1, w_2) \) step. Note that each time point can belong to only one interval in a normalized instance. Based on Lemma \([9]\) we have two conflicting concrete facts \( u_1 \) and \( u_2 \) with the interval \([s, e]\). We define a formula homomorphism \( h \) as follows

\[
h(z) = \begin{cases} 
  h_{t_0}(z) & \text{if } h_{t_0}(z) \text{ is a constant and } z \text{ is a non-temporal variable} \\
  N^{[s, e]} & \text{if } h_{t_0}(z) \text{ is an annotated null } N^{t_0} \\
  [s, e] & h_{t_0}(z) = t_0
\end{cases}
\]

Now we need to show \( h \) is a formula homomorphism from \( \sigma_k \) to \( u_1 \) and \( u_2 \). Wlog, we show this for \( u_1 \). When we apply \( h_{t_0} \) to \( \sigma_k \), we get the tuple

\[
w_1 = R(h_{t_0}(x_1), ..., h_{t_0}(x_m), h_{t_0}(y_1), ..., h_{t_0}(y_n), h_{t_0}(t)) = R(c_1, ..., c_m, v'_1, ..., v'_n, t_0)
\]
where $c_1, ..., c_m$ are constants and $v_j' (1 \leq j \leq n)$ is either a constant or an annotated null. Now by applying $h$ on $\sigma_k$ the following fact is obtained

$$R(h(x_1), ..., h(x_m), h(y_1), ..., h(y_n), h(t)) = R(c_1, ..., c_m, v_1, ..., v_n, [s, e])$$

where $v_j, 1 \leq j \leq n$, is either a constant or an interval-annotated null. Observe that $v_j = v_j'$ if $v_j'$ is a constant and $v_j = N^{(s,e)}$ if $v_j' = N^{(s,e)}$. Therefore, $u_1 = R(c_1, ..., c_m, v_1, ..., v_n, [s, e])$. Therefore, we can apply $\sigma_k$ on $u_1$ and $u_2$ (i.e. $c$-chase$^{\sigma_k}(u_1, u_2)$).

\begin{proof}

Each equality in $E_{J^c, \Sigma_k}$, such as $N^{(s,e)} = val$, is due to a tlc $c$-chase step on conflicting concrete facts $u_1$ and $u_2$, such that $u_1[T] = u_2[T] = [s, e]$. Based on Lemma 9 and Lemma 10, we know there is a tlc $a$-chase step for each pair of conflicting abstract facts $w_1$ and $w_2$ w.r.t $\sigma_k$ such that $w_1[T] = w_2[T] = t_0$ and $t_0 \in [s, e]$. Therefore, based on Definition 11, for each equality in $E_{J^c, \Sigma_k}$, there is an equivalent equality in $E_{J^c, \Sigma_k}$ up to renaming the point-annotated nulls.

The other direction is straightforward as well. Therefore, $[E_{J^c, \Sigma_k}] \equiv E_{J^c, \Sigma_k}$.

\end{proof}

\begin{proof}[Proof of Theorem 8]

This proof is very similar to the proof of Theorem 7. First we show $[c$-chase$^{\Sigma_k}_{J^c}(J_c)] \rightarrow a$-chase$^{\Sigma_k}_{J^c}[J_c]$. Let $w$ be a fact in $[c$-chase$^{\Sigma_k}_{J^c}(J_c)]$ such that $w = R(a_1, ..., a_m, t_0)$, where $a_i$ is either a constant or a point-annotated null. This means that there is a concrete fact $u$ in $J^c = c$-chase$^{\Sigma_k}_{J^c}(J_c)$ such that $w \in [u]$. Let $u = R(b_1, ..., b_m, [s, e])$, where $b_i$ is either a constant or an interval-annotated null. By definition of the semantic mapping, we have

- if $a_i$ is a constant, $b_i$ is a constant as well.
- if $a_i$ is a point-annotated null such as $N^{t_0}$, $b_i$ is an interval-annotated null such as $N^{(s,e)}$.
- $t_0 \in [s, e]$ for some positive integers $s$ and $e$.

The concrete fact $u$ is the result of $c$-chase$^{\Sigma_k}_{J^c}$ (Definitions 23 and 24) on concrete instance $J_c$ where $\Sigma_k$ is a set of temporal key constraints. We distinguish two cases:

- The concrete fact $u$ is in $J_c$ and it is not conflicting with any other concrete tuple with respect to $\Sigma_k$. Therefore, it will remain intact in $J^c$ after applying parallel chase $c$-chase$^{\Sigma_k}_{J^c}$.
- Some interval-annotated nulls in the fact $u$ were replaced by constants or other interval-annotated nulls based on the equalities in $E_{J^c, \Sigma_k}$ during a successful $c$-chase$^{\Sigma_k}_{J^c}$ step.

Now we need to show in either case, there is a fact $w'$ in $a$-chase$^{\Sigma_k}_{J^c}[J_c]$ that is equivalent to $w$ up to renaming point-annotated nulls (without changing their context)

**Case(a):** Since $u$ is a concrete fact in $J_c$ and $w \in [u]$ and we are a-chasing $[J_c]$ with the same set $\Sigma_k$ of tlc, the abstract fact $w$ is not conflicting with any other fact in $[J_c]$ w.r.t any $\sigma_k \in \Sigma_k$. Hence, $w'$ is equal to $w \in J_a = a$-chase$^{\Sigma_k}_{J_a}([J_c])$.

**Case(b):** There is a fact $u'$ in $J_c$ such that $u$ can be obtained from $u'$ by replacing its interval-annotated nulls by either constants or other interval-annotated nulls. These replacements are inferred during parallel $c$-chase by $E_{J^c, \Sigma_k}$. Based on Lemma 12, we have $[E_{J^c, \Sigma_k}] \equiv E_{J^c, \Sigma_k}$, thus, the point-annotated nulls that are replaced in $w' \in a$-chase$^{\Sigma_k}_{J_a}[w']$ are based on an equivalent set of equalities (i.e. $E_{J^c, \Sigma_k}$). Note that fact $w'$ is not identical to $w$ because we do not have control over the arbitrary point-annotated null that is chosen to replace the others which are equivalent to it, but they have the same context. Therefore, there is an abstract homomorphism from $w'$ to $w$. Proof of $a$-chase$^{\Sigma_k}_{J_a}([J_a]) \rightarrow [c$-chase$^{\Sigma_k}_{J_c}(J_c)]$ is similar.

\end{proof}
Proof of Lemma [13]

Proof. The proof is same as proof of case 1 in Lemma 3.4 in [7], which shows there is a homomorphism from the result of an arbitrary chase step in the sequence of chase steps to any instance that satisfies $\sigma_a$. Lemma [13] is a special case of Lemma 3.4, where the initial instance is $(I_a, \emptyset)$. Therefore there is no need for the assumption that there exists a homomorphism from $(I_a, \emptyset)$ to the instance $(J'_a)$ in our case) that satisfies $\sigma_a$. In other words, an abstract homomorphism from $(I_a, \emptyset)$ to $(I_a, J'_a)$ is the identity mapping (same as homomorphisms in [7]).

Proof of Lemma [15]

Proof. Wlog, we assume all tkcs in $\Sigma_a$ have one equality on rhs, that is:

$$\sigma : R(x_1, ..., x_m, y_1, t) \land R(x_1, ..., x_m, y_2, t) \rightarrow y_1 = y_2$$

Let $R(c_1, ..., c_m, v, t_0)$ be a fact in $J'_a$, where $c_1, ..., c_m$ are constants and $v$ is either a constant or a point-annotated null. We need to show that $R(h_1(c_1), ..., h_1(c_m), h_1(v), h_1(t_0))$ is in $J'_a$. If $R(v_1, ..., v_n, t_0) \in J_a$, then since $h_1 : J_a \mapsto J'_a$, we have $R(h_1(v_1), ..., h_1(v_n), h_1(t_0)) \in J'_a$. Moreover, since $J'_a$ satisfies $\sigma$ it follows that $h_1(v)$ is equal to $h_1(v')$ in $J'_a$. Therefore, $R(h_1(v_1), ..., h_1(c_m), h_1(v'), h_1(t_0)) \in J'_a$.

Proof of Theorem [18]

Proof. First, we show $\text{[certain}_a(Q, I_a, M)] \subseteq \text{[certain}_a(Q, [I_a], M)$. Let $w$ be a tuple in $\text{[certain}_a(Q, I_a, M)]$, so $w$ is a complete abstract fact. There must be a complete concrete fact $u$ in $\text{[certain}_a(Q, I_a, M)$ such that $w \in [u]$. Based on Definition [17] $u \in \bigcap \{Q(J_a) \mid J_a$ is a concrete solution for $I_a\}$. Let $J'_c$ be the concrete solution produced by concrete chase algorithm on $I_c$ w.r.t $M$. Therefore, $u \in Q(J'_c)$ and $w \in [Q(J'_c)]$. Let $J_a$ be the abstract solution obtained by the abstract chase algorithm on $[I_a]$ w.r.t $M$. Based on Theorems [7] and [8] we know that $J_a$ is homomorphically equivalent to $[J'_c]$. Since $w$ is a complete fact in $[Q(J'_c)]$, unions of conjunctive queries are preserved under abstract homomorphisms, and abstract homomorphisms are identities on constants, $w$ is in $Q(J_a)$. The solution $J_a$ is a universal abstract solution (Corollary [16]). Therefore, $w$ is in the result of evaluating $Q$ on every abstract solution for $[I_a]$, which means $w \in \text{[certain}_a(Q, [I_a], M)$. 
For the other direction, we have to prove certain\(_a(Q, [I_c], \mathcal{M}) \subseteq [\text{certain}(Q, I_e, \mathcal{M})].\)

Let \(w \in \text{certain}_a(Q, [I_c], \mathcal{M}),\) which means \(w\) is a complete abstract tuple in:

\[
\bigcap \{Q(J_a) \mid J_a \text{ is an abstract solution for } [I_c]\}.
\]

Let \(J'_a\) be the abstract solution obtained by abstract chase algorithm on \([I_c]\) w.r.t \(\mathcal{M}\). We have \(w \in Q(J'_a)\). Note that \(J'_a\) is a universal abstract solution. Let \(J_t\) be the concrete solution produced by concrete chase on \(I_e\) w.r.t \(\mathcal{M}\). Based on Theorems 2 and 3, we know that \(w \in [Q(J_t)]\). Now we have to show that \(w\) is in \([Q(J'_a)]\), where \(J'_c\) is any arbitrary concrete solution for \(I_c\). Suppose \(w \notin [Q(J'_a)]\). It is obvious that \([J'_a]\) is also a solution for \([I_c]\) w.r.t \(\mathcal{M}\). Since \(J_a\) is a universal abstract solution, there must be an abstract homomorphism from \(J_a\) to \([J'_a]\). However, it is a contradiction with universality of \(J_a\) if \(w \in Q(J_a)\) and not in \([J'_a]\). Therefore \(w\) must be in \([J'_a]\).

### Naïve evaluation for abstract instances

Naïve evaluation [1, 3, 9, 7] is a technique commonly used in the literature to find certain answers for unions of conjunctive queries on naïve tables (i.e. tables with labeled nulls). Let \(J_a\) be a universal abstract solution for a source instance \(I_a\) and a schema mapping. Let \(Q\) be a union of conjunctive queries of the form \(Q_1 \lor \ldots \lor Q_m\), where each \(Q_i\) \((1 \leq i \leq m)\) is a conjunctive query (i.e. \(\exists y \beta(x, y, t)\)), where \(\beta\) is a conjunction of atomic formulas (i.e. \(\beta = \beta_1 \land \ldots \land \beta_n\)) over the target schema. In order to calculate the result of \(Q\) on the abstract instance \(J_a\) with naïve evaluation, denoted by \(Q(J_a)\), the following steps should be taken:

- All point-annotated nulls are treated as distinct fresh constants. In particular, the point-annotated null \(N^{t_0}\) is replaced with a fresh constant \(c^{t_0}\) everywhere it occurs in \(J_a\).
- Query \(Q\) is evaluated by finding all formula homomorphisms from variables in \(Q\) to \(J_a\).
- In particular, the variable \(t\) is mapped to a time point. The result of this step is denoted by \(Q(J_a)\).

#### Tuples with fresh constants are dropped from \(Q(J_a)\).

### Proof of Theorem 19

**Proof.** Wlog, we prove the theorem for one disjunct of \(Q'\). Let \(Q\) be an arbitrary disjunct in \(Q'\). This equality is proved by first showing that \([Q(J_a)] \subseteq [Q([I_c])_1]\) and then \([Q(J_c)] \subseteq [Q(J_a)]\). Like the previous proofs, for each direction, we take a tuple in the left hand side and show it must be in the right hand side as well and vice versa.

For the first part, let \(w = (a_1, \ldots, a_k, t_0)\) be a complete abstract tuple in \([Q(J_a)]\). This means there should be a complete concrete tuple \(u = (a'_1, \ldots, a'_k, [s, e])\) in \(Q(J_c)\) such that \(w \in [u]\). So, \(t_0 \in [s, e]\). Furthermore, since naïve evaluation discards tuples with fresh constants (that were nulls), all non-temporal values in \(w\) and \(u\) must be constants. That is, \(a_i = a'_i\) and they are constants, \(1 \leq i \leq k\).

Having \(u \in Q(J_c)\) means that there is a formula homomorphism \(h\) from the variables in \(Q\) to the target instance \(J_c\) such that \(u = (h(\bar{x}), h(t))\). We need to show that there is a formula homomorphism \(h'\) from \(Q\) to \([J_c]\) such that \(w = (h'(\bar{x}), h'(t))\). We build \(h'\) as follows:

\[
h'(z) = \begin{cases} h(z) & \text{if } h(z) \text{ is a constant and } z \text{ is a non-temporal variable} \\ n^{c_{t_0}} & \text{if } h(z) \text{ is a fresh constant } n^{c_{t_0}} \\ t_0 & \text{if } h(z) = [s, e] \end{cases}
\]

Observe that \(h'(\bar{x}) = h(\bar{x})\) because \(h(\bar{x})\) consists of constants. Now we have to show \(h'\) is a formula homomorphism from \(Q\) to \([J_c]\). Let \(\beta(x_1, \ldots, x_k, y_1, \ldots, y_m, t)\) be an arbitrary atom in
From Theorem 18: 

Naïve evaluation of unions of conjunctive queries on a universal abstract solution 

Proof of Theorem 20: 

Observe that \( h(z) = \begin{cases} 
    h'(z) & \text{if } h'(z) \text{ is a constant and } z \text{ is a non-temporal variable} \\
    nc_{j,s,e}^{z} & \text{if } h'(z) \text{ is a fresh constant } nc_{j,s,e}^{z} \\
    [s,e] & \text{if } h'(z) = t_{0} 
\end{cases} \)

Observe that \( h(\bar{x}) = h'(\bar{x}) \), because \( h'(\bar{x}) \) is mapped to constants. Let \( \beta_{i}(x_{1},\ldots,x_{k},y_{1},\ldots,y_{m},t) \) be an arbitrary atom in \( Q \). Under \( h' \), the atom maps to an abstract fact \( w' \) in \( [J_{c}] \) such that \( w' = \beta_{i}(a_{1},\ldots,a_{k},v'_{1},\ldots,v'_{m},t_{0}) \), where \( a_{i}, 1 \leq i \leq k \) are constants and \( v_{j}, 1 \leq j \leq m \) are constants or fresh constants. Under \( h \), the atom \( \beta_{i} \) is \( w' = \beta_{i}(a_{1},\ldots,a_{k},v'_{1},\ldots,v'_{m},[s,e]) \), where \( v'_{j} = v_{j} \) if \( v_{j} \) is a constant and \( v'_{j} = nc_{j,s,e}^{z} \) if \( v_{j} \) is a fresh constant \( nc_{j,s,e}^{z} \).

Observe that \( w' \in [u'] \). Since \( J_{c} \) is normalized, there is a unique clopen interval among all time values in \( J_{c} \) that contains \( t_{0} \). As mentioned earlier, this interval is \([s,e] \). Hence, if \( w' \) is a fact in \([J_{c}] \), then \( u' \) is the unique fact in \( J_{c} \) such that \( w \in [u'] \). Since there is only one temporal variable \( t \) and \( \beta_{i}(h(\bar{x}),h(\bar{y}),h(t)) \) is in \([J_{c}] \), it follows that \( \beta_{i}(h(\bar{x}),h(\bar{y}),h(t)) \) is a fact in \( J_{c} \). This shows \( h \) is a formula homomorphism from \( Q \) to \( J_{c} \).

Hence, \( u = (h(\bar{x}),h(t)) \) is in \( Q(J_{c})_{\downarrow} \), and so we have \( w \in [Q(J_{c})_{\downarrow}] \). Since the disjuncts of \( Q' \) are independent and \( Q \) is an arbitrary disjunct in \( Q' \), it is concluded that \([Q'(J_{c})_{\downarrow}] = Q'(J_{c})_{\downarrow} \). 

Proof of Theorem 20

Proof. Naïve evaluation of unions of conjunctive queries on a universal abstract solution \([J_{c}] \) gives certain answers. The proof is the same as in \([3,7] \). Therefore, 

\[ \text{certain}_{u}(Q,[J_{c}],[M]) = Q([J_{c}]_{\downarrow}) \]

From Theorem 13 

\[ [\text{certain}_{c}(Q,J_{c},[M])] = Q([J_{c}]_{\downarrow}) \]

By replacing \( Q([J_{c}]_{\downarrow}) \) with \([Q(J_{c})_{\downarrow}] \) from Theorem 10 

\[ [\text{certain}_{c}(Q,J_{c},[M])] = [Q(J_{c})_{\downarrow}] \]

\[ \blacksquare \]