UNIVERSITY OF CALIFORNIA
SANTA CRUZ

SOME ASPECTS OF TEMPORAL DATA EXCHANGE

A dissertation submitted in partial satisfaction of the
requirements for the degree of

DOCTOR OF PHILOSOPHY

in

COMPUTER SCIENCE

by

Zehui Cheng

September 2023

The Dissertation of Zehui Cheng
is approved:

Professor Phokion G. Kolatis, Chair

Professor Rada Chirkova

Professor Lindsey Kuper

Peter Biehl
Vice Provost and Dean of Graduate Studies

Contents

List of Figures

List of Tables

Abstract

Some Aspects of Temporal Data Exchange

by

Zehui Cheng

The transformation of data from one schema, called the *source* schema, to another schema, called the *target* schema, is the focus of data exchange. The primary challenge in data exchange is to develop methods that transfer data from a source schema to a target schema using schema mappings that specify the relationship between the two schemas so that the resulting data accurately reflects the source data. Over the past two decades, extensive research has been conducted on data exchange, starting from the original formalization of the problem for relational schemas. While temporal databases have been extensively studied for many years, the exploration of temporal data exchange is a relatively recent development. We investigate the notion of temporal schema mappings specifying the relationship between two temporal schemas. Temporal schema mappings contain at least one temporal relation symbol and use Allen's relations to describe the relationship between temporal variables. Temporal data in such databases can be represented either through time intervals (concrete time) or time points (abstract time). Taking this into account, we design a chase algorithm for temporal data exchange settings with multiple temporal variables in the context of concrete time. We demonstrate that this algorithm produces universal solutions, provided that it does not fail. Furthermore, we investigate the relationship between universal solutions in the context of concrete time and universal solutions in the context of abstract time. We find that challenges arise even when temporal schema mappings involve a single temporal variable, but we also identify scenarios where these challenges can be overcome.

Different applications may use data structured in various formats, such as relational data or RDF. Depending on the applications, the target data may need to adhere to a relational schema, while others require it to conform to an RDF-expressed domain ontology. Our research also focuses on the following problem: Given a set of temporal schema mappings, how to exchange data with temporal information from a relational source schema into a target RDF-expressed ontology, so that we can enrich both the data and the ontology with temporal information from the relational sources? To address this challenge, we design a domain-independent algorithm that materializes target RDF data via a version of data exchange. This algorithm ensures the enrichment of both the data and the ontology with temporal data obtained from the sources.

The aforementioned work assumes that the temporal schema mappings have already been derived. Earlier studies on relational data exchange addressed the problem of active learning of (non-temporal) global-as-view (GAV) schema mappings. In the last part of this dissertation, we initiate an investigation on the automatic derivation of relational-to-RDF temporal schema mappings (based on our earlier work on relational-to-RDF temporal data exchange). To achieve this, we design an active learning algorithm and validate it using the metadata generator iBench, which was originally developed to generate relational-to-relational schema mappings and their corresponding data examples. We enhance iBench with additional features, referred to as temporal iBench, to enable the generation of relational-to-RDF temporal schema mappings and their data examples. Additionally, we carry out a comprehensive experimental evaluation which demonstrates the effectiveness of our active learning algorithm.

Chapter 1

Introduction

In this chapter, we offer an introduction to our research and summarize the main contributions. Our research consists of three parts: *universal solutions in relational-to-relational data exchange*, *relational-to-RDF temporal data exchange*, and *active learning of temporal schema mappings*. The introduction and summary of each part are presented in Section 1.1, Section 1.2, and Section 1.3, respectively. After that, we provide the structure of this dissertation in Section 1.4.

1.1 Universal Solutions in Relational-to-relational Data Exchange

Data exchange is concerned with the transformation of data structured under one schema, called the *source* schema, into data structured under a different schema, called the *target* schema. Since the original formalization of the data exchange problem between relational schemas in [32], an extensive study of data exchange has been carried out in several different settings, including XML data exchange [11], data exchange between graph databases [18], and relational to Resource Description Framework *(RDF)* data exchange [22]; an overview of the main results in this area can be found in the monograph [14]. Temporal databases constitute a mature area of research that has been studied in depth over several decades; for overviews, see, e.g., the book [58] or the book chapter [30]. Data exchange and temporal databases have advanced independently and, rather surprisingly, their paths did not cross until recently, when Golshanara and Chomicki [36] published the first paper on *temporal data exchange*, that is, data exchange between temporal databases.

Data exchange is formalized using *schema mappings*, i.e., tuples of the form $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$, where $\mathbf{S}$ is the source schema, $\mathbf{T}$ is the target schema, and Σ is a finite set of constraints in some suitable logical formalism that describe the relationship between source and target. Every fixed schema mapping $\mathcal{M}$ gives rise to the *data exchange problem with respect to* $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$: given a source instance I, find a *solution* for I, that is, a target instance J so that $(I,J) \models \Sigma$. In general, no solution for I may exist or multiple solutions for I may exist. Fagin et al. [32] introduced the concept of a *universal* solution and made a case that universal solutions are the "best" solutions to materialize, provided solutions exist. In a precise sense (formalized using homomorphisms), a universal solution is the most general solution, thus it embodies no more and no less information than what the constraints in Σ specify. By now, universal solutions have been widely adopted as the preferred semantics in data exchange; furthermore, a concerted research effort has been dedicated to discovering when universal solutions exist and how to compute them. The main tool for computing universal solutions is the *chase* algorithm [32] and its variants (see [38] for a survey).

In temporal databases, there are two different models of time, namely, *concrete* time and *abstract* time; in the first model, time is represented by time intervals, while in the second by time points [30, 65]. Concrete temporal databases can be converted to abstract temporal databases using the *semantic function*[1] $[\![.]\!]$, which takes as input a concrete temporal database D and returns as output the abstract temporal database $[\![D]\!]$ where intervals of time in D are replaced by all points of time in them. The semantic function is often deployed to transfer results about concrete temporal databases to results about abstract temporal databases.

As already mentioned, Golshanara and Chomicki [36] were the first to investigate temporal data exchange, namely *relational-to-relational temporal data exchange*. Specifically, they considered *temporal* schema mappings $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st}, \Sigma_t)$, where $\mathbf{S}$ and $\mathbf{T}$ are schemas consisting of relation symbols each of which has exactly one temporal attribute, Σ_{st} is a set of temporal source-to-target tuple-generating dependencies (temporal s-t tgds) and Σ_t is a set of temporal target equality-generating dependencies (temporal target egds) with the restriction that each such constraint contains exactly one temporal variable. This means that each constraint in Σ_{st} is of the form $\forall \mathbf{x} \forall t(\varphi(\mathbf{x}, t) \rightarrow \exists \mathbf{y} \psi(\mathbf{x}, \mathbf{y}, t))$, where t is the only temporal variable, $\varphi(\mathbf{x}, t)$ is a conjunction of source atoms, and $\psi(\mathbf{x}, \mathbf{y}, t)$ is a conjunction of target atoms. Also, each constraint in Σ_t is of the form $\forall \mathbf{x} \forall t(\theta(\mathbf{x}, t) \rightarrow x_k = x_l)$, where t is the only temporal variable, the variables x_k and x_l are variables in $\mathbf{x}$, and $\theta(\mathbf{x}, t)$ is a conjunction of target atoms.

Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st}, \Sigma_t)$ be a temporal schema mapping as above. Golshanara and Chomicki's main result [36] is the discovery of a variant of the chase algorithm that has the following properties: (a) it runs in polynomial time; (b) given a concrete source instance I, it detects if $[\![I]\!]$ has a solution with respect to $\mathcal{M}$; and (c) if $[\![I]\!]$ has such a solution, then it produces

[1]In the temporal databases literature, $[\![.]\!]$ is called the *semantic mapping*. Here, we chose to call it the *semantic function* to avoid confusion with the term *schema mapping*, which will be used repeatedly throughout this chapter.

a concrete target instance J such that J is *semantically adequate* for I, i.e., the abstract target instance $[\![J]\!]$ is a universal solution for the abstract source instance $[\![I]\!]$. In the sequel, we call *normalizing* chase the variant of the chase used in [36]. It is a natural extension of the chase algorithm to temporal dependencies, but with the twist that first a *normalization* step is performed on the given concrete source instance I and then the temporal s-t tgds are applied to the resulting normalized instance $\mathcal{N}(I)$; after this, a second normalization step is performed on the resulting concrete target instance and then the temporal target egds are applied. Our investigation began when we noticed that Golshanara and Chomicki [36] do not address the question of whether or not the normalizing chase always produces a universal solution for a given concrete source instance, provided the normalizing chase does not fail. As a matter of fact, Golshanara and Chomicki never introduce the notion of a universal solution for a concrete source instance.

Summary of results

- The first part of our research focuses on universal solutions in the context of concrete time. We define the notions of a *concrete schema mapping* with multiple temporal variables in the constraints, a *homomorphism* between concrete instances, a *concrete solution*, and a *concrete universal solution*. This framework contains (standard) data exchange between relational schemas as a special case. We then give a polynomial-time *concrete chase* algorithm for concrete schema mappings and concrete source instances, and we show that the chase algorithm can produce concrete universal solutions for the given concrete source instance, provided the concrete chase algorithm does not fail on that instance. We also show, however, that the concrete chase algorithm may fail, yet solutions (and, in fact, universal solutions) for the given concrete source may exist. Therefore, the concrete chase algorithm is incomplete. In view of this finding, we identify quite tight sufficient conditions on concrete schema mappings, so that the concrete chase algorithm is complete, i.e., if the chase algorithm does not fail on the given concrete source instance, then it produces a universal solution, while if it fails, then no solution for that instance exists. Specifically, the aforementioned sufficient conditions consist of two parts (a) every concrete s-t tgd is full (i.e., its consequent contains no existential quantifiers); (b) if a concrete s-t tgd is not full, then it contains exactly one temporal variable, and this temporal variable occurs in every atom of the consequent of the concrete s-t tgd; moreover, every target egd contains at most one temporal variable.

We also consider schema mappings whose constraints contain at most one temporal variable, and we explore the relationship between universal solutions in the context of concrete time and universal solutions in the context of abstract time. By doing so, we aim to gain insight into the semantic adequacy of universal solutions in the context of concrete time.

In particular, we investigate the question: Which temporal schema mappings admit semantically adequate concrete universal solutions? We make some progress towards answering this question by identifying sufficient conditions that guarantee the existence of semantically adequate concrete universal solutions.

This first part of our research is presented in Chapter 3. We published this work as a conference paper in the $27th$ International Symposium on Temporal Representation and

Reasoning (TIME 2020) [28] and as a journal paper invited in the special issue from TIME 2020 of Journal Information and Computation [29].

1.2 Relational-to-RDF Temporal Data Exchange

In various application domains spanning industry, government, science, and global health, data collection is often carried out independently by different teams over time. While some real-life applications format their source and target data in relational databases, other applications require the target data format to align with standard domain vocabularies. These vocabularies, known as domain *ontologies*, are developed by experts and include concepts, relationships, and domain rules governing their interactions. Our work specifically addresses a common scenario in which ontologies and ontology-compliant data are expressed using *RDF/S* capabilities, which encompass the RDF data model [54] enriched with additional *RDFS* specifications [55]. In this scenario, namely relational-to-RDF data exchange scenario, the source data remains in a relational format.

In applications conforming to this relational-to-RDF data exchange scenario, such as studies on antimicrobial resistance *(AMR)* [51], the source data may contain crucial temporal information. However, the applicable target domain ontologies lack temporal components. (In AMR this is the case with the Antibiotic Resistance Ontology (ARO).[2]) At the same time, solutions for relational-to-RDF data exchange problem as found in studies like [49], do not directly apply here, as they do not incorporate the temporal semantics of the data in easy-to-use ways. Consequently, temporal information from sources may be lost during the data

[2]`http://www.obofoundry.org/ontology/aro.html`

exchange process, making it hard or even impossible for domain scientists to efficiently obtain accurate answers to temporal queries based on the source data and the target ontologies. Gutierrez, Hurtado, and Vaisman [41] augmented RDF Schema with temporal components. This enables the formalization and study of data exchange between temporal relational databases and temporally-augmented RDF. The second part of our research focuses on the data exchange problem of how to transfer the temporal data from temporal relational databases into RDF-expressed ontologies enriched with temporal features. This work was completed in collaboration with researchers Jing Ao and Rada Chirkova from North Carolina State University. The details are presented in Chapter 4, and was published as a short conference paper in the proceedings of Advances in Databases and Information Systems - 24th European Conference (ADBIS 2020) [9] and as a journal paper in the ACM Journal of Data and Information Quality [10].

Summary of results We formally define the relational-to-RDF temporal data exchange problem for source schemas that may also include relation symbols with no temporal attributes and target schemas that are RDF-expressed ontoligies. We adopt the concept of temporal annotation from existing research on RDF graphs and use it to enrich RDF with temporal components in such a way that we can define the notion of temporal RDF graph schema. We then define notions of a temporal RDF graph schema, an instance of a temporal RDF graph schema, a relational-to-RDF full s-t tgd with multiple temporal variables and relations between temporal variables, a GAV constraint, and a relational-to-RDF schema mapping. We then design a chase algorithm, which takes a relational-to-RDF temporal schema mapping and a temporal relational database as inputs. We show that the results generated by the chase algorithm are universal solutions for the given source instance w.r.t. the given relational-to-RDF schema mapping.

1.3 Active Learning of Temporal Schema Mappings

The aforementioned data exchange problem, namely relational-to-relational temporal data exchange and relational-to-RDF temporal data exchange, are facilitated by the wide usage of schema mappings. Real-world applications can benefit from the utilization of schema mappings. However, designing and refining schema mappings manually by domain experts can be laborious, and different experts may produce different schema mappings. Hence, it is crucial to develop an automatic or semi-automatic tool for generating schema mappings. Several frameworks exist for generating schema mappings between relational schemas, such as the fitting framework, the repair framework, the learning framework [62, 64], and the Interactive Mapping Specification (IMS) framework proposed in [25]. In particular, paper [62] initiated the investigation of GAV schema-mapping generation in the learning framework, where a GAV schema mapping is defined as a relation-to-relational schema mapping specified by GAV constraints. Their work primarily focused on the theoretical aspects and introduced the EXACTGAV algorithm, which aims to identify a goal schema mapping. Building upon the EXACTGAV algorithm, paper [64] developed the GAVLEARN algorithm to identify a GAV schema mapping in practical scenarios. However, none of the aforementioned papers considers temporal attributes and relationships between temporal attributes when generating schema mappings. In addition, all of the earlier work concentrates solely on schema mappings specifying relationship between non-temporal relational databases, and not on temporal data in the temporal relational databases. Moreover, the schema mappings that specify the relationship between a temporal source relational schema and a target temporal RDF graph schema are under explored. Thus, the third part of our research presented in Chapter 5 aims to investigate the automatic or semi-automatic generation of relational-to-RDF temporal schema mappings, and it will be submitted for publication in the near future. The main contributions in the third part of our work is outlined below:

Summary of Results In the research described in Section 1.1 and Section 1.2, we assume that relational-to-relational/ relational-to-RDF schema mappings are carefully designed by domain experts and are provided in the data exchange problem. In the last part of this dissertation, we study the data exchange problem from temporal relational database to RDF when no relational-to-RDF temporal schema mappings are provided. Our work aims to derive schema mappings based on a set of data examples. We first define the notion of a symbolic instance for a relational-to-RDF full temporal s-t tgd. With the symbolic instances, we are able to represent a relational-to-RDF full temporal s-t tgd by a pair of symbolic instances, one for the left-hand side of the s-t tgd and one for the right-hand side of the s-t tgd. After that, we define the notion of a canonical constraint which is a relational-to-RDF full s-t tgd where the left-hand side is a conjunction of relational atoms, each of which is transferred from a fact in a given source instance; and the right-hand side is a conjunction of RDF atoms, each of which is transferred from a triple in a given target instance. We first identify challenges and then design an algorithm to generate a canonical constraint for a given source instance and a target instance.

Generating canonical constraints plays a crucial role in the active learning algorithm, which takes a set of data examples as input and returns a relational-to-RDF GAV schema mapping that best describes the data examples. A data example is a source instance and a target instance which is a universal solution for the source instance with a specific schema mapping. A set of data examples provide the information about how source data is transferred to align with the target RDF graph schema in the data exchange process transferring temporal relational databases into RDF. We take EXACTGAV as our starting point in designing an active learning algorithm. We repeatedly retrieve a GAV constraint from canonical constraints generated from data examples.

We also enhance the existing metadata generator *iBench* with a new feature that enables it to generate benchmark data specifically tailored for temporal data exchange from temporal relational databases to RDF data. We produce schema mappings and data examples by the metadata generator, and report an extensive experimental evaluation of our active learning algorithm.

1.4 Organization of This Dissertation

The remainder of the thesis is structured as follows. In Chapter 2, we provide the reader with an introduction to the essential background and preliminary concepts, including the data exchange problem and temporal databases. As previously mentioned, our main contributions are presented from Chapter 3 to Chapter 5. Lastly, in Chapter 6 we conclude our research, and we discuss some directions for future research.

1.5 Related Work

In this section, we provide additional pointers to related work. Specifically, we will provide a literature review of research related to schema mappings, with a particular focus on the operations on schema mappings (see Section 1.5.1), as well as a review of studies concerning chase procedures (see Section 1.5.2). Moreover, we will also present related research in the domain of data exchange for non-relational data (see Section 1.5.3), including XML documents and graph databases. We will also provide additional references to temporal databases, particularly focusing on studies that investigated both abstract and concrete temporal databases (see Section 1.5.4). As mentioned earlier, the third part of our work explores the relational-to-RDF temporal schema mapping generation. Previous studies primarily focused on deriving schema mappings for relational databases, and no existing work investigated how to generate relational-to-RDF temporal schema mappings using an automatic tool. As stated before, there are several frameworks for schema mapping generation in previous studies, and we will elaborate on these frameworks in Section 1.5.5.

1.5.1 Operations on Schema Mappings

Schema mappings play a significant role in the data exchange problem. They are usually specified using a high-level declarative formalism that describes correspondences between different schemas at a logical level. In data management systems, schema mappings are also regarded as *metadata*. Bernstein [20] pointed out the importance of manipulation on schema mappings. To this end, Bernstein has introduced a general framework, called *model management*, where high-level algebraic operators were defined for manipulating schemas and mappings [34, 31]. Two of the most fundamental operators in this framework are the *composition* and the *inversion* of schema mappings. Intuitively, the composition operator combines successive schema mappings, in which the target of a schema mapping is also the source of another schema mapping, into a single schema mapping. In contrast, an inverse of a schema mapping, which specifies a relationship from a source schema into a new target schema, is a new mapping that describes the reverse relationship from the target schema to the source schema. The composition operator is widely used in the data exchange problem when target schemas evolve, called *schema evolution*, while the inversion operator is mainly used for exchanging data back to the source.

In fact, there have been extensive studies on the composition operator and the inversion operator after Bernstein's work [20]. In a subsequent research, Fagin et al. [34] studied the composition of a finite set of s-t tgds with a finite set of full s-t tgds. They showed that the composition may not be definable by any set (finite and infinite) of s-t tgds; furthermore, it may not be definable by any formula of least fixed-point logic. To solve this problem, they introduced a class of existential second-order formulas with function symbols and equalities called *second-order tgds*. Moreover, they proved that second-order tgds are closed under composition and have desirable properties for data exchange. On top of this work, in another research on the composition of schema mappings, Nash et al. [50] discussed a broader class of constraints, i.e., (first-order) embedded dependencies, full dependencies (i.e., full s-t tgds), and the second-order constraints that arise from Skolemizing embedded dependencies. For each of the three types of constraints, Nash et al. designed an algorithm to compute the composition and provided sufficient conditions on the input schema mappings such that the algorithm does not fail. Furthermore, they showed that full dependencies are not closed under composition and that the second-order dependencies that are not limited to s-t tgds are not closed under restricted composition. Arenas et al. [12] presented an overview and analysis of the prior research on the semantics of the composition operator and the inversion operator for schema mappings consisting of s-t tgds.

1.5.2 Chase Procedures

The chase procedure initially emerged as a crucial tool testing logical implication between sets of embedded dependencies [19, 47]. Subsequently, it underwent adaptations to accommodate various types of dependencies, including functional, join, and multivalued dependencies [68]. Moreover, the chase procedure demonstrated its utility in determining if two database instances (that may contain nulls) represent the same set of possible instances under a specific set of dependencies [48]. Ullman [66] leveraged the chase for testing query equivalence, while Johnson and Klug [44] applied the chase for containment under database constraints. Later on, the chase procedure was applied in data exchange [32, 33], data integration [46], and ontologies [27]. One of the main issues in this area is to check if the chase can terminate for a specific set of constraints on all instances. Earlier, we mentioned that Fagin et al. [32] proposed a chase algorithm for generating universal solutions in the data exchange problem. In addition, the concept of a *weakly acyclic* was defined; this concept yields a sufficient condition for the tractability of the existence of solutions and for efficient computation of a universal solution by the chase algorithm. For the purpose of summarizing the existing chase procedures, Onet [52] provided an overview of chase variants and their properties as well as their applications in data exchange.

1.5.3 Data Exchange for Non-relational Data Structure

In the last two decades, substantial research has been conducted regarding data exchange for relational data. Due to the increased need for exchange data between different platforms that structure data in various formats, Arenas and Libkin [11] initiated an investigation on basic theoretical issues of data exchange for XML documents. The paper defined the XML data exchange problem, where each constraint specifies the relationships between a source document type definition (DTD) and a target DTD. Indeed, such constraints refer to the hierarchical structure of the data. In a subsequent research, Amano et al. [5] studied the schema mappings between XML documents and considered horizontal navigation and data comparisons. Arenas et al. [13] provided a summary on the XML data exchange.

In addition, the notion of data exchange was extended into data transferring between graph databases. Barceló et al. [18] analyzed different possibilities for specifying mappings in graph databases. They developed an expressive mapping language based on graph query languages such that their mappings can express complex navigational properties, such as exporting entire paths satisfying some regular conditions. They also investigated solutions to the data exchange problem for graph databases (called the *graph data exchange problem*). Chase algorithms developed for the relational-to-relational data exchange problem were applied in this context to produce universal representatives of the target graph databases as the solution to the graph data exchange problem. In that paper, Barceló et al. showed that the chase procedure is no longer in polynomial time for the data exchange problem for graph databases. Thus, they identified a restricted class of mappings for which the chase procedure runs in polynomial time. However, the data exchange setting described in this paper does not fully capture the characteristics of practical graph databases. In a subsequent paper, Francis and Libkin [35] explored the data exchange for graph databases focusing on property graphs, which have been widely adopted as the standard model by graph databases implementation in practice.

1.5.4 Temporal Databases

Temporal databases are a specialized class of databases used to handle the temporal data. They are crucial for applications that are sensitive to time, such as financial systems, health care systems, and version control systems. Kulkarni and Michels [45] added suitable databases features to the $SQL : 2011$ standard to support temporal databases, and these features were applied to major database management systems such as DB2 and Oracle. In previous research, Toman [65] and Chomicki and Toman [30] in the field of temporal databases have introduced two perspectives on temporal data: the *abstract model of time* and the *concrete model of time*. The abstract model of time represents temporal data via a point-based data model (i.e., time points), while the concrete model of time represents temporal data via an interval-based data model (i.e., time interval). Chomicki and Toman [30] introduced query languages for the temporal databases in the abstract model of time and query languages for the temporal databases in the concrete model of time. They also showed that all first-order queries can be asked using a point-based first-order query language, which could be translated into an interval-based query language.

1.5.5 Relational-to-relational Schema Mapping Generation

As mentioned in Section 1.3, there is a body of work concerning the derivation of schema mappings between relational databases within the fitting, repair, and learning frameworks, as well as a different framework IMS.

Fitting Framework In the fitting framework, Alexe et al. [1] studied the following problem: given a finite set of data examples, determine whether or not there exists a relational-to-relational schema mapping, called a *Global-and-Local-as-View(GLAV) schema mapping*, that "fits" these data examples. Here, fitting means that the examples are universal examples, i.e., they are the "most general" examples for the schema mapping. To solve this problem, Alexe et al. developed a system for the interactive design of schema mappings. This system ensures that the resulting schema mapping is considered the "most general" if there exists a schema mapping that fits the input data examples. Additionally, they proved that this problem is harder than NP-complete.

Repair Framework Gottlob and Senellart [37] introduced the repair framework for schema-mapping discovery, in which schema mappings are derived from a single data example, and the derivation of a schema mapping is cast as an optimization problem. Subsequently, ten Cate et al. [63] studied this framework in depth and designed a polynomial-time $log(n)$-approximation algorithm for computing optimal schema mappings from a given set of data examples (where n is the combined size of the given data examples) for a restricted class of schema mappings.

IMS Framework IMS [25] starts from a set of *fully informative* data examples and derives GLAV schema mappings through a sequence of interactions with a non-expert user. However, the produced schema mappings may not be satisfied by the data examples under this framework.

Learning Framework In the learning framework, a comprehensive understanding has been established regarding the complexity of deriving GAV schema mappings. Alexe et al. [2] and ten Cate et al. [61] investigated the problem of which schema mappings can be uniquely characterized via a finite set of data examples. Alexe et al. [2] found that there is a class of GAV schema mappings that are uniquely characterizable by a finite set of universal examples. Building upon this result, ten Cate et al. [62] demonstrated that GAV mappings are learnable using universal examples (but not necessarily efficiently learnable) with queries that can return the universal solutions for a given source instance w.r.t. a schema mapping. Drawing from these discoveries, ten Cate et al. [62] designed the EXACTGAV algorithm to derive GAV schema mappings. EXACTGAV is in polynomial time with the assistance of two oracles that have the knowledge of a goal schema mapping. One of the two oracles requires the actual knowledge of the specification of the goal schema mapping. Therefore, EXACTGAV is a theoretical algorithm because of the difficulties in implementing such an oracle. Later, ten Cate et al. [64] proposed the GAVLEARN algorithm in practice scenario based on EXACTGAV. We provide more details about EXACTGAV and GAVLEARN in Chapter 5.

Chapter 2

Preliminaries

In this chapter, we will introduce the background material for this dissertation. In Section 2.1, we offer a summary of relational-to-relational data exchange problem and a solution to the problem. In Section 2.2, we will provide some background material from temporal databases.

2.1 Relational-to-relational Data Exchange

Relational-to-relational data exchange was first formalized and studied in [32]. Since this is the most well-developed and extensively studied variant of data exchange, we will use the term *standard* data exchange to refer to it.

Databases A relational schema is a finite collection $\mathbf{R}$ of relation symbols of the form $R(A_1,\ldots,A_k)$, where A_1, …, A_k are the *attributes* of R and k is its arity. An $\mathbf{R}$-*instance* I is a finite collection of finite relations R^I, one for each relation symbol R in $\mathbf{R}$ and such that the arity of R^I matches that of R. In the rest of this dissertation, R denotes both relation symbol and relation that interprets it. If a tuple f occurs in a relation R, then we write $R(f)$ to denote this association, and we call it a *fact*. By definition, the *active domain* of an instance I, denoted by $adom(I)$, is the set of all values occurring in the relations of that instance.

Constraints and Schema Mappings Let $\mathbf{S}$ and $\mathbf{T}$ be two relational schemas, called the *source* schema and the *target* schema, where $\mathbf{S}$ and $\mathbf{T}$ have no relation symbols in common. Data exchange from $\mathbf{S}$ to $\mathbf{T}$ is formalized using constraints in some logical formalism that describe the relationship between these two schemas [32]. The most widely used such constraints are *source-to-target tuple-generating dependencies* (s-t tgds) and *target equality-generating dependencies* (target egds). An s-t tgd, also called a *GLAV constraint*, is a first-order sentence of the form $\forall \mathbf{x}(\varphi(\mathbf{x}) \rightarrow \exists \mathbf{y} \psi(\mathbf{x}, \mathbf{y}))$, where $\varphi(\mathbf{x})$ is a conjunction of source atoms, and $\psi(\mathbf{x}, \mathbf{y})$ is a conjunction of target atoms. Such constraints can express a variety of data transformation tasks, including copying a relation, projecting a relation, augmenting a relation with an extra column, and joining two or more relations, where, in each case, the result of the transformation is moved to the target [60]. A *Global-as-View(GAV)* constraint is a special class of GLAV where the consequent of the GAV constraint is a single atomic formula. A target egd is a first-order sentence of the form $\forall \mathbf{x}(\theta(\mathbf{x}) \rightarrow x_k = x_l)$, where $\theta(\mathbf{x})$ is a conjunction of target atoms and x_k, x_l are variables occurring in $\mathbf{x}$. Target egds include target key constraints as an important special case.

A *schema mapping* is a tuple $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st}, \Sigma_t)$, where $\mathbf{S}$ and $\mathbf{T}$ are disjoint relational schemas, the set Σ_{st} is a finite set of s-t tgds, and Σ_t is a finite set of target egds.

In the remainder of this chapter, we will use the terms *standard s-t tgds*, *standard target egds*, and *standard schema mappings* for the preceding concepts.

Values in Source and Target Instances The source instances contain values from a countable domain $\textsc{Const}$ of objects, called *constants*, while the target instances may contain values from the union $\textsc{Const} \cup \textsc{Null}$, where $\textsc{Null}$ is a countable set of distinct *labeled nulls* $N_1, N_2, \ldots$, which are typically used to witness the existentially quantified variables in the right-hand sides of standard s-t tgds. Thus, a labeled null represents an unknown value.

Combined Instances Let $\mathbf{S}$ and $\mathbf{T}$ be two disjoint schemas and let I and J be two instances over the $\mathbf{S}$ and $\mathbf{T}$, respectively. A *combined instance* is a tuple $\langle I, J\rangle$ over the schema $\mathbf{S} \cup \mathbf{T}$. Sometimes, we will use the term *instances* to refer to combined instances.

Solutions, Homomorphisms, and Universal Solutions Let $\mathbf{T}$ be a target schema and let J and J' be two target databases. As discussed above, the relations in J and J' may contain constants, labeled nulls as values.

Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st}, \Sigma_t)$ be a standard schema mapping and I a source instance. A target instance J is a *solution for I* w.r.t. $\mathcal{M}$ if $\langle I, J\rangle$ *satisfies* every standard s-t tgd in Σ_{st}, and J *satisfies* every standard egd in Σ_t. The semantics of satisfaction is the usual semantics of first-order logic, which we spell out below for the sake of completeness.

- $\langle I, J\rangle$ satisfies a standard s-t tgd $\forall \mathbf{x}(\varphi(\mathbf{x}) \to \exists \mathbf{y} \psi(\mathbf{x}, \mathbf{y}))$ means that for every assignment s from the variables in $\mathbf{x}$ to the active domain of I, if $s(\mathbf{x})$ is a tuple $\mathbf{a}$ such that $I \models \varphi(\mathbf{a})$, then there is an assignment s' from the variables in $\mathbf{x}$ and $\mathbf{y}$ to the active domain of J that agrees with s on $\mathbf{x}$ and assigns to the variables $\mathbf{y}$ a tuple $\mathbf{b}$ of constants and/or labeled nulls from the active domain of J, such that $J \models \psi(\mathbf{a}, \mathbf{b})$.

- J satisfies a standard target egd $\forall \mathbf{x}(\theta(\mathbf{x}) \to x_k = x_l)$ means that for every assignment s from the variables in $\mathbf{x}$ to the active domain of J, if $s(\mathbf{x})$ is a tuple $\mathbf{a}$ such that $J \models \theta(\mathbf{a})$, then $a_k = a_l$ (i.e., a_k and a_l are the same constant or the same labeled null).

A *homomorphism* from J to J' is a function h from the active domain of J to the active domain of J' such that: (a) if v is a constant, then $h(v) = v$; (b) if v is a labeled null N_j, then $h(v)$ is either a constant or a labeled null N_k; and c) if a fact $R(v_1, \ldots, v_m)$ belongs to a relation R^J of the instance J, then $R(h(v_1), \ldots, h(v_m))$ belongs to the relation $R^{J'}$ of the instance J'.

A target instance J is a *universal solution for I* w.r.t. $\mathcal{M}$ if J is a solution for I w.r.t. $\mathcal{M}$ and, for every solution J' for I w.r.t. $\mathcal{M}$, there is a homomorphism from J to J'. Universal solutions can be produced using the *chase* algorithm, which we describe next.

Chase Steps Let I be a source instance and let K be the current target instance in a run of the chase.

- A chase step over an s-t tgd for the instance $\langle I, K\rangle$: If $\sigma = \forall \mathbf{x}(\varphi(\mathbf{x}) \rightarrow \exists \mathbf{y}\psi(\mathbf{x}, \mathbf{y}))$ is an s-t tgd in Σ_{st} and if s is an assignment from every variable in $\mathbf{x}$ to $adom(I)$, such that $s(\mathbf{x})$ is a tuple $\mathbf{a}$ from the $adom(I)$ and $I \models \varphi(\mathbf{a})$, but $K \not\models \exists \mathbf{y}\psi(\mathbf{a}, \mathbf{y})$, then the chase generates a tuple $\mathbf{b}$ of distinct labeled nulls for the variables in $\mathbf{y}$ and adds tuples to the relations in K so that the resulting instance K' satisfies $\psi(\mathbf{a}, \mathbf{b})$. We say that (a) the constraint σ can be applied to $\langle I, K\rangle$ with assignment s; (b) the result of applying the constraint σ to $\langle I, K\rangle$ with the assignment s is $\langle I, K'\rangle$, and write $\langle I, K\rangle \xrightarrow{\sigma, s} \langle I, K'\rangle$.

- A chase step over a target egd for the instance $\langle I, K\rangle$: If $\forall \mathbf{x}(\theta(\mathbf{x}) \rightarrow x_k = x_l)$ is a target egd in Σ_t and if s is a variable assignment from every variable in $\mathbf{x}$ to $adom(K)$, such that $s(\mathbf{x})$ is a tuple $\mathbf{a}$ and $K \models \theta(\mathbf{a})$, then the following cases are considered: (1) if both a_k and a_l are labeled nulls, then one of the two is replaced by the other throughout K; (2) if one of

a_k and a_l is a constant and the other is a labeled null, then the labeled null is replaced by the constant throughout K; (3) if a_k and a_l are different constants, then the chase fails. We denote such chase step as $\langle I, K\rangle \xrightarrow{\sigma,s} \perp$ if the chase fails; otherwise, we denote the chase step as $\langle I, K\rangle \xrightarrow{\sigma,s} \langle I, K'\rangle$.

The Standard Chase Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st}, \Sigma_t)$ be a schema mapping and let I be a source instance. All s-t tgd chase steps are applied sequentially (this is called the standard s-t tgd chase round) and produce a target instance; then all target edg chase steps are applied sequentially to that target instance (this is called the standard target egd chase round), so that if the chase does not fail, then the resulting target instance is generated.

In what follows, we will use the term the *standard chase algorithm* to refer to the standard chase described above. If $\mathcal{M}$ is a schema mapping and I is a source instance, then the *standard chase algorithm* will start with $\langle I, \emptyset\rangle$. A successful chase will produce the result $\langle I, J\rangle$, where J is the result of the target instance generated by the standard chase algorithm on $\langle I, \emptyset\rangle$. In the sequel, we will use the notation $\text{chase}_{\mathcal{M}}(I)$ to denote the target instance produced by the standard chase algorithm on I. The next result was proved in [32].

Theorem 1. Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st}, \Sigma_t)$ be a standard schema mapping. If I is a source instance over the schema $\mathbf{S}$, then the following statements hold.

- If the standard chase algorithm does not fail on I, then the target instance $\text{chase}_{\mathcal{M}}(I)$ returned by this algorithm is a universal solution for I w.r.t. $\mathcal{M}$.
- If the standard chase algorithm fails on I, then there is no solution for I w.r.t. $\mathcal{M}$.

Furthermore, the running time of the standard chase algorithm is bounded by a polynomial in the size of I.

Example 1. We consider a data exchange scenario concerning research papers submitted for publication in scholarly conferences. A paper is first registered to a conference, but there may be registered papers that did not materialize to a submission. After a registered paper is actually submitted, then the status of the paper is "under submission" until it is assigned to reviewers. The status of the paper then is "under review". After the paper is reviewed and a decision is made to accept it, then the status of the paper is "to be published".

Let $\mathbf{S}$ be a source schema consisting of the following relation symbols:

$$\mathsf{Reg}(title, conference, author), \mathsf{uSub}(title, conference),$$

$$\mathsf{uRew}(title, conference), \mathsf{TbPub}(title, conference, year),$$

where the relation Reg lists papers registered to conferences, the relation uSub lists papers under submission, the relation uRew lists papers under review, and the relation TbPub lists papers to be published.

Let $\mathbf{T}$ be a target schema consisting of relation symbols

$$\mathsf{Pub}(title, conference, author, year, page) \text{ and } \mathsf{InPrcs}(title, conference, author, editor),$$

where the relation Pub lists papers published in the conference and the relation InPrcs lists papers in process.

Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st}, \Sigma_t)$ be a schema mapping where Σ_{st} consists of the constraints

$$\forall x_1, x_2, x_3, x_4 (\mathsf{Reg}(x_1, x_2, x_3) \wedge \mathsf{TbPub}(x_1, x_2, x_4) \rightarrow \exists y\, \mathsf{Pub}(x_1, x_2, x_3, x_4, y))$$

$$\forall x_1,x_2,x_3(\mathsf{uSub}(x_1,x_2)\wedge \mathsf{Reg}(x_1,x_2,x_3)\to \exists y\, \mathsf{InPrcs}(x_1,x_2,x_3,y))$$

$$\forall x_1,x_2,x_3(\mathsf{uRew}(x_1,x_2)\wedge \mathsf{Reg}(x_1,x_2,x_3)\to \exists y\, \mathsf{InPrcs}(x_1,x_2,x_3,y))$$

and Σ_t consists of the constraint

$$\forall x_1,x_2,x_3,x_4,x_5(\mathsf{InPrcs}(x_1,x_2,x_3,x_4)\wedge \mathsf{InPrcs}(x_1,x_2,x_3,x_5)\to x_4 = x_5)$$

Let I be the source instance whose relations are depicted in Table 2.1. Let $\text{chase}_{\mathcal{M}}(I)$ denote the universal solution produced by the standard chase algorithm on I w.r.t. $\mathcal{M}$; its relations are depicted in Table 2.2.

Table 2.1: The relations Reg, uSub, uRew, and TbPub in the source instance I.

(a) Reg

title	conference	author
a_1	b_1	c_1
a_2	b_1	c_2
a_3	b_1	c_1
a_4	b_2	c_3
a_5	b_2	c_4
a_6	b_2	c_4

(b) uSub

title	conference
a_5	b_2
a_6	b_2

(c) uRew

title	conference
a_3	b_1

(d) TbPub

title	conference	year
a_1	b_1	1
a_2	b_1	2
a_4	b_2	1

Table 2.2: The relations Pub and InPrcs of the universal solution $\text{chase}_{\mathcal{M}}(I)$.

(a) Pub

title	conference	author	year	page
a_1	b_1	c_1	1	N_1
a_2	b_1	c_2	2	N_2
a_4	b_2	c_3	1	N_3

(b) InPrcs

title	conference	author	editor
a_3	b_1	c_1	N_4
a_5	b_2	c_4	N_5
a_6	b_2	c_4	N_6

2.2 Temporal Databases

Abstract Model of Time Let $\mathbb{N} = \{1, 2, \dots\}$ be the set of all natural numbers. In the *abstract* model of time, natural numbers represent *time points*. In addition, $<$ is a discrete linear order on $\mathbb{N}$ without endpoints. For example, given two time points, t and t', the relation between them could be $t < t'$ or $t' < t$ or $t = t'$.

Concrete Model of Time Let $\mathbb{N} = \{1, 2, \dots\}$ be the set of all natural numbers. In the *concrete* model of time, closed-open intervals $[s, e) = \{t \in \mathbb{N} : s \leq t < e\}$, where s and e are natural numbers with $s < e$, represent *time intervals*. Unbounded time intervals of the form $[s, \infty)$ are also allowed. Let $[s, e)$ and $[s', e')$ be two arbitrary time intervals. These two time intervals can be related via one of Allen's relations [4], namely, o (for *overlaps*), d (for *during*), $\prec$ (for *before*), m (for *meets*), s (for *starts*), f (for *finishes*), and $=$ (for *equals*), which are defined as follows:

$$\mathsf{o} := \{([s,e),[s',e')) : s < s' < e < e'\} \text{ and } \mathsf{d} := \{([s,e),[s',e')) : s' < s < e < e'\},$$

$$\prec := \{([s,e),[s',e')) : s < e < s' < e'\} \text{ and } \mathsf{m} := \{([s,e),[s',e')) : s < e = s' < e'\},$$

$$\mathsf{s} := \{([s,e),[s',e')) : s' = s < e < e'\} \text{ and } \mathsf{f} := \{([s,e),[s',e')) : s' < s < e = e'\},$$

$$= := \{([s,e),[s',e')) : s = s' \text{ and } e = e'\}$$

For example, the time interval $[2010, 2013)$ represents the years 2010, 2011, and 2012, while the time interval $[2013, 2014)$ represents the year 2013. Below are examples of the Allen's relations o and m:

$$[2010,2013)\ \mathsf{o}\ [2011,2014) \text{ and } [2010,2013)\ \mathsf{m}\ [2013,2014)$$

Temporal Databases A *temporal* relation symbol is a relation symbol $R(A_1,\ldots,A_k)$ in which one or more of its attributes are designated as temporal attributes, i.e., they can only take temporal values and either all these values are from the concrete model of time or all these values are from the abstract model of time. In this dissertation, we assume that every temporal relation symbol has exactly one temporal attribute, which, without loss of generality, is the last attribute in the list. A *temporal* relational schema is a relational schema $\mathbf{R}$ containing at least one temporal relation symbol. For such a schema $\mathbf{R}$, a *concrete* $\mathbf{R}$*-instance* is an $\mathbf{R}$-instance in which the values of the temporal attributes are time intervals. An *abstract* $\mathbf{R}$*-instance* is an $\mathbf{R}$-instance in which the values of the temporal attributes are time points. We will use the term *temporal database* to refer to both abstract instances and concrete instances.

If I is an abstract $\mathbf{R}$-instance and if t is a time point, then the *snapshot* I_t *of* I *at time* t is the instance consisting of the non-temporal facts of I and the projections of the temporal facts of I at time t. More formally, I_t is the instance obtained from I as follows: if $R(A_1,\ldots,A_k)$ is a relation symbol of $\mathbf{R}$ with no temporal attribute, then $R^{I_t} = R^I$; if $R(A_1,\ldots,A_k)$ is a relation symbol of $\mathbf{R}$ in which A_k is the temporal attribute, then $R^{I_t} = \pi_{A_1,\ldots,A_{k-1}}(\sigma_{A_k=t}(R))$, where π is the projection operator and σ is the selection operator of relational algebra. Thus, every abstract **R**-instance I can be identified with the sequence $I_{t_0}, I_{t_1}, \ldots$ of its snapshots, where $t_0, t_1, \ldots$ are the time points in the active domain $adom(I)$ of I.

Chapter 3

Relational-to-relational Temporal Data Exchange

In this chapter, we formalize relational-to-relational temporal data exchange problem and explore universal solutions to this problem. In Section 3.1, we provide the definitions for key concepts related to concrete temporal data exchange and show how the chase algorithm can be extended to this setting. In Section 3.2, we investigate the semantic adequacy for the relational-to-relational temporal data exchange problem.

3.1 Concrete Chase Algorithm

3.1.1 Syntactic Notions

Let **R** be a temporal relational schema. An *atomic formula* over **R** is either of the form $R(x_1, \dots, x_k)$ if there is a relation symbol $R(A_1, \dots, A_k)$ in **R** with no temporal attribute, or of the form $R(x_1,\dots,x_{k-1},t)$ if there is a relation symbol $R(A_1,\dots,A_k)$ in $\mathbf{R}$ with a temporal attribute A_k. Let $\mathbf{S}$ be a temporal source relational schema and let $\mathbf{T}$ be a target relational schema that is disjoint from $\mathbf{S}$. In the rest of this dissertation, we will on occasion call atomic formulas over $\mathbf{S}$ the *source atoms* and call atomic formulas over $\mathbf{T}$ the *target atoms*. Furthermore, an *Allen atomic formula* is an expression of the form $t\ \delta\ t'$, where δ is one of Allen's relations and t, t' are two temporal variables.

The first step in formalizing data exchange between temporal relational schemas is to extend the concepts of s-t tgds and target egds to incorporate temporal variables and Allen's relations between them. Let $\mathbf{S}$ be a temporal source relational schema, and let $\mathbf{T}$ be a temporal relational schema disjoint from $\mathbf{S}$. In the concrete model of time, a *concrete s-t tgd* is a formula of the form:

$$\sigma_{st} = \forall\mathbf{x}\forall\mathbf{t}\big(\varphi(\mathbf{x},\mathbf{t}) \wedge \pi(\mathbf{t}) \to \exists\mathbf{y}\psi(\mathbf{x},\mathbf{y},\mathbf{t})\big), \tag{3.1}$$

where all temporal variables are those in $\mathbf{t}$, the formula $\varphi(\mathbf{x},\mathbf{t})$ is a conjunction of atomic formulas over the temporal schema $\mathbf{S}$, $\pi(\mathbf{t})$ is a Boolean combination of Allen atomic formulas involving temporal variables in $\mathbf{t}$, and the formula $\psi(\mathbf{x},\mathbf{y},\mathbf{t})$ is a conjunction of target atoms over $\mathbf{T}$. Furthermore, we assume that the universally quantified variables $\mathbf{x}$ and $\mathbf{t}$ appear free in $\varphi(\mathbf{x},\mathbf{t})$. A concrete s-t tgd must contain at least one temporal variable. A concrete s-t tgd is *full* if it contains no existential quantifiers $\exists\mathbf{y}$, i.e., it is of the form

$$\sigma_{st} = \forall\mathbf{x}\forall\mathbf{t}\big(\varphi(\mathbf{x},\mathbf{t}) \wedge \pi(\mathbf{t}) \to \psi(\mathbf{x},\mathbf{t})\big).$$

A *concrete target egd*, is a formula of the form:

$$\sigma_t = \forall\mathbf{x}\forall\mathbf{t}(\theta(\mathbf{x},\mathbf{t}) \wedge \rho(\mathbf{t}) \to x_k = x_l),$$

where the only temporal variables are those in $\mathbf{t}$, the formula $\theta(\mathbf{x},\mathbf{t})$ is a conjunction of target atoms, the formula $\rho(\mathbf{t})$ is a Boolean combination Allen atomic formulas involving variables from $\mathbf{t}$, and x_k, x_l are among the variables in $\mathbf{x}$. A concrete target egd must contain at least one temporal variable. Note that these concrete target egds also subsume concrete target egds of the form $\forall\mathbf{x}\forall\mathbf{t}(\theta(\mathbf{x},\mathbf{t}) \to x_k = x_l)$.

It should be pointed out that if a concrete s-t tgd σ_{st} contains exactly one temporal variable, then there is no point in having a subformula $\pi(\mathbf{t})$; also, if a concrete target egd σ_t contains exactly one temporal variable, then there is no point in having a subformula formula $\rho(\mathbf{t})$. The reason is that every such subformula evaluates to true or false, independently of the value of the temporal variable t.

We now discuss two examples of temporal data exchange.

Example 2. As in Example 1, we consider a data exchange scenario concerning the publication of papers in conferences, but we now take time into account. At a given point of time, each submitted paper can be in an "under submission" or in an "under review" or in a "to be published" status.

Let $\mathbf{S}$ be a temporal source schema consisting of the following relation symbols:

$$\mathsf{Reg}(title, conference, author), \mathsf{uSub}(title, conference, Stime),$$

$$\mathsf{uRew}(title, conference, Rtime), \mathsf{TbPub}(title, conference, year, Ptime),$$

where the relation Reg lists papers registered in conferences; the relation uSub lists papers and their time intervals during which those papers are under submission; the relation uRew lists papers and their time intervals during which they are under review, and the relation TbPub lists papers and their time intervals during which they are to be published. Let **T** be a temporal target schema consisting of relation symbols:

$$\mathsf{Pub}(title, conference, author, year, page, Atime) \text{ and } \mathsf{InPrcs}(title, conference, author, editor),$$

where the relation Pub now contains a new temporal attribute.

A concrete s-t tgd over **S** and **T** is

$$\forall x_1, x_2, x_3, x_4, t_1, t_2 (\mathsf{uRew}(x_1, x_2, t_1) \wedge \mathsf{TbPub}(x_1, x_2, x_3, t_2) \wedge \mathsf{Reg}(x_1, x_2, x_4) \wedge (t_1 \mathsf{m} t_2)$$
$$\rightarrow \exists y\, \mathsf{Pub}(x_1, x_2, x_4, x_3, y, t_2)).$$

A concrete target egd over **T** is

$$\forall x_1, x_2, x_3, x_4, x_5, x_6, t (\mathsf{Pub}(x_1, x_2, x_3, x_4, x_5, t) \wedge \mathsf{Pub}(x_1, x_2, x_3, x_4, x_6, t) \rightarrow x_5 = x_6).$$

Example 3. Let **S** be a temporal source schema consisting of the relation symbols

$$E(Name, Company, Time) \text{ and } S(Name, Salary, Time),$$

and let **T** be a temporal target schema containing a relation symbol

$$Emp(Name, Company, Salary, Position, Time).$$

A concrete s-t tgd over **S** and **T** is

$$\forall n, l, c, t (E(n, c, t) \wedge S(n, l, t) \rightarrow \exists p\, Emp(n, c, l, p, t)).$$

A concrete target egd over **T** is

$$\forall n, c, l, p_1, p_2, t (Emp(n, c, l, p_1, t) \wedge Emp(n, c, l, p_2, t) \rightarrow p_1 = p_2).$$

Definition 1 (Concrete Schema Mapping). A *concrete schema mapping* is a tuple $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st}, \Sigma_t)$, where $\mathbf{S}$ is a temporal source relational schema, $\mathbf{T}$ is a temporal target relational schema disjoint from $\mathbf{S}$, the set Σ_{st} is a finite set of standard s-t tgds or concrete s-t tgds, and Σ_t is a finite set of standard target egds or concrete target egds.

3.1.2 Semantic Notions

Values in Source and Target Instances As discussed earlier, in data exchange between relational schemas, the source instances contain values from a countable domain CONST of objects, called *constants*, while the target instances may contain values from the union CONST $\cup$ NULL, where NULL is a countable set of distinct *labeled nulls* $N_1, N_2, \ldots$ that represent unknown values. In concrete data exchange, the values occurring in source and target instances may also be time intervals. Furthermore, the use of null values in target instances requires delicate handling because such null values may need to take into account the temporal context in which they are introduced. For this reason, concrete target instances may contain values that are constants, time intervals in the concrete model of time, labeled nulls $N_1, N_2, \ldots$, and *concrete time-stamped* nulls, that is, null values of the form $N_1^{\mathbf{t}}, N_2^{\mathbf{t}}, \ldots$, where $\mathbf{t}$ is a finite sequence of time intervals. Two such concrete time-stamped nulls are equal if and only if they have the same subscript (label) and the same time-stamp (i.e., the same sequence of time intervals). Intuitively, a concrete time-stamped null represents unknown values in the context of its time-stamp. For example, a concrete time-stamped null $N_j^{[2,5)}$ represents three unknown values, one at time-point 2, one at time-point 3, and one at time-point 4.

Definition 2 (Concrete Solutions). Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st}, \Sigma_t)$ be a concrete schema mapping and let I be a concrete source instance.

A concrete target instance J is a *concrete solution for I* w.r.t. $\mathcal{M}$ if $\langle I,J\rangle$ *satisfies* every constraint in Σ_{st} and J *satisfies* every constraint in Σ_t. If the constraint is a standard s-t tgd or a standard target egd, then the semantics of satisfaction is the same as the semantics given in Section 2.1. We now spell out the semantics of satisfaction for concrete s-t tgds and concrete target egds.

- $\langle I,J\rangle$ satisfies a concrete s-t tgd $\forall\mathbf{x}\forall\mathbf{t}(\varphi(\mathbf{x},\mathbf{t})\wedge\pi(\mathbf{t})\to\exists\mathbf{y}\psi(\mathbf{x},\mathbf{y},\mathbf{t}))$ means that for every assignment s from the variables in $\mathbf{x}$ and $\mathbf{t}$ to the active domain of I, if $s(\mathbf{x})$ is a tuple $\mathbf{a}$ of constants and $s(\mathbf{t})$ is a tuple $\mathbf{i}$ of time intervals such that $I\models\varphi(\mathbf{a},\mathbf{i})\wedge\pi(\mathbf{i})$, then there is an assignment s' from the variables in $\mathbf{x}$, $\mathbf{y}$, and $\mathbf{t}$ to the active domain of J that agrees with s on $\mathbf{x}$ and $\mathbf{t}$, and assigns a tuple $\mathbf{b}$ of constants, labeled nulls, and time-stamped nulls to the variables $\mathbf{y}$, such that every time-stamped null in $\mathbf{b}$ has the time-stamp $\mathbf{i}$ and $J\models\psi(\mathbf{a},\mathbf{b},\mathbf{i})$.
- J satisfies a concrete target egd $\forall\mathbf{x}\forall\mathbf{t}(\theta(\mathbf{x},\mathbf{t})\wedge\rho(\mathbf{t})\to x_k=x_l)$ means that for every assignment s from the variables of in $\mathbf{x}$ and $\mathbf{t}$ to the active domain of J, if $s(\mathbf{x})$ is a tuple $\mathbf{a}$ and $s(\mathbf{t})$ is a tuple $\mathbf{i}$ of time intervals such that $J\models\theta(\mathbf{a},\mathbf{i})\wedge\rho(\mathbf{i})$, then $a_k=a_l$ (i.e., a_k and a_l are the same constant or the same labeled null N_j or the same time-stamped null $N_j^{\mathbf{i}'}$, where $\mathbf{i}'$ is some time-stamp that may be different from $\mathbf{i}$).

Definition 3 (**Homomorphisms and Concrete Universal Solutions**). Let $\mathbf{T}$ be a temporal target schema and let J and J' be two concrete target instances. As discussed earlier, the relations in J and J' may contain constants, labeled nulls, and time-stamped nulls as values.

A *homomorphism* from J to J' is a function h from the active domain of J to the active domain of J' such that: (a) if v is a constant or a time interval then $h(v) = v$; (b) if v is a labeled null N_j, then $h(v)$ is either a constant or a labeled null N_k; (c) if v is a time-stamped null $N_j^{\mathbf{i}}$, then $h(N_j^{\mathbf{i}})$ is a constant or a time-stamped null $N_k^{\mathbf{i}}$ with the same time-stamp or a labeled null N_k (without a time-stamp); (d) if a tuple $(v_1, \ldots, v_m)$ belongs to a relation R^J of J, then $h(v_1, \ldots, v_m)$ belongs to the relation $R^{J'}$ of J'.

The intuition behind this definition is that if there is a homomorphism from J to J', then J is "more general" than J'. Time-stamped nulls are "more general" than labeled nulls since the latter represent a single unknown value, while the former may represent multiple unknown values, depending on the time-stamp used. That explains the different treatment of labeled nulls and time-stamped nulls in conditions (b) and (c), respectively, in the definition.

A concrete target instance J is a *concrete universal solution for I* w.r.t. $\mathcal{M}$ if J is a concrete solution for I w.r.t. $\mathcal{M}$ and, for every solution J' for I w.r.t. $\mathcal{M}$, there a homomorphism from J to J'.

3.1.3 The Concrete Chase Algorithm

In the case of standard data exchange, universal solutions are produced using the (standard) chase algorithm as stated in Theorem 1. We now describe how the chase algorithm can be adapted from standard schema mappings to concrete schema mappings $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st}, \Sigma_t)$. Note that, in the rest of the paper, we regard a standard target egd of the form $\forall \mathbf{x}(\theta(\mathbf{x}) \rightarrow x_k = x_l)$ as a special case of the concrete target egd $\forall \mathbf{x} \forall \mathbf{t}(\theta(\mathbf{x}, \mathbf{t}) \wedge \rho(\mathbf{t}) \rightarrow x_k = x_l)$.

Definition 4 (Concrete Chase Steps). Let I be a concrete source instance and let K be the current concrete target instance in the run of the chase.

- A concrete chase step over a standard s-t tgd[1] for the instance $\langle I, K\rangle$: If $\sigma = \forall \mathbf{x}(\varphi(\mathbf{x}) \rightarrow \exists \mathbf{y}\psi(\mathbf{x}, \mathbf{y}))$ is a standard s-t tgd in Σ_{st} and if s is an assignment from the variables in $\mathbf{x}$ to the active domain $adom(I)$ of I such that $s(\mathbf{x})$ is a tuple $\mathbf{a}$ from $adom(I)$ and $I \models \varphi(\mathbf{a})$, but $K \not\models \exists \mathbf{y}\psi(\mathbf{a}, \mathbf{y})$, then the chase generates a tuple $\mathbf{b}$ of distinct labeled nulls for the variables in $\mathbf{y}$ and adds tuples to the relations in K so that the resulting instance K' satisfies $\psi(\mathbf{a}, \mathbf{b})$. We say that the constraint σ can be applied to $\langle I, K\rangle$ with assignment s; we also say that the result of applying the constraint σ to $\langle I, K\rangle$ with the assignment s is $\langle I, K'\rangle$, and write $\langle I, K\rangle \xrightarrow{\sigma, s} \langle I, K'\rangle$.

- A concrete chase step over a concrete s-t tgd for the instance $\langle I, K\rangle$: If $\forall \mathbf{x}\forall \mathbf{t}(\varphi(\mathbf{x}, \mathbf{t}) \wedge \pi(\mathbf{t}) \rightarrow \exists \mathbf{y}\psi(\mathbf{x}, \mathbf{y}, \mathbf{t}))$ is a concrete s-t tgd in Σ_{st} and if s is an assignment from the variables in $\mathbf{x}$ and $\mathbf{t}$ to $adom(I)$ such that $s(\mathbf{x}) = \mathbf{a}$, $s(\mathbf{t}) = \mathbf{i}$, and $I \models \varphi(\mathbf{a}, \mathbf{i}) \wedge \pi(\mathbf{i})$, but $K \not\models \exists \mathbf{y}\psi(\mathbf{a}, \mathbf{y}, \mathbf{i})$, then the chase generates a tuple $\mathbf{b}$ of distinct concrete time-stamped nulls with time-stamp $\mathbf{i}$ for the variables in $\mathbf{y}$ and adds tuples to the relations in K so that the resulting instance K' satisfies $\psi(\mathbf{a}, \mathbf{b}, \mathbf{i})$. We say that σ can be applied to $\langle I, K\rangle$ with the assignment s; we also say that the result of applying the constraint σ to $\langle I, K\rangle$ with the assignment s is $\langle I, K'\rangle$, and write $\langle I, K\rangle \xrightarrow{\sigma, s} \langle I, K'\rangle$.

- A concrete chase step over a concrete target egd for the instance $\langle I, K\rangle$: If $\forall \mathbf{x}\forall \mathbf{t}(\theta(\mathbf{x}, \mathbf{t}) \wedge \rho(\mathbf{t}) \rightarrow x_k = x_l)$ is a concrete target egd in Σ_t and if s is an assignment from the variables in $\mathbf{x}$ and $\mathbf{t}$ to $adom(K)$ such that $s(\mathbf{x}) = \mathbf{a}$ and $s(\mathbf{t}) = \mathbf{i}$, and $K \models \theta(\mathbf{a}, \mathbf{i}) \wedge \rho(\mathbf{i})$, then the

[1]This step is the same as the corresponding step in the standard chase algorithm, but we include it here for the convenience of the reader.

following cases are considered: (1) if both a_k and a_l are labeled nulls or both are concrete time-stamped nulls with the same time-stamp, then one of the two is replaced by the other throughout K; (2) if one of a_k and a_l is a constant and the other is a labeled null or a concrete time-stamped null, then the labeled null or the concrete time-stamped null is replaced by the constant throughout K; (3) if one of a_k and a_l is a labeled null and the other is a concrete time-stamped null, then the concrete time-stamped null is replaced by the labeled null throughout K; (4) if a_k and a_l are concrete time-stamped nulls with different time-stamps or if a_k and a_l are different constants, then the chase fails. If the chase fails, then we say the result of applying the constraint σ to $\langle I,K\rangle$ with the assignment s is $\perp$, and write $\langle I,K\rangle \xrightarrow{\sigma,s} \perp$; otherwise, we let K' be the instance obtained from K in cases (a), (b), or (c), and we say that the result of applying the constraint σ to $\langle I,K\rangle$ with the assignment s is $\langle I,K'\rangle$, and write $\langle I,K\rangle \xrightarrow{\sigma,s} \langle I,K'\rangle$.

For a constraint $\sigma \in \Sigma_{st} \cup \Sigma_t$, if there is a chase step and an assignment such that $\langle I,K\rangle \xrightarrow{\sigma,s} \langle I,K'\rangle$, we say that σ can be *applied* to $\langle I,K\rangle$ with the assignment s.

Definition 5 (The Concrete Chase Algorithm). Let $\mathcal{M} = (\mathbf{S},\mathbf{T},\Sigma_{st},\Sigma_t)$ be a temporal schema mapping and let I be a concrete source instance.

The concrete chase algorithm has two rounds. In the concrete s-t tgd round and starting with the instance $\langle I,\emptyset\rangle$, all concrete chase steps involving a standard s-t tgd or a concrete s-t tgd are applied sequentially and produce a concrete target instance. After this, in the concrete target egd round, all concrete target egd steps are applied so that at the end either the chase fails or a concrete target instance is produced such that no further concrete target egd step is possible.

Thus, the concrete chase algorithm amounts to a sequence $\langle I, K_j \rangle \xrightarrow{\sigma, s_j} \langle I, K_{j+1} \rangle$, $0 \leq j < m$, such that $K_0 = \emptyset$ and $K_m = \perp$ or there is no constraint σ in $\Sigma_{st} \cup \Sigma_t$ that can be applied to K_m. We say that K_m is the result of the concrete chase. In case, $K_m = \perp$, we talk about a *failing chase*, while if $K_m \neq \perp$, we talk about a *successful* case.

In the rest of this dissertation, we will use the notation c-chase$_{\Sigma_{st}}(I)$ to denote the concrete target instance produced in the concrete s-t tgd chase round on I; we will also use the notation c-chase$_{\mathcal{M}}(I)$ to denote the concrete target instance produced by the concrete chase algorithm on I w.r.t. $\mathcal{M}$.

Lemma 2. Let $\mathbf{R}$ be a temporal relational schema and let K,K' be two instances over $\mathbf{R}$. Let $\phi(\mathbf{x}, \mathbf{t}) = \varphi(\mathbf{x}, \mathbf{t}) \wedge \pi(\mathbf{t})$ be a formula, where $\varphi(\mathbf{x}, \mathbf{t})$ is a conjunction of atomic formulas over the schema $\mathbf{R}$, and $\pi(\mathbf{t})$ is a Boolean combination of Allen atomic formulas. For every homomorphism h: $K \rightarrow K'$ and every assignment s, if $K, s \models \phi(\mathbf{x}, \mathbf{t})$, then the composition of the assignment s with the homomorphism h is an assignment s' from the variables in $\mathbf{x}$ and $\mathbf{t}$ to the active domain $adom(K')$ of the instance K', such that $K', s' \models \phi(\mathbf{x}, \mathbf{t})$.

Proof. Since h is a homomorphism, the following statements are true for the composition s' of s with h:

- $s'(x) = h(s(x))$ if the value of $s(x)$ is a concrete time-stamped null or a labeled null from the instance K;
- $s'(x) = h(s(x)) = s(x)$ if $s(x)$ is a constant from the instance K;
- $s'(t) = h(s(t)) = s(t)$ and $s(t)$ is a time interval from the instance K.

The formula $\varphi(\mathbf{x},\mathbf{t})$ is a conjunction of atomic formulas over $\mathbf{R}$, while the formula $\pi(\mathbf{t})$ is a Boolean combination of Allen atomic formulas; moreover, homomorphisms preserve facts between instances. Thus, the preceding statements about s' easily imply that $K',s' \models \phi(\mathbf{x},\mathbf{t})$. $\square$

In the remainder of the paper, we adopt the notation $h \circ s$ to denote the composition of s with h.

Lemma 3. Let $\langle I,K_1\rangle \xrightarrow{\sigma,s} \langle I,K_2\rangle$ be a concrete chase step, where $K_2 \neq \bot$ and σ is a standard s-t tgd, a concrete s-t tgd, a standard target egd, or a concrete target egd. Let $\langle I,K\rangle$ be an instance such that: $\langle I,K\rangle$ satisfies σ and there exists a homomorphism h_1: $\langle I,K_1\rangle \to \langle I,K\rangle$. Then there exists a homomorphism h_2: $\langle I,K_2\rangle \to \langle I,K\rangle$.

Proof. Before we give the proof, it should be noted that both the instance $\langle I,K_1\rangle$ and the instance $\langle I,K\rangle$ may contain constants, labeled nulls, and/or concrete time-stamped nulls.

- *Case 1*: Assume that $\sigma = \forall\mathbf{x}(\varphi(\mathbf{x}) \to \exists\mathbf{y}\psi(\mathbf{x},\mathbf{y}))$, that is, σ is a standard s-t tgd (with no temporal variables). The proof is similar to that of the Lemma 3.4 in paper [32]. We know that the result of applying the constraint σ to $\langle I,K_1\rangle$ with the assignment s is $\langle I,K_2\rangle$. By definition of the chase step over a standard s-t tgd, we have that $\langle I,K_1\rangle,s \models \varphi(\mathbf{x})$. Composing an assignment and a homomorphism yields an assignment by Lemma 2. Therefore, $h_1 \circ s$ is an assignment with which $\langle I,K\rangle$ satisfies $\varphi(\mathbf{x})$, i.e., $\langle I,K\rangle,h_1 \circ s \models \varphi(\mathbf{x})$. In addition, since the combined instance $\langle I,K\rangle$ satisfies the constraint σ with every assignment, there is an assignment s' from all variables $\mathbf{x}$ and $\mathbf{y}$ occurring in the formula $\varphi(\mathbf{x}) \wedge \psi(\mathbf{x},\mathbf{y})$ to the active domain $adom(\langle I,K\rangle)$ of $\langle I,K\rangle$ (the active domain $adom(\langle I,K\rangle) = adom(I) \cup adom(K)$), such that $\langle I, K\rangle, s' \models \varphi(\mathbf{x}) \wedge \psi(\mathbf{x}, \mathbf{y})$, and the assignment s' is an extension of the assignment $h_1 \circ s$, i.e., $h_1(s(\mathbf{x})) = s'(\mathbf{x})$. For every variable y in $\mathbf{y}$, we denote by Δ'' the labeled null replacing the variable y in the chase step. Define h_2 on labeled nulls or concrete time-stamped nulls in $\langle I, K_2\rangle$ as follows: (a) if Δ is a labeled null occurring in $\langle I, K_1\rangle$, then $h_2(\Delta) = h_1(\Delta)$; (b) if Δ' is a concrete time-stamped null occurring in $\langle I, K_1\rangle$, then $h_2(\Delta') = h_1(\Delta')$; (c) and $h_2(\Delta'') = s'(y)$ for each variable y in $\mathbf{y}$.

It can be proved that h_2 is a homomorphism from $\langle I, K_2\rangle$ to $\langle I, K\rangle$. For all facts appearing in both $\langle I, K_2\rangle$ and $\langle I, K_1\rangle$, we have that h_2 maps each of them in $\langle I, K_2\rangle$ to a fact of $\langle I, K\rangle$ because the homomorphism h_2, as defined, agrees with the homomorphism h_1 on all values appearing in $\langle I, K_1\rangle$, and thus h_1 and h_2 maps every such fact into the same fact in $\langle I, K\rangle$. For other facts in $\langle I, K_2\rangle$, assume that $R(\mathbf{x}_0, \mathbf{y}_0)$ is an arbitrary atom in the formula ψ (note that $\mathbf{x}_0 \subseteq \mathbf{x}$ and $\mathbf{y}_0 \subseteq \mathbf{y}$). Then $\langle I, K_2\rangle$ contains a fact $R(s(\mathbf{x}_0), \mathbf{\Delta}''_0)$, and

$$R(h_2(s(\mathbf{x}_0)), h_2(\mathbf{\Delta}''_0)) = R(h_1(s(\mathbf{x}_0)), s'(\mathbf{y}_0)) = R(s'(\mathbf{x}_0), s'(\mathbf{y}_0)),$$

where $\mathbf{\Delta}''_0$ is a set of labeled nulls produced for $\mathbf{y}_0$ in the chase step. Since s' is an assignment mapping all atoms in the $\phi \wedge \psi$ into facts in $\langle I, K\rangle$, the fact $R(s'(\mathbf{x}_0), s'(\mathbf{y}_0))$ is contained in the combined instance $\langle I, K\rangle$. Consequently, for every fact $\langle I, K_2\rangle$ there is a corresponding fact in $\langle I, K\rangle$ under the homomorphism h_2. Therefore, we have that h_2 is a homomorphism. Furthermore, $\langle I, K_2\rangle$ satisfies the first item in the definition of a concrete solution.

- *Case 2*: Assume that $\sigma = \forall \mathbf{x}, \mathbf{t}(\varphi(\mathbf{x}, \mathbf{t}) \wedge \pi(\mathbf{t}) \rightarrow \exists \mathbf{y}\ \psi(\mathbf{x}, \mathbf{y}, \mathbf{t}))$, where the concrete s-t tgd contains at least one temporal variable. We know that the result of applying the constraint

σ to $\langle I, K_1 \rangle$ with the assignment s is $\langle I, K_2 \rangle$. By the definition of the chase step, we have that $\langle I, K_1 \rangle, s \models \varphi(\mathbf{x}, \mathbf{t}) \wedge \pi(\mathbf{t})$. Similarly, according to Lemma 2, composing the assignment s and the homomorphism h_1 yields an assignment $h_1 \circ s$ from all variables in $\{\mathbf{x}, \mathbf{t}\}$ occurring in the formula $\varphi(\mathbf{x}, \mathbf{t}) \wedge \pi(\mathbf{t})$ to the active domain $adom(\langle I, K \rangle)$ of $\langle I, K \rangle$, such that $\langle I, K \rangle, h_1 \circ s \models \varphi(\mathbf{x}, \mathbf{t}) \wedge \pi(\mathbf{t})$. In addition, since $\langle I, K \rangle$ satisfies σ with every assignment from all variables in σ to the active domain $adom(\langle I, K \rangle)$ of $\langle I, K \rangle$, there exists an assignment s' such that $\langle I, K \rangle, s' \models \varphi(\mathbf{x}, \mathbf{t}) \wedge \pi(\mathbf{t}) \wedge \psi(\mathbf{x}, \mathbf{y}, \mathbf{t})$, and the assignment s' is an extension of the composed homomorphism $h_1 \circ s$, i.e., $h_1(s(\mathbf{x})) = s'(\mathbf{x})$, $h_1(s(\mathbf{t})) = s'(\mathbf{t}) = s(\mathbf{t})$, and $s'(\mathbf{y})$ is a set of constants or labeled nulls or concrete time-stamped nulls with the time-stamp $s'(\mathbf{t})$ by the first item in the definition of *concrete solutions*. For each variable y, the chase step replaces it by a fresh concrete time-stamped null $\Delta^{s(\mathbf{t})}$ with the time-stamp $s(\mathbf{t})$. Hence, if the fact contains a null, then it is a concrete time-stamped null, and its time-stamp must be the tuple $s(\mathbf{t})$ of time intervals, because, by the definition of the concrete chase step over a concrete s-t tgd, all the concrete time-stamped nulls produced in this chase step have the same time-stamp $s(\mathbf{t})$. We adopt the symbol $\Delta^{s(\mathbf{t})}$ to represent a concrete time-stamped null in $\langle I, K_2 \rangle$ which replaces y in the chase step. Define h_2 on labeled nulls or concrete time-stamped nulls in $\langle I, K_2 \rangle$ as follows:

- $h_2(\Delta^{s(\mathbf{t})}) = h_2(\Delta^{s'(\mathbf{t})}) = s'(y)$ for each y in $\mathbf{y}$;
- if Δ' is a concrete time-stamped null occurring in $\langle I, K_1 \rangle$, then $h_2(\Delta') = h_1(\Delta')$;
- if Δ is a labeled null occurring in $\langle I, K_1 \rangle$, then $h_2(\Delta) = h_1(\Delta)$.

It should be noted that (a) $s'(y)$ is a labeled null or a concrete time-stamped null with the time-stamp $s'(\mathbf{t})$ or a constant; (b) and h_2 preserves constants and time intervals, i.e., $h_2(a) = a = h_1(a)$ if a is a constant and $h_2(i) = i = h_1(i)$ if i is a time interval.

We need to prove that h_2 is a homomorphism which maps each fact of $\langle I, K_2\rangle$ into a fact of $\langle I, K\rangle$. As in Case 1, for those facts in both $\langle I, K_2\rangle$ and $\langle I, K_1\rangle$, this is true. For those new facts produced by the chase step, we firstly assume an arbitrary atom $R(\mathbf{x}_0, \mathbf{y}_0, t_0)$ in $\psi(\mathbf{x}, \mathbf{y}, \mathbf{t})$ (note that $\mathbf{x}_0 \subseteq \mathbf{x}$, $\mathbf{y}_0 \subseteq \mathbf{y}$, and $t_0 \in \mathbf{t}$). Then $\langle I, K_2\rangle$ contains the fact $R(s(\mathbf{x}_0), \mathbf{\Delta}_0^{s(\mathbf{t})}, s(t_0))$ besides all those facts in $\langle I, K_1\rangle$. Thus, we have

$$\begin{aligned} R(h_2(s(\mathbf{x}_0)), h_2(\mathbf{\Delta}_0^{s(\mathbf{t})}), h_2(s(t_0))) &= R(h_1(s(\mathbf{x}_0)), s'(\mathbf{y}_0), h_1(s(t_0))) \\ &= R(s'(\mathbf{x}_0), s'(\mathbf{y}_0), s'(t_0)). \end{aligned}$$

Since s' is an assignment that

$$\langle I, K\rangle, s' \models \varphi(\mathbf{x}, t) \wedge \psi(\mathbf{x}, \mathbf{y}, t),$$

there exists a fact $R(s'(\mathbf{x}_0), s'(\mathbf{y}_0), s'(t_0))$ in $\langle I, K\rangle$. Therefore, for every fact of $\langle I, K_2\rangle$, there is a corresponding fact in $\langle I, K\rangle$, and hence h_2 is a homomorphism. Furthermore, $\langle I, K_2\rangle$ satisfies the second item of the definition of *concrete solutions*.

- *Case 3*: The constraint σ is a target egd of the form $\forall\mathbf{x}\forall\mathbf{t}(\theta(\mathbf{x}, \mathbf{t}) \wedge \rho(\mathbf{t}) \to x_k = x_l)$, where the constraint may contain zero or more temporal variables. We know that the result of applying the constraint σ to $\langle I, K_1\rangle$ with the assignment s is $\langle I, K_2\rangle$. By the definition of the chase step, we have that $K_1, s \models \theta(\mathbf{x}, \mathbf{t}) \wedge \rho(\mathbf{t})$ (or $\langle I, K_1\rangle, s \models \theta(\mathbf{x}, \mathbf{t}) \wedge \rho(\mathbf{t})$). Composing the assignment s and the homomorphism h_1 yields an assignment s', i.e., $h_1 \circ s$, such that $\langle I, K\rangle, h_1 \circ s \models \theta(\mathbf{x}, \mathbf{t}) \wedge \rho(\mathbf{t})$. We define the homomorphism h_2 on labeled nulls occurring in the instance $\langle I, K_2 \rangle$ as follows: if Δ is a labeled null or a concrete time-stamped null occurring in both $\langle I, K_1 \rangle$ and $\langle I, K_2 \rangle$, then $h_2(\Delta) = h_1(\Delta)$. Since this chase step only collapses nulls in $\langle I, K_1 \rangle$, all values occurring in the instance $\langle I, K_2 \rangle$ also occur in the instance $\langle I, K_1 \rangle$. Therefore, the homomorphism h_2 agrees with the homomorphism h_1 on all the values appearing in the instance $\langle I, K_2 \rangle$. We need to ensure that h_2 is still a homomorphism from $\langle I, K_2 \rangle$ to $\langle I, K \rangle$. The only way that h_2 fails to be a homomorphism on $\langle I, K_2 \rangle$ is if h_1 maps $s(x_k)$ and $s(x_l)$ into two different values (either of them could be a constant, a labeled null, or a concrete time-stamped null). But this is not the case because if h_1 maps $s(x_k)$ and $s(x_l)$ to two different constants or labeled nulls or concrete time-stamped nulls, then this can not be true because the combined instance $\langle I, K \rangle$ satisfies σ and $\langle I, K \rangle, h_1 \circ s \models \theta(\mathbf{x}, \mathbf{t}) \wedge \rho(\mathbf{t})$, thus, $h_1(s(x_k)) = h_1(s(x_l))$.

□

Theorem 4. Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st}, \Sigma_t)$ be a fixed concrete schema mapping. For every concrete source instance I, if the concrete chase algorithm does not fail on I, then the concrete target instance c-chase$_{\mathcal{M}}(I)$ returned by this algorithm is a concrete universal solution for I w.r.t. $\mathcal{M}$. Furthermore, the running time of the concrete chase algorithm is bounded by a polynomial in the size of I.

Proof. Let $J =$ c-chase$_{\mathcal{M}}(I)$ be the result of the concrete chase on I w.r.t. $\mathcal{M}$. Assume that J' is an arbitrary solution for I w.r.t. $\mathcal{M}$. Thus, J' satisfies $\Sigma_{st} \cup \Sigma_t$. Moreover, the identity mapping: from $\langle I, \emptyset \rangle$ to $\langle I, J' \rangle$ is a homomorphism. By applying Lemma 3 at each chase step, we obtain a homomorphism h: from $\langle I, J \rangle$ to $\langle I, J' \rangle$. Thus J is a universal solution for I w.r.t. $\mathcal{M}$.

We now examine the data complexity of the concrete chase algorithm. Since the concrete schema mapping $\mathcal{M}$ is fixed, the number of concrete s-t tgd chase steps is bounded by a polynomial in the size of the given concrete source instance I (the degree of the polynomial depends, of course, on the arities of the relations in the source and target schemas and the size of the concrete s-t tgds in $\mathcal{M}$). Similarly, the number of concrete target egd steps is bounded by a polynomial in the size of the target instance c-chase$_{\Sigma_{st}}(I)$ produced at the end of the concrete s-t tgd round. Thus, the running time of the concrete chase algorithm on I w.r.t. $\mathcal{M}$ is bounded by a polynomial in the size of I. $\square$

Remark 1. Note that, during a run of the concrete chase algorithm, it is possible that different constraints and/or different assignments may be applicable to the pair $\langle I, K\rangle$, where K is the current concrete target instance. Therefore, unless we have some default rule for selecting which constraint and assignment to apply, we may have different chase sequences that may yield different target instances as results. It is easy to see, however, that these different target instances are unique up to *homomorphic equivalence*, that is, if J and J' are two such target instances, then there is a homomorphism from J to J', and also a homomorphism from J' to J. Indeed, according to Theorem 4, both J and J' are universal solutions for I. Therefore, by definition, there will be a homomorphism from the universal solution J to the solution J', and there will be a homomorphism from the universal solution J' to the solution J.

The next example illustrates the preceding Theorem 4.

Example 4. Consider the same temporal source schema and the same temporal target schema as in Example 2. Specifically, let **S** be a source schema consisting of the relation symbols:

$$\mathsf{Reg}(title, conference, author), \mathsf{uSub}(title, conference, Stime),$$

$$\mathsf{uRew}(title, conference, Rtime), \mathsf{TbPub}(title, conference, year, Ptime);$$

and let **T** be a target schema consisting of the relation symbols:

$$\mathsf{Pub}(title, conference, author, year, page, Atime)$$

and

$$\mathsf{InPrcs}(title, conference, author, editor).$$

Let $\mathcal{M}^* = (\mathbf{S}, \mathbf{T}, \Sigma^*_{st}, \Sigma^*_t)$ be the schema mapping in which Σ^*_{st} consists of the constraints

$$\forall x_1, x_2, x_3, x_4, t_1, t_2 (\mathsf{uRew}(x_1, x_2, t_1) \wedge \mathsf{TbPub}(x_1, x_2, x_3, t_2) \wedge \mathsf{Reg}(x_1, x_2, x_4) \wedge (t_1 \mathsf{m} t_2)$$

$$\rightarrow \exists y_1 \; \mathsf{Pub}(x_1, x_2, x_4, x_3, y_1, t_2))$$

$$\forall x_1, x_2, x_3, t_1 (\mathsf{uSub}(x_1, x_2, t_1) \wedge \mathsf{Reg}(x_1, x_2, x_3) \rightarrow \exists y_1 \; \mathsf{InPrcs}(x_1, x_2, x_3, y_1))$$

$$\forall x_1, x_2, x_3, t_1 (\mathsf{uRew}(x_1, x_2, t_1) \wedge \mathsf{Reg}(x_1, x_2, x_3) \rightarrow \exists y_1 \; \mathsf{InPrcs}(x_1, x_2, x_3, y_1)),$$

and Σ^*_t consists of the constraints

$$\forall x_1, x_2, x_3, x_4, x_5, x_6, t_1, t_2 (\mathsf{Pub}(x_1, x_2, x_3, x_4, x_5, t_1) \wedge \mathsf{Pub}(x_1, x_2, x_3, x_4, x_6, t_2) \wedge (t_1 {=} t_2)$$

$$\rightarrow x_5 = x_6)$$

$$\forall x_1, x_2, x_3, x_4, x_5 (\mathsf{InPrcs}(x_1, x_2, x_3, x_4) \wedge \mathsf{InPrcs}(x_1, x_2, x_3, x_5) \rightarrow x_4 = x_5).$$

Let I^* be the concrete source instance whose relations are depicted in Table 3.1.

Let J_1 denote the universal solution produced by the concrete chase algorithm on I^* w.r.t. $\mathcal{M}^*$, where the concrete chase algorithm applies the concrete s-t tgds and the target egds

Table 3.1: The relations Reg, uSub, uRew, and TbPub in the concrete source instance I^*.

(a) Reg

title	conference	author
a_1	b_1	c_1
a_2	b_1	c_2
a_3	b_1	c_1
a_4	b_2	c_3
a_5	b_2	c_4

(b) uSub

title	conference	Stime
a_1	b_1	$[1,3)$
a_2	b_1	$[2,4)$
a_3	b_1	$[1,3)$
a_4	b_2	$[5,6)$
a_5	b_2	$[6,7)$

(c) uRew

title	conference	Rtime
a_1	b_1	$[3,5)$
a_2	b_1	$[4,5)$
a_3	b_1	$[3,4)$
a_4	b_2	$[6,8)$

(d) TbPub

title	conference	year	Ptime
a_1	b_1	1	$[5,6)$
a_2	b_1	2	$[5,7)$
a_4	b_2	1	$[8,10)$

in the order they are listed; its relations are depicted in Table 3.2. Let J_2 denote the universal solution produced by the concrete chase algorithm on I^* w.r.t. $\mathcal{M}^*$, where the concrete chase algorithm applies the first concrete s-t tgd, the third concrete s-t tgd, and the second concrete s-t tgd; after that the concrete chase applies the target egds in the order they are listed; its relations are depicted in Table 3.3. It is easy to verify that the universal solution J_1 is homomophically equivalent to the universal solution J_2.

Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st}, \Sigma_t)$ be an arbitrary temporal schema mapping and let I be a concrete source instance. According to Theorem 4, if the concrete chase algorithm succeeds on I, then

Table 3.2: The relations Pub and InPrcs of the universal solution J_1 by the concrete chase algorithm.

(a) Pub

title	conference	author	year	page	Atime
a_1	b_1	c_1	1	$N_1^{[3,5),[5,6)}$	$[5,6)$
a_2	b_1	c_2	2	$N_2^{[4,5),[5,7)}$	$[5,7)$
a_4	b_2	c_3	1	$N_3^{[6,8),[8,10)}$	$[8,10)$

(b) InPrcs

title	journal	author	editor
a_1	b_1	c_1	N_1
a_2	b_1	c_2	N_2
a_3	b_1	c_1	N_3
a_4	b_2	c_3	N_4
a_5	b_2	c_4	N_5

Table 3.3: The relations Pub and InPrcs of the universal solution J_2 produced by the concrete chase algorithm.

(a) Pub

title	conference	author	year	page	Atime
a_1	b_1	c_1	1	$N_1^{[3,5),[5,6)}$	$[5,6)$
a_2	b_1	c_2	2	$N_2^{[4,5),[5,7)}$	$[5,7)$
a_4	b_2	c_3	1	$N_3^{[6,8),[8,10)}$	$[8,10)$

(b) InPrcs

title	journal	author	editor
a_1	b_1	c_1	N_1
a_2	b_1	c_2	N_2
a_3	b_1	c_1	N_3
a_4	b_2	c_3	N_4
a_5	b_2	c_4	N_9

it produces a concrete universal solution for I w.r.t. $\mathcal{M}$. If the concrete chase algorithm fails, then Theorem 4 provides no information concerning the existence or non-existence of a concrete universal solution for I w.r.t. $\mathcal{M}$. The next result shows that there are concrete schema mappings and concrete source instances such that the concrete chase algorithm fails, yet these instances have universal solutions.

Theorem 5. There are three concrete schema mappings $\mathcal{M}_1$, $\mathcal{M}_2$, and $\mathcal{M}_3$ with the following properties.

1. $\mathcal{M}_1 = (\mathbf{S}_1, \mathbf{T}_1, \Sigma^1_{st}, \Sigma^1_t)$, where Σ^1_{st} consists of two concrete s-t tgds each of which contains a single existentially quantified variable and a single temporal variable occurring in every atom of the consequent; moreover, Σ^1_t consists of one concrete target egd that contains two temporal variables.

2. $\mathcal{M}_2 = (\mathbf{S}_2, \mathbf{T}_2, \Sigma^2_{st}, \Sigma^2_t)$, where Σ^2_{st} consists of two concrete s-t tgds, one of which contains a single existentially quantified variable and a single temporal variable, while the other contains a single existentially quantified variable and two temporal variables; moreover, Σ^2_t consists of one concrete target egd that contains a single temporal variable.

3. $\mathcal{M}_3 = (\mathbf{S}_3, \mathbf{T}_3, \Sigma^3_{st}, \Sigma^3_t)$, where Σ^3_{st} consists of two concrete s-t tgds, one of which contains two existentially quantified variables and a single temporal variable that does not occur in the consequent, while the other contains an existentially quantified variable and a single temporal variable that occur in only one of the two atoms of the consequent; moreover, Σ^3_t consists of one standard target egd (i.e., it has no temporal variables).

4. There are three concrete source instances I_1, I_2, and I_3, such that for each $j = 1, 2, 3$, the instance I_j has a universal solution w.r.t. $\mathcal{M}_j$, but the concrete chase algorithm fails on I_j w.r.t. $\mathcal{M}_j$.

Proof. The proof has three parts, one for each concrete schema mapping as in the statement of the theorem.

Part 1: Let $\mathcal{M}_1 = (\mathbf{S}_1, \mathbf{T}_1, \Sigma^1_{st}, \Sigma^1_t)$ be the schema mapping where Σ^1_{st} consists of the constraints

$$\sigma^1_{st} = \forall n,l,t(E(n,t) \wedge S(n,l,t) \rightarrow \exists c\, Emp(n,c,l,t))$$

$$\sigma^2_{st} = \forall n,p,t(P(n,p,t) \rightarrow \exists c\, EmpPos(n,c,p,t))$$

and Σ^1_t consists of the constraint

$$\sigma_t = \forall n,c_1,c_2,l,p,t_1,t_2(Emp(n,c_1,l,t_1) \wedge EmpPos(n,c_2,p,t_2) \rightarrow c_1 = c_2).$$

Let I_1 be the concrete source instance whose relations are depicted in Table 3.4.

Table 3.4: The relations E, S, and P of the concrete source instance I_1.

(a) E

Name	Time
Ada	$[2013,2018)$
Bob	$[2012,2015)$

(b) S

Name	Salary	Time
Ada	18000	$[2013,2018)$
Bob	13000	$[2012,2015)$

(c) P

Name	Position	Time
Ada	Manager	$[2015,2017)$
Bob	Consultant	$[2012,2015)$

Table 3.5: The relations Emp and $EmpPos$ of the concrete target instance c-chase$_{\Sigma^1_{st}}(I_1)$.

(a) Emp

Name	Company	Salary	Time
Ada	$N_1^{[2013,2018)}$	18000	$[2013,2018)$
Bob	$N_2^{[2012,2015)}$	13000	$[2012,2015)$

(b) $EmpPos$

Name	Company	Position	Time
Ada	$N_3^{[2015,2017)}$	Manager	$[2015,2017)$
Bob	$N_4^{[2012,2015)}$	Consultant	$[2012,2015)$

Let c-chase$_{\Sigma^1_{st}}(I_1)$ be the concrete target instance produced by the concrete chase algorithm on I_1 with Σ^1_{st}. During the concrete target egd chase round with Σ^1_t, the concrete chase algorithm fails because there is an assignment s, where

$$s(n) = Ada, s(c_1) = N_1^{[2013,2018)},\ s(l) = 18000, s(t_1) = [2013,2018), s(c_2) = N_3^{[2015,2017)},$$

$$s(p) = Manager,\ \ s(t_2) = [2015,2017),$$

and

$$\begin{aligned} s(Emp(n,c_1,l,t_1)) &= Emp(s(n),s(c_1),s(l),s(t_1)) \\ &= Emp(Ada, N_1^{[2013,2018)}, 18000, [2013,2018)), \\ s(EmpPos(n,c_2,p,t_2)) &= Emp(s(n),s(c_2),s(p),s(t_2)) \\ &= EmpPos(Ada, N_3^{[2015,2017)}, Manager, [2015,2017)), \end{aligned}$$

such that

$$\langle I_1, \text{c-chase}_{\Sigma^1_{st}}(I_1)\rangle, s \models Emp(n,c_1,l,t_1) \wedge EmpPos(n,c_2,p,t_2),$$

but $s(c_1)$ and $s(c_2)$ are two concrete time-stamped nulls with different time-stamps.

Although the concrete chase algorithm fails on I_1 w.r.t. $\mathcal{M}_1$, we claim that there exists a universal solution J_1 for I_1 w.r.t. $\mathcal{M}_1$; the relations of J_1 are depicted in Table 3.6.

Table 3.6: The relations *Emp* and *EmpPos* of the concrete universal solution J_1 for I_1 w.r.t. $\mathcal{M}_1$.

(a) *Emp*

Name	Company	Salary	Time
Ada	N_1	18000	$[2013,2018)$
Bob	$N_2^{[2012,2015)}$	13000	$[2012,2015)$

(b) *EmpPos*

Name	Company	Position	Time
Ada	N_1	Manager	$[2015,2017)$
Bob	$N_2^{[2012,2015)}$	Consultant	$[2012,2015)$

To verify that J_1 is indeed a universal solution for I_1 w.r.t. $\mathcal{M}_1$, we have to verify that the following two statements are true: (1) $\langle I_1, J_1 \rangle \models \Sigma^1_{st}$ and $J_1 \models \Sigma^1_t$, hence J_1 is a solution for I_1 w.r.t. $\mathcal{M}_1$; (2) if K is a solution for I w.r.t. $\mathcal{M}$, then there is a homomorphism from J_1 to K.

The first statement can be easily verified by inspecting the relations of J_1. To verify the second statement, let K be some arbitrary, but fixed, solution for I_1 w.r.t. $\mathcal{M}_1$. This means that $\langle I_1, K \rangle$ satisfies the two concrete s-t tgds in Σ^1_{st}, and K satisfies the concrete target egd in Σ^1_t. We have to show that there is a homomorphism h from J_1 to K. The active domain $adom(J_1)$ of J_1 consists of constants, time intervals, the labeled null N_1, and the concrete time-stamped null $N_2^{[2012,2015)}$. Since homomorphisms map constants and time intervals to themselves, it is enough to define the values $h(N_1)$ and $h(N_2^{[2012,2015)})$. Consider the concrete s-t tgd σ^1_{st} in Σ^1_{st} and let s_1 be the assignment from the variables n, l, t to the active domain $adom(I_1)$ of I_1 such that

$$s_1(n) = Ada,\ s_1(l) = 18000,\ s_1(t) = [2013, 2018).$$

Since $\langle I_1, K \rangle \models \sigma^1_{st}$ and $I \models E(s_1(n), s_1(t)) \wedge S(s_1(n), s_1(l), s_1(t))$, there must exist an element Δ_1 in the active domain $adom(K)$ of K such that $K \models Emp(s_1(n), \Delta_1, s_1(l), s_1(t))$, i.e., $K \models Emp(Ada, \Delta_1, 18000, [2013, 2018))$. Next, let s_2 be the assignment from the variables n, l, t to the active domain $adom(I_1)$ of I_1 such that

$$s_2(n) = Bob,\ s_2(l) = 13000,\ s_2(t) = [2012, 2015).$$

Since $\langle I_1, K \rangle \models \sigma^1_{st}$ and $I \models E(s_2(n), s_2(t)) \wedge S(s_2(n), s_2(l), s_2(t))$, there must exist an element Δ_2 in the active domain $adom(K)$ of K such that $K \models Emp(s_2(n), \Delta_2, s_2(l), s_2(t))$, i.e., $K \models Emp(Bob, \Delta_2, 13000, [2012, 2015))$.

Let h be the function from the active domain $adom(J_1)$ of J_1 to the active domain

$adom(K)$ of K such that $h(N_1) = \Delta_1$; $h(N_2^{[2012,2015)}) = \Delta_2$; and h is the identity on the constants and time intervals in $adom(J_1)$. We claim that h is a homomorphism from J_1 to K. For this, we have to show that h maps every fact of J_1 to some fact of K. Observe that J_1 has four facts, two in the relation Emp^{J_1} and two in the relation $EmpPos^{J_1}$. By the properties of the elements Δ_1 and Δ_2, it is obvious that h maps the two facts of the relation Emp^{J_1} to two facts of the relation Emp^K. It remains to show that h maps the two facts of the relation $EmpPos^{J_1}$ to two facts of the relation $EmpPos^K$. Since the two facts of $EmpPos^{J_1}$ are the tuples $(Ada, N_1, Manager, [2015, 2017))$ and $(Bob, N_2^{[2012,2015)}, Consultant, [2012, 2015))$, we have to show that the tuples $(Ada, \Delta_1, Manager, [2015, 2017))$ and $(Bob, \Delta_2, Consultant, [2012, 2015))$ belong to the relation $EmpPos^K$. Consider the s-t tgd σ^2_{st} in Σ^1_{st} and let s_3 be the assignment from the variables n, p, t to the active domain $adom(I_1)$ of I_1 such that

$$s_3(n) = Ada,\ s_3(p) = Manager,\ s_3(t) = [2015, 2017).$$

Since $\langle I_1, K\rangle \models \sigma^2_{st}$ and $I \models P(s_3(n), s_3(p), s_3(t))$, there must exist an element Δ_3 in the active domain $adom(K)$ of K such that $K \models EmpPos(s_3(n), \Delta_3, s_3(p), s_3(t))$, i.e.,

$$K \models EmpPos(Ada, \Delta_3, Manager, [2015, 2017))$$

. Consider now the concrete target egd σ_t in Σ_t. Since $K \models \sigma_t$ and also

$$K \models Emp(Ada, \Delta_1, 18000, [2013, 2018)) \wedge EmpPos(Ada, \Delta_3, Manager, [2015, 2017)),$$

we must have that $\Delta_1 = \Delta_3$. Therefore, the tuple $(Ada, \Delta_1, Manager, [2015, 2017))$ belongs to the relation $EmpPos^K$. By repeating the same argument with the assignment

$$s_4(n) = Bob,\ s_4(p) = Consultant,\ s_4(t) = [2012, 2015),$$

we obtain that the tuple $(Bob, \Delta_2, Consultant, [2012, 2015))$ belongs to the relation $EmpPos^K$. This completes the proof that the function h is a homomorphism from J_1 to K, and also completes the proof of Part I.

Part 2: Let $\mathcal{M}_2 = (\mathbf{S}_2, \mathbf{T}_2, \Sigma^2_{st}, \Sigma^2_t)$ be a concrete schema mapping where Σ^2_{st} consists of the concrete s-t tgds

$$\sigma^1_{st} = \forall x_1, x_2, t_1 (R_1(x_1, x_2, t_1) \rightarrow \exists y\, T_1(x_1, y, t_1))$$

$$\sigma^2_{st} = \forall x_1, x_2, x_3, t_1, t_2 (R_2(x_1, x_2, t_1) \wedge R_3(x_1, x_3, t_2) \wedge (t_2 \mathsf{\ m\ } t_1) \rightarrow \exists y\, T_2(x_1, y, t_2))$$

and Σ^2_t consists of the concrete target egd

$$\sigma_t = \forall x_1, x_2, x_3, t_1 (T_1(x_1, x_2, t_1) \wedge T_2(x_1, x_3, t_1) \rightarrow x_2 = x_3).$$

Let I_2 be the concrete source instance whose relations are depicted in Table 3.7.

Let c-chase$_{\Sigma^2_{st}}(I_2)$ be the concrete target instances produced by the concrete chase algorithm on I_2 with Σ^2_{st}. The relations of c-chase$_{\Sigma^2_{st}}(I_2)$ are depicted in Table 3.8.

During the concrete target egd chase round with Σ^2_t, the concrete chase algorithm fails on I_2 because there is an assignment s, where

$$s(x_1) = a_1, s(x_2) = N_2^{[1,4)}, s(t_1) = [1,4), s(x_3) = N_3^{[4,6),[1,4)},$$

and

$$s(T_1(x_1, x_2, t_1)) = T_1(s(x_1), s(x_2), s(t_1)) = T_1(a_1, N_2^{[1,4)}, [1,4)),$$

$$s(T_2(x_1, x_3, t_1)) = T_2(s(x_1), s(x_3), s(t_1)) = T_2(a_1, N_3^{[4,6),[1,4)}, [1,4)),$$

such that

$$\langle I_2, \text{c-chase}_{\Sigma^2_{st}}(I_2)\rangle, s \models T_1(x_1, x_2, t_1) \wedge T_2(x_1, x_3, t_1),$$

but $s(x_2)$ and $s(x_3)$ are two concrete time-stamped nulls with different time-stamps. Although the concrete chase algorithm fails on I_2 w.r.t. $\mathcal{M}_2$, we claim that there exists a universal solution for I_2 w.r.t. $\mathcal{M}_2$; the relations of J_2 are depicted in Table 3.9. To verify that J_2 is indeed a universal solution for I_2 w.r.t. $\mathcal{M}_2$, we have to verify the following two statements: (1) $\langle I_2, J_2 \rangle \models \Sigma_{st}^2$ and $J_2 \models \Sigma_t^2$, hence J_2 is a solution for I_2 w.r.t. $\mathcal{M}_2$; (2) if K is a solution for I_2 w.r.t. $\mathcal{M}_2$, then there is a homomorphism from J_2 to K. The first statement can be easily verified by inspecting the relations of J_2. The second statement can be proved using an argument analogous to the one used in Part 1 to show that J_1 is a universal solution for I_1 w.r.t. $\mathcal{M}_1$. The details are left to the reader. This completes the proof of Part 2.

Table 3.7: The relations R_1, R_2, and R_3 in the concrete source instance I_2.

(a) R_1

name	position	Ptime
a_1	d_1	$[1,3)$
a_1	d_2	$[1,4)$

(b) R_2

name	address	Stime
a_1	b_1	$[4,6)$

(c) R_3

name	city	Ctime
a_1	e_1	$[1,4)$

Table 3.8: The relations T_1 and T_2 in the target instance c-chase$_{\Sigma_{st}^2}(I_2)$.

(a) T_1

name	school	Ptime
a_1	$N_1^{[1,3)}$	$[1,3)$
a_1	$N_2^{[1,4)}$	$[1,4)$

(b) T_2

name	school	Ctime
a_1	$N_3^{[4,6),[1,4)}$	$[1,4)$

Part 3: Let $\mathcal{M}_3 = (\mathbf{S}_3, \mathbf{T}_3, \Sigma_{st}^3, \Sigma_t^3)$ be a concrete schema mapping where the source schema $\mathbf{S}_3$ is

Table 3.9: The relations T_1 and T_2 of the concrete universal solution J_2 for I_2 w.r.t. $\mathcal{M}_2$.

(a) T_1

name	school	Ptime
a_1	$N_1^{[1,3)}$	$[1,3)$
a_1	N_2	$[1,4)$

(b) T_2

name	school	Ctime
a_1	N_2	$[1,4)$

the schema $\mathbf{S}_2$ from Part 2; the set Σ_{st}^3 consists of the concrete s-t tgds

$$\sigma_{st}^1 = \forall x_1, x_2, t (R_2(x_1, x_2, t) \to \exists y_1, y_2\ T_2(x_1, y_1, y_2))$$

$$\sigma_{st}^2 = \forall x_1, x_2, x_3, t\ (R_1(x_1, x_2, t) \wedge R_3(x_1, x_3, t) \to \exists y\ T_1(x_1, y, t) \wedge T_2(x_1, x_3, y))$$

and Σ_t^3 consists of the concrete target egds

$$\sigma_t^1 = \forall x_1, x_2, x_3, x_4, x_5\ (T_2(x_1, x_2, x_3) \wedge T_2(x_1, x_4, x_5) \to x_2 = x_4)$$

$$\sigma_t^2 = \forall x_1, x_2, x_3, x_4, x_5\ (T_2(x_1, x_2, x_3) \wedge T_2(x_1, x_4, x_5) \to x_3 = x_5).$$

Let I_3 be the concrete source instance I_2 (as in the previous part), whose relations are depicted in Table 3.7.

Let c-chase$_{\Sigma_{st}^3}(I_3)$ be the concrete target instances produced by the concrete chase algorithm on I_3 with Σ_{st}^3 in $\mathcal{M}_3$. The relations of c-chase$_{\Sigma_{st}^3}(I_3)$ are depicted in Table 3.10.

During the concrete target egd chase round with Σ_t^3, the concrete chase algorithm fails on I_3 because there is an assignment s, where

$$s(x_1) = a_1, s(x_2) = e_1, s(x_3) = N_1^{[1,4)}, s(x_4) = N_2^{[4,6)}, s(x_5) = N_3^{[4,6)},$$

and

$$s(T_2(x_1,x_2,x_3)) = T_2(s(x_1),s(x_2),s(x_3)) = T_2(a_1,e_1,N_1^{[1,4)}),$$

$$s(T_2(x_1,x_4,x_5)) = T_2(s(x_1),s(x_4),s(x_5)) = T_2(a_1,N_2^{[4,6)},N_3^{[4,6)}),$$

such that

$$\langle I_3, \text{c-chase}_{\Sigma_{st}^3}(I_3)\rangle, s \models T_2(x_1,x_2,x_3) \wedge T_2(x_1,x_4,x_5),$$

but $s(x_3)$ and $s(x_5)$ are two concrete time-stamped nulls with different time-stamps. Although the concrete chase algorithm fails on I_3 w.r.t. $\mathcal{M}_3$, we claim that there exists a universal solution for I_3 w.r.t. $\mathcal{M}_3$; the relations of J_3 are depicted in Table 3.11. To verify that J_3 is indeed a universal solution for I_3 w.r.t. $\mathcal{M}_3$, we have to verify the following two statements: (1) $\langle I_3, J_3\rangle \models \Sigma_{st}^3$ and $J_3 \models \Sigma_t^3$, hence J_3 is a solution for I_3 w.r.t. $\mathcal{M}_3$; (2) if K is a solution for I_3 w.r.t. $\mathcal{M}_3$, then there is a homomorphism from J_3 to K. The first statement can be easily verified by inspecting the relations of J_3. The second statement can be proved using an argument analogous to the one used in Part 1 to show that J_1 is a universal solution for I_1 w.r.t. $\mathcal{M}_1$. The details are left to the reader. This completes the proof of Part 3 and also the proof of the theorem.

Table 3.10: The relations T_1 and T_2 in the target instance c-chase$_{\Sigma_{st}^3}(I_3)$.

(a) T_1

name	school	Ptime
a_1	$N_1^{[1,4)}$	$[1,4)$

(b) T_2

name	city	school
a_1	e_1	$N_1^{[1,4)}$
a_1	$N_2^{[4,6)}$	$N_3^{[4,6)}$

□

Table 3.11: The relations T_1 and T_2 of the concrete universal solution J_3 for I_3 w.r.t. $\mathcal{M}_3$.

(a) T_1

name	school	Ptime
a_1	N_1	$[1,4)$

(b) T_2

name	city	school
a_1	e_1	N_1

Remark 2. Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st}, \Sigma_t)$ be an arbitrary temporal schema mapping and let I be a concrete source instance. Theorem 4 tells that if the concrete chase algorithm succeeds on I, then the concrete target instance c-chase$_{\mathcal{M}}(I)$ returned by this algorithm is a concrete universal solution for I w.r.t. $\mathcal{M}$. However, Theorem 5 shows that the concrete chase algorithm may fail on I, yet a concrete universal solution for I w.r.t. $\mathcal{M}$ may exist. This state of affairs raises the problem to identify classes of schema mappings for which the concrete chase algorithm is sound and complete, that is, given a concrete source instance I, if the concrete chase algorithm succeeds, then a universal solution for I exists, while if the concrete chase algorithm fails, then no solution for I exists. Clearly, such classes should not contain schema mappings of the kind encountered in Theorem 5.

Theorem 6. Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st}, \Sigma_t)$ be a concrete schema mapping, such that one of the following two conditions holds: (a) Every concrete s-t tgd in Σ_{st} is full (i.e., its consequent contains no existential quantifiers); (b) If a concrete s-t tgd in Σ_{st} is not full, then it contains exactly one temporal variable, and this temporal variable occurs in every atom of the consequent of the concrete s-t tgd; moreover, every target egd in Σ_t contains at most one temporal variable. If I is a concrete source instance, then the following statements hold:

1. If the concrete chase algorithm succeeds on I, then the concrete target instance c-chase$_{\mathcal{M}}(I)$

returned by this algorithm is a concrete universal solution for I w.r.t. $\mathcal{M}$.

2. If the concrete chase algorithm fails on I, there is no solution for I w.r.t. $\mathcal{M}$.

Moreover, the running time of the concrete chase algorithm is bounded by a polynomial in the size of I.

Before embarking on the proof of Theorem 6, we state and prove an auxiliary result.

Lemma 7. Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st}, \Sigma_t)$ be a concrete schema mapping such that if a concrete s-t tgd in Σ_{st} is not full, then it contains exactly one temporal variable, and this temporal variable occurs in every atom of the consequent of the concrete s-t tgd. Let I be a concrete source instance over $\mathbf{S}$ and let $K \neq \perp$ be a target instance produced at some step of the concrete chase algorithm for I w.r.t. $\mathcal{M}$. If a concrete time-stamped null N^i occurs in a fact of K, then the time interval i also occurs in that fact; moreover, the time-stamp of every concrete time-stamped null occurring in that fact is i.

Proof. We first consider the concrete s-t tgd round and show, by induction, that every concrete target instance K_j produced in a step of the concrete s-t tgd round has the desired properties (note that $K_j \neq \perp$, for every such instance K_j). To begin with, this holds trivially true for the initial instance $K_0 = \emptyset$. Let K_j be the current concrete target instance, and let $\langle I, K_j \rangle \xrightarrow{\sigma, s} \langle I, K_{j+1} \rangle$ be a chase step in the concrete s-t tgd round. By induction hypothesis, assume that K_j has the desired properties. We have to show that K_{j+1} also has the desired properties. Since K_{j+1} is obtained from K_j by adding one or more facts, it suffices to show that the facts added to K_j have the desired properties. Assume that σ is the concrete s-t tgd $\forall \mathbf{x}, t(\varphi(\mathbf{x}, t) \rightarrow \exists \mathbf{y} \psi(\mathbf{x}, \mathbf{y}, t))$. By the hypothesis of the lemma, σ contains exactly one temporal variable t, and this temporal variable t occurs in every atom of the consequent $\psi(\mathbf{x},\mathbf{y},t)$. The chase step involving the assignment s introduces one or more fresh concrete time-stamped nulls all of which have $s(t)$ as their time-stamp, because the only temporal variable occurring in σ is t. Moreover, since t occurs in every atom in the consequent $\psi(\mathbf{x},\mathbf{y},t)$, the time interval must occur in every new fact created. Thus, K_{j+1} has the desired properties.

We now consider the concrete target egd round. Assume that $\langle I,K_j\rangle \xrightarrow{\sigma,s} \langle I,K_{j+1}\rangle$ is a step in the concrete target egd round and that K_j and K_{j+1} are concrete target instances such that K_j has the desired properties and $K_{j+1} \neq \perp$. We have to show that K_{j+1} also has the desired properties. Assume that σ is the concrete target egd $\forall\mathbf{x}\forall\mathbf{t}(\theta(\mathbf{x},\mathbf{t}\wedge\rho(\mathbf{t})) \rightarrow x_k = x_l)$. Since $K_{j+1} \neq \perp$, we must have that K_{j+1} is obtained from K_j in one of two ways: (1) a null (with or without a time-stamp) is replaced by a constant or by a null with no time-stamp; (2) a concrete time-stamped null with some time-stamp i is replaced by another time-stamped null with the same time-stamp i. In either case and since K_j had the desired properties, it follows that K_{j+1} also has the desired properties. This completes the proof of the lemma. $\square$

We are now ready to prove Theorem 6.

Proof. (Proof of Theorem 6) Let $\mathcal{M}$ be a concrete schema mapping satisfying the hypothesis of the theorem and let I be a concrete source instance. If the concrete chase algorithm succeeds on I, then the concrete target instance $\text{chase}_{\mathcal{M}}(I)$ returned by this algorithm is a concrete universal solution for I w.r.t. $\mathcal{M}$. Indeed, this follows immediately from Theorem 4, since that theorem is about arbitrary concrete schema mappings. So, assume now that the concrete chase algorithm fails on I. We have to show that no solution for I w.r.t. $\mathcal{M}$ exists. We consider two cases about $\mathcal{M}$.

Case 1: Assume that every concrete s-t tgd in Σ_{st} is full. It follows that no nulls and no concrete time-stamped nulls are generated during the chase, hence the target instances constructed during the chase contain no nulls and no concrete time-stamped nulls.

Let $\langle I, J\rangle \xrightarrow{\sigma,s} \perp$ be the last chase step of the failing chase and let σ be the (concrete) target egd

$$\sigma_t = \forall \mathbf{x} \forall \mathbf{t} (\theta(\mathbf{x}, \mathbf{t}) \wedge \rho(\mathbf{t}) \rightarrow x_k = x_l).$$

Since the chase fails at this step and since J contains no nulls and no concrete time-stamped nulls, there must exist an assignment s from the variables in $\mathbf{x}$ and $\mathbf{t}$ to the active domain $adom(J)$ of J such that $J \models \theta(s(\mathbf{x}), s(\mathbf{t})) \wedge \rho(s(\mathbf{t}))$, and the values $s(x_k)$, $s(x_l)$ are different constants, say, $s(x_k) = c_1 \neq c_2 = s(x_l)$.

Towards a contradiction, suppose that there exists a solution J' for I w.r.t. $\mathcal{M}$. Since the identity is a homomorphism from $\langle I, \emptyset\rangle$ to $\langle I, J'\rangle$, Lemma 3 implies that there is a homomorphism g from $\langle I, J\rangle$ to $\langle I, J'\rangle$. Then $g \circ s$ is an assignment from the variables $\mathbf{x}$ and $\mathbf{t}$ to the active domain $adom(J')$ of J' such that $J' \models \theta(g \circ s(\mathbf{x}), g \circ s(\mathbf{t})) \wedge \rho(g \circ s(\mathbf{t}))$. Moreover, since J' is a solution for I w.r.t. $\mathcal{M}$, we must have that $g \circ s(x_k) = g \circ s(x_l)$. However, since g is the identity on constants, we have that $s(x_k) = s(x_l)$, and so we have arrived at a contradiction.

Case 2: Assume that for every concrete s-t tgd in Σ_{st} if it is not full, then it contains exactly one temporal variable occurring in every atom of its consequent. Moreover, every target egd in Σ_t contains at most one temporal variable.

Let $\langle I, J\rangle \xrightarrow{\sigma,s} \perp$ be the last chase step of the failing chase. Since σ is a (concrete) target egd in Σ_t, it must contains at most one temporal variable. Suppose first that the only temporal variable occurring in σ is t, hence σ is a formula of the form

$$\sigma_t = \forall \mathbf{x} \forall t (\theta(\mathbf{x}, t) \rightarrow x_k = x_l).$$

Since the chase fails at this step, there exists an assignment s from the variables $\mathbf{x}$ and t to the active domain $adom(J)$ of J such that either $s(x_k)$ and $s(x_l)$ are two distinct constants c_1 and c_2 or $s(x_k)$ and $s(x_l)$ are two distinct concrete time-stamped nulls $N_1^{i_1}$ and $N_2^{i_2}$ with two different time-stamps $i_1 \neq i_2$.

We now claim that $s(x_k)$ and $s(x_l)$ cannot be concrete time-stamped nulls with different time stamps. By Lemma 7, if a concrete time-stamped null N^i occurs in a fact of J, then the time interval i must also occur in that fact. Since t is the only temporal variable occurring in σ, it follows that if $s(x_k)$ and $s(x_l)$ were concrete time-stamped nulls, then they would have the same time-stamp $s(t)$. Thus, we are left with the possibility that $s(x_k) = c_1$ and $s(x_l) = c_2$, where c_1 and c_2 are distinct constants. Towards a contradiction, suppose that there exists a solution J' for I w.r.t. $\mathcal{M}$. As in the previous case and by using Lemma 3, we have that there is a homomorphism g from $\langle I, J \rangle$ to $\langle I, J' \rangle$. Then $g \circ s$ is an assignment from the variables $\mathbf{x}$ and t to the active domain $adom(J')$ of J' such that $J' \models \theta(g \circ s(\mathbf{x}), g \circ s(t))$. Moreover, since J' is a solution for I w.r.t. $\mathcal{M}$, we must have that $g \circ s(x_k) = g \circ s(x_l)$. Since g is the identity on constants, we conclude that $s(x_k) = s(x_l)$, which is a contradiction.

Next, suppose that σ contains no temporal variables, i.e., σ is a standard target egd of the form $\forall \mathbf{x}(\theta(\mathbf{x}) \rightarrow x_k = x_l)$. Since the chase fails at this step, there is an assignment s from the variables $\mathbf{x}$ to the active domain $adom(J)$ of J such that $J \models \theta(s(\mathbf{x}))$ and either $s(x_k)$ and $s(x_l)$

are two distinct constants or $s(x_k)$ and $s(x_l)$ are two concrete time-stamped nulls with different time stamps. Since σ contains no temporal variables, Lemma 7 implies that no fact in $\theta(s(\mathbf{x}))$ contains a time-stamped null, hence $s(x_k)$ and $s(x_l)$ must be different constants. From this and with the same reasoning as before, it follows that there is no solution for I w.r.t. $\mathcal{M}$.

Finally, as proved in the Theorem 4, the running time of the concrete chase algorithm is bounded by a polynomial in the size of I. This completes the proof of Theorem 6. $\square$

Remark 3. In Theorem 6, we identified two sufficient conditions on schema mappings for which the concrete chase algorithm is sound and complete. These two sufficient conditions are:

(a) Every concrete s-t tgd in Σ_{st} is full (i.e., its consequent contains no existential quantifiers);

(b) If a concrete s-t tgd in Σ_{st} is not full, then it contains exactly one temporal variable, and this temporal variable occurs in every atom of the consequent of the concrete s-t tgd; moreover, every target egd in Σ_t contains at most one temporal variable.

We will show that these two conditions are not necessary for the conclusion of Theorem 6 to hold. For this, we have to show that there exists a concrete schema mapping for which the conclusion of Theorem 6 holds, yet the schema mapping violates both conditions (a) and (b). Let $\mathcal{M}_4 = (\mathbf{S}, \mathbf{T}, \Sigma_{st}^4, \Sigma_t^4)$ be the concrete schema mapping, where Σ_{st}^4 consists of the three concrete s-t tgds

$$\sigma_{st}^1 = \forall x,t(P(x,t) \rightarrow P'(x,t)),\ \ \sigma_{st}^2 = \forall x,t(Q(x,t) \rightarrow Q'(x,t)),$$

$$\sigma_{st}^3 = \forall x,t_1,t_2(P(x,t_1) \wedge Q(x,t_2) \rightarrow \exists y R(x,y)),$$

and Σ_t consists of the concrete target egd

$$\sigma_t = \forall x, y, t_1, t_2 (R(x,y) \land P'(x,t_1) \land Q'(x,t_2) \to x = y).$$

Clearly, the concrete schema mapping $\mathcal{M}_4$ violates both conditions (a) and (b). Yet, we claim that the concrete chase algorithm is sound and complete w.r.t. $\mathcal{M}_4$, hence the conclusion of Theorem 6 holds. In fact, we claim that, given an arbitrary concrete source instance I, the concrete chase algorithm never fails on I w.r.t. $\mathcal{M}_4$. Therefore, by Theorem 4, the result returned by the concrete chase algorithm on I w.r.t. $\mathcal{M}_4$ is a universal solution for I w.r.t. $\mathcal{M}_4$.

We now explain why this claim holds. Observe that the concrete s-t tgds σ^1_{st} and σ^2_{st} simply copy the relations P and Q in the source to, respectively, the relations P' and Q' in the target. Therefore, the concrete chase algorithm populates the first attributes of the relations P' and Q' with constants from the source. In contrast, the concrete chase algorithm populates the second attribute in the relation R with time-stamped nulls, since R is populated only when the concrete s-t tgd σ^3_{st} is applied. Therefore, during the run of the chase, every assignment that satisfies the antecedent of the concrete target egd σ_t will assign a constant to x and a time-stamped null to y; consequently, the concrete chase algorithm will replace the time-stamped null by that constant, and so the chase algorithm will not fail on I w.r.t. $\mathcal{M}_4$.

3.2 Semantic Adequacy for Temporal Data Exchange with a Single Temporal Variable

3.2.1 Constraints in the Abstract Model of Time

Let $\mathbf{S}$ be a temporal source relational schema, and let $\mathbf{T}$ be a temporal target relational schema that is disjoint from $\mathbf{S}$. An s-t tgd in the abstract model of time, called an *abstract s-t tgd*, is a formula of the form:

$$\sigma_{st} = \forall \mathbf{x} \forall t \big(\varphi\big(\mathbf{x}, t\big) \rightarrow \exists \mathbf{y} \psi\big(\mathbf{x}, \mathbf{y}, t\big)\big), \tag{3.2}$$

where t is the only temporal variable, the formula $\varphi(\mathbf{x}, t)$ is a conjunction of atomic formulas of the form $R(\mathbf{x}', t)$ or of the form $R(\mathbf{x}')$ over the schema $\mathbf{S}$ (note that $\mathbf{x}' \subseteq \mathbf{x}$), and the formula $\psi\big(\mathbf{x}, \mathbf{y}, t\big)$ is a conjunction of similar target atoms over $\mathbf{T}$. We assume that all variables in $\mathbf{x}$ and the temporal variable t appear free in $\varphi(\mathbf{x}, t)$. An abstract target egd in the abstract model of time, called an *abstract target egd*, is a formula of the form:

$$\sigma_t = \forall \mathbf{x} \forall t (\theta(\mathbf{x}, t) \rightarrow x_k = x_l), \tag{3.3}$$

where the only temporal variable is t, the formula $\theta(\mathbf{x}, t)$ is a conjunction of target atoms over $\mathbf{T}$, and x_k, x_l are among the variables in $\mathbf{x}$.

Example 5. Let $\mathbf{S}$ be a temporal source schema consisting of the relation symbols

$$E(Name, Company, Time) \text{ and } S(Name, Salary, Time),$$

and let $\mathbf{T}$ be a temporal target schema consisting of the relation symbol $Emp(Name$, $Company$, *Salary*, *Position*, *Time*). An example of an abstract s-t tgd over **S** is

$$\forall n,s,c,t(\mathsf{E}(n,c,t)\wedge \mathsf{S}(n,s,t)\rightarrow \exists p\ \mathsf{Emp}(n,c,s,p,t)).$$

An example of an abstract target egd over **T** is

$$\forall n,c,s,p_1,p_2,t(\mathsf{Emp}(n,c,s,p_1,t)\wedge \mathsf{Emp}(n,c,s,p_2,t)\rightarrow p_1=p_2).$$

3.2.2 Abstract Data Exchange

Definition 6 (Abstract Schema Mapping)**.** An *abstract schema mapping* is a tuple $\mathcal{M} = (\mathbf{S},\mathbf{T},\Sigma_{st},\Sigma_t)$, where **S** is a temporal source relational schema, **T** is a temporal target relational schema that is disjoint from **S**, Σ_{st} is a finite set of standard s-t tgds or abstract s-t tgds, and Σ_t is a finite set of standard target egds or abstract target egds.

If $\mathcal{M} = (\mathbf{S},\mathbf{T},\Sigma_{st},\Sigma_t)$ is an abstract temporal schema mapping, then we write $\mathcal{M}' = (\mathbf{S}',\mathbf{T}',\Sigma'_{st},\Sigma'_t)$ for the standard (non-temporal) schema mapping obtained from $\mathcal{M}$ by removing the temporal attributes in the relation symbols in **S** and **T**, and also by removing the temporal variables from all constraints in Σ_{st} and Σ_t and replacing the atoms from **S** and **T** by the corresponding atoms from $\mathbf{S}'$ and $\mathbf{T}'$.

Values in Source and Target Instances As discussed earlier, in data exchange between relational schemas, the source instances contain values from a countable domain CONST of objects, called *constants*, while the target instances may contain values from the union CONST $\cup$ NULL, where NULL is a countable set of distinct *labeled nulls* $N_1, N_2, \ldots$ that represent some unknown value and are typically used to witness the existentially quantified variables in the right-hand sides of s-t tgds. In abstract data exchange, the values occurring in source and target instances may also be time points. Furthermore, abstract target instances may contain labeled nulls $N_1, N_2, \ldots$.

Definition 7 (Homomorphisms, Abstract Solutions, and Abstract Universal Solutions). Let J and J' be two abstract instances over the same schema. A *homomorphism* from J to J' is a sequence of functions $h_0, h_1, \ldots$ such that (a) each h_i is a homomorphism from the snapshot J_{t_i} of J to the snapshot J'_{t_i} of J'; (b) for every two homomorphisms $h_l : J_{t_l} \to J'_{t_l}$ and $h_k : J_{t_k} \to J'_{t_k}$, if a labeled null N_j appears both in the snapshot J_{t_l} of J and the snapshot J_{t_k} of J, then $h_l(N_j) = h_k(N_j)$.

Let $\mathcal{M}$ be an abstract schema mapping and let $\mathcal{M}'$ be the standard schema mapping obtained from $\mathcal{M}$ as discussed above. Assume that I is an abstract source instance. We say that a target abstract instance J is an *abstract solution* for I w.r.t. $\mathcal{M}$ if each snapshot J_t of J is a solution for the corresponding snapshot I_t of I w.r.t. $\mathcal{M}'$. A target instance J is an *abstract universal solution* for I if J is an abstract solution for I w.r.t. $\mathcal{M}$ and, for every abstract solution J' for J w.r.t. $\mathcal{M}$, there is a homomorphism from J to J'.

Golshanara and Chomicki [36] introduced a *snapshot chase algorithm* for abstract schema mappings and abstract source instances. In their setting, every relation symbol in the source and the target has exactly one temporal attribute, while in the setting described here every relation in the source and the target has at most one temporal attribute; in particular, some relations may have no temporal attributes. The snapshot chase can easily be extended to the latter setting.

The Snapshot Chase Algorithm Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st}, \Sigma_t)$ be an abstract schema mapping and let $\mathcal{M}'$ be the standard schema mapping obtained from $\mathcal{M}$ by removing all temporal variables from the relations and the constraints, as described earlier. Let I be a finite abstract source instance over the schema $\mathbf{S}$ and let $\langle I_0, I_1, \ldots, I_n \rangle$ be the sequence of its snapshots. Recall that each snapshot I_i contains all facts of I in which the time point i occurs; furthermore, every fact of a non-temporal relation belongs to every snapshot.

The *snapshot chase algorithm on I with respect to $\mathcal{M}$* applies the standard chase algorithm on each snapshot I_i of I w.r.t. the standard schema mapping $\mathcal{M}'$; moreover, for each snapshot, the algorithm produces fresh labeled nulls that are different from the labeled nulls produced in other snapshots. If the standard chase algorithm fails on at least one snapshot I_i, then we say that the snapshot chase algorithm on I w.r.t. $\mathcal{M}$ *fails*. Otherwise, the snapshot chase algorithm generates the abstract target instance s-chase$_{\mathcal{M}}(I)$ whose snapshot at time point i is the result of the standard chase on the snapshot I_i w.r.t. $\mathcal{M}'$, i.e.,

$$\text{s-chase}_{\mathcal{M}}(I) = \langle \text{chase}_{\mathcal{M}'}(I_0), \text{chase}_{\mathcal{M}'}(I_1), \ldots, \text{chase}_{\mathcal{M}'}(I_n) \rangle.$$

The next result extends Proposition 4 in [36] to the setting of abstract schema mappings considered here.

Proposition 1. Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st}, \Sigma_t)$ be an abstract schema mapping (thus, every constraint in $\Sigma_{st} \cup \Sigma_t$ contains at most one temporal variable). If I is an abstract source instance, then the following statements hold:

1. If the snapshot chase algorithm does not fail on I, then the abstract target instance s-chase$_{\mathcal{M}}(I)$ returned by this algorithm is an abstract universal solution for I w.r.t. $\mathcal{M}$.

2. If the abstract chase algorithm fails on I, then there is no abstract solution for I w.r.t. $\mathcal{M}$.

Furthermore, the running time of the *snapshot chase algorithm* is bounded by a polynomial in the size of the abstract instance I.

3.2.3 Semantic Adequacy of Concrete Universal Solutions

Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st}, \Sigma_t)$ be a schema mapping in which each constraint in $\Sigma_{st} \cup \Sigma_t$ contains at most one temporal variable. Such a schema mapping $\mathcal{M}$ is meaningful in both the concrete model of time and the abstract model of time without changing the constraints in $\Sigma_{st} \cup \Sigma_t$. In the first case, the temporal variable is ranging over time intervals and in the second over time points. We will often refer to such schema mappings as temporal schema mappings; it will be clear from the context if we regard them as concrete schema mappings or as abstract schema mappings. In this section, we will explore the interplay between concrete universal solutions and abstract universal solutions for such temporal schema mappings.

In what follows, we will also assume that all concrete source instances I are *coalesced*, that is, if $c_1, \ldots, c_m$ are constants and i, i' are time intervals such that $(c_1, \ldots, c_m, i)$ and $(c_1, \ldots, c_m, i')$ belong to the same relation of I, then i and i' are disjoint intervals. Clearly, every concrete source instance can be easily transformed to an "equivalent" coalesced one (see [30] for details).

Definition 8 (**Semantic Function).** As mentioned in Chapter 1, the semantic function $\llbracket . \rrbracket$ converts concrete instances to abstract instances by replacing time intervals with all points of time in them. We now spell out the precise definition of the semantic function.

- If $\mu = (c_1, \ldots, c_m, [s, e))$ is a tuple in which each c_k is a constant and $[s, e)$ is an interval, then $[\![\mu]\!] = \{(c_1,\ldots,c_m,t) : s \le t < e\}$.

- If $I = (R_1,\ldots,R_n)$ is a concrete source instance, then $[\![I]\!]$ is the abstract source instance $[\![I]\!] = ([\![R_1]\!],\ldots,[\![R_n]\!])$, where $[\![R_l]\!] = \bigcup\limits_{\mu\in R_l} [\![\mu]\!]$, for $1 \le l \le n$.

Definition 9 (Compatible Tuples). We say that a tuple $\nu = (a_1,\ldots,a_m,[s,e))$ is *compatible* if each a_k is a constant or a labeled null or a concrete time-stamped null $N_j^{[s,e)}$, and all time-stamped nulls in ν have the same time-stamp. If ν is a compatible tuple, then $[\![\nu]\!]$ is the set of all tuples $(b_1,\ldots,b_m,t)$, such that the following conditions hold:

1. $s \le t < e$.

2. If a_l is a constant or a labeled null, then $b_l = a_l$.

3. If a_l is a concrete time-stamped null $N_j^{[s,e)}$, then b_l is a new labeled null.

Let $J = (T_1,\ldots,T_m)$ be the concrete target instance produced by the concrete chase algorithm w.r.t. some temporal schema mapping $\mathcal{M}$ on a source instance I. It is easy to verify that every tuple occurring in one of the relations of J is compatible. Then $[\![J]\!]$ is the abstract target instance $[\![J]\!] = ([\![T_1]\!],\ldots,[\![T_m]\!])$, where $[\![T_l]\!] = \bigcup\limits_{\nu\in T_l} [\![\nu]\!]$, for $1 \le l \le m$.

Definition 10 (Semantic Adequacy). Let $\mathcal{M} = (\mathbf{S},\mathbf{T},\Sigma_{st},\Sigma_t)$ be a temporal schema mapping with at most one temporal variable per constraint and let I be a concrete source instance. We say that a concrete target instance J is *semantically adequate* for I if the abstract target instance $[\![J]\!]$ is a universal solution for $[\![I]\!]$ w.r.t. $\mathcal{M}$. In particular, $[\![J]\!]$ is homomorphically equivalent to the abstract universal solution J^a produced by the abstract chase on $[\![J]\!]$ w.r.t. $\mathcal{M}$.

The concept of semantic adequacy is illustrated in Figure 3.1.

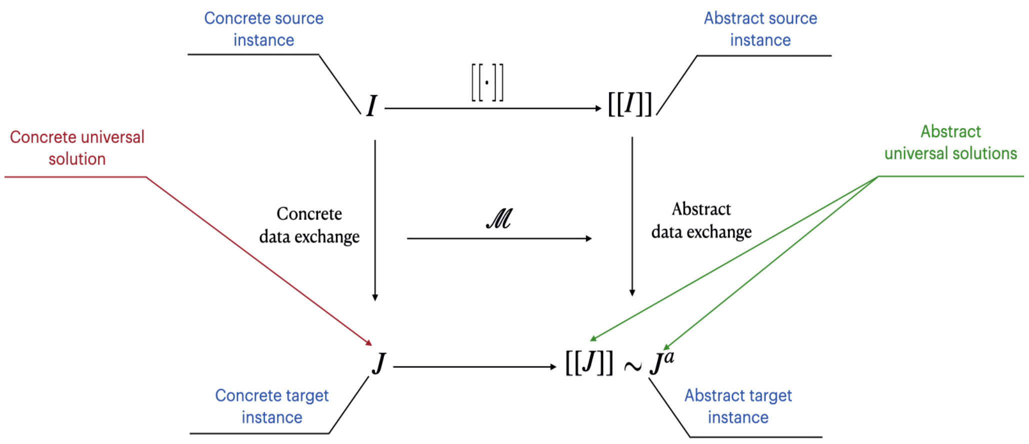


Figure 3.1: Illustration of the notion of semantic adequacy: if a concrete target instance J is semantically adequate for I w.r.t. $\mathcal{M}$, then $[\![J]\!]$ is a concrete universal solution for $[\![I]\!]$ w.r.t. $\mathcal{M}$; hence, $[\![J]\!]$ is homomorphically equivalent to every abstract universal solution J^a for I w.r.t. $\mathcal{M}$.

3.2.3.1 Lack of Semantically Adequate Concrete Universal Solutions

We begin by focusing more narrowly on schema mappings in the setting of Golshanara and Chomicki [36], that is, temporal schema mappings $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st}, \Sigma_t)$ such that each relational symbol in $\mathbf{S}$ and $\mathbf{T}$ has one temporal attribute and each constraint in $\Sigma_{st} \cup \Sigma_t$ has *exactly* one temporal variable. Note that such class of schema mappings does not contain standard schema mappings as a special case.

Let $\mathcal{M}$ be such a temporal schema mapping. Then, since every relation symbol contains a temporal attribute and since every s-t tgd contains exactly one temporal variable, we have that the temporal variable occurs in every atom in the antecedent of every s-t tgd; moreover, every target egd contains one temporal variable. Therefore, such temporal schema mappings satisfy the hypotheses of Theorem 6 in Section 3.1.3. Consequently, if I is a concrete source instance, such that the concrete chase algorithm succeeds on I w.r.t. $\mathcal{M}$, then this algorithm produces a concrete universal solution for I w.r.t. $\mathcal{M}$; furthermore, if this algorithm fails, then no solution for I w.r.t. $\mathcal{M}$ exists.

In their study of concrete data exchange, Golshanara and Chomicki [36] considered a variant of the concrete chase algorithm, which here we will call the *concrete n-chase algorithm*.

- The main difference between the concrete n-chase algorithm and the concrete chase algorithm we introduced here is that the concrete n-chase algorithm performs a *normalization* step before the constraints in Σ_{st} are applied and another normalization step before the constraints in Σ_t are applied. In particular, the concrete n-chase algorithm does not chase the given concrete source instance I with Σ_{st}, but, instead, chases the normalized instance $\mathcal{N}(I)$ with Σ_{st}. We refer the reader to Section 4.2 in [36] for the definition of normalization.

- In terms of similarities between the concrete n-chase algorithm and the concrete chase algorithm, we first note that, since the constraints have exactly one temporal variable, the target instances produced by either algorithm contain no labeled nulls, but, of course, they may contain time-stamped nulls in which the time-stamp is a single interval. Furthermore, the tuples occurring in the target instances produced by either algorithm are compatible.

In what follows, if $\mathcal{M}$ is a temporal schema mapping and I is a concrete source instance, we will write c-chase$_{\mathcal{M}}(I)$ and n-chase$_{\mathcal{M}}(I)$ to denote the concrete target instance produced by the concrete chase algorithm and, respectively, the concrete n-chase algorithm on I.

The following lemma about the concrete n-chase algorithm was proved in [36].

Lemma 8. *(Lemma 18 in [36])* Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st}, \Sigma_t)$ be a temporal schema mapping, such that each relational symbol in $\mathbf{S}$ and $\mathbf{T}$ has one temporal attribute and each constraint in $\Sigma_{st} \cup \Sigma_t$ has *exactly* one temporal variable. Let I be a normalized concrete source instance over $\mathbf{S}$ w.r.t. a constraint σ, let K be the current concrete target instance over $\mathbf{T}$ in the run of concrete n-chase algorithm, and let $\langle I, K\rangle \xrightarrow{\sigma,s} \langle I, K'\rangle$ be a chase step, where $\langle I, K'\rangle \neq \bot$. Assume that $\langle [\![I]\!], K^a\rangle$ is an abstract instance such that $\langle [\![I]\!], K^a\rangle$ satisfies σ and there exists a homomorphism h_1: $[\![\langle I, K\rangle]\!] \to \langle [\![I]\!], K^a\rangle$. Then there exists a homomorphism h_2: $[\![\langle I, K'\rangle]\!] \to \langle [\![I]\!], K^a\rangle$.

We are now ready to state the main result in [36].

Theorem 9. *(Theorem 19 in [36])* Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st}, \Sigma_t)$ be a temporal schema mapping, such that each relational symbol in $\mathbf{S}$ and $\mathbf{T}$ has one temporal attribute and each constraint in $\Sigma_{st} \cup \Sigma_t$ has *exactly* one temporal variable. If I is a concrete source instance, then the following statements are true.

- If the concrete n-chase algorithm on I fails, then there is no solution for $[\![I]\!]$ w.r.t. $\mathcal{M}$.
- If the concrete n-chase algorithm on I does not fail, then the concrete target instance n-chase$_{\mathcal{M}}(I)$ produced by the algorithm is semantically adequate for I.

We note that the normalization steps in the concrete n-chase algorithm guarantee that there is a homomorphism from the left-hand side of a constraint in Σ_{st} or in Σ_t to a concrete instance K, provided there is a homomorphism from the left-hand side of that constraint to the abstract instance $[\![K]\!]$.

As mentioned in Chapter 1, Golshanara and Chomicki [36] did not address the question of whether or not their concrete n-chase algorithm produces a concrete universal solution. In fact, the notion of a concrete universal solution is *not* introduced in [36]. Our next result provides a strong negative answer to the question of whether the concrete n-chase algorithm produces concrete universal solutions.

Theorem 10. There is a temporal schema mapping $\mathcal{M}^* = (\mathbf{S}, \mathbf{T}, \Sigma^*_{st}, \Sigma^*_t)$ with one temporal variable in each constraint in $\Sigma^*_{st} \cup \Sigma^*_t$ and there is a concrete source instance I^* such that the following properties hold:

1. The concrete target instance n-chase$_{\mathcal{M}^*}(I^*)$ returned by the concrete n-chase algorithm on I^* is neither a solution for I^* w.r.t. $\mathcal{M}^*$ nor a solution for the normalized instance $\mathcal{N}(I^*)$ w.r.t. $\mathcal{M}^*$.

2. There is a concrete universal solution for I^* w.r.t. $\mathcal{M}^*$, but there is no concrete universal solution for I^* w.r.t. $\mathcal{M}^*$ that is semantically adequate for I^*.

3. There is a concrete universal solution for $\mathcal{N}(I^*)$ w.r.t. $\mathcal{M}^*$, but there is no concrete universal solution for $\mathcal{N}(I^*)$ w.r.t. $\mathcal{M}^*$ that is semantically adequate for $\mathcal{N}(I^*)$.

Proof. Let $\mathcal{M}^* = (\mathbf{S}, \mathbf{T}, \Sigma^*_{st}, \Sigma^*_t)$ be the schema mapping where Σ^*_{st} consists of the constraints

$$\forall n, s, c, t (\mathsf{E}(n, c, t) \wedge \mathsf{S}(n, s, t) \rightarrow \mathsf{Emp}(n, c, s, t))$$

$$\forall n, p, t (\mathsf{P}(n, p, t) \rightarrow \exists c \ \mathsf{EmpPos}(n, c, p, t))$$

and Σ^*_t consists of the constraint

$$\forall n, c_1, c_2, s, p, t (\mathsf{Emp}(n, c_1, s, t) \wedge \mathsf{EmpPos}(n, c_2, p, t) \rightarrow c_1 = c_2).$$

Let I^* be the concrete source instance whose relations are depicted in Table 3.12. After normalizing I^* w.r.t. Σ^*_{st} (see [36] for the precise definition of normalization), we obtain the normalized instance $\mathcal{N}(I^*)$ whose relations are depicted in Table 3.13.

Table 3.12: The relations E, S, and P of the concrete source instance I^*.

(a) E

Name	Company	Time
Ada	IBM	$[2013,2018)$
Bob	IBM	$[2012,2015)$

(b) S

Name	Salary	Time
Ada	18000	$[2014,2018)$
Bob	13000	$[2013,2015)$

(c) P

Name	Position	Time
Ada	Manager	$[2015,2017)$
Bob	Consultant	$[2012,2015)$

Table 3.13: The relations E, S, and P of the normalized instance $\mathcal{N}(I^*)$.

(a) E

Name	Company	Time
Ada	IBM	$[2013,2014)$
Ada	IBM	$[2014,2018)$
Bob	IBM	$[2012,2013)$
Bob	IBM	$[2013,2015)$

(b) S

Name	Salary	Time
Ada	18000	$[2014,2018)$
Bob	13000	$[2013,2015)$

(c) P

Name	Position	Time
Ada	Manager	$[2015,2017)$
Bob	Consultant	$[2012,2015)$

Let n-chase$_{\mathcal{M}^*}(I^*)$ be the concrete target instance produced by the concrete n-chase algorithm on I^*; its relations are depicted in Table 3.14. It is easy to see that n-chase$_{\mathcal{M}^*}(I^*)$ is neither a solution for I^* nor a solution for $\mathcal{N}(I^*)$. This proves the first part of the theorem.

Let c-chase$_{\mathcal{M}^*}(I^*)$ and c-chase$_{\mathcal{M}^*}(\mathcal{N}(I^*))$ be the concrete target instances produced by the concrete chase algorithm on I^* and on $\mathcal{N}(I^*)$. The relations of c-chase$_{\mathcal{M}^*}(I^*)$ are depicted in Table 3.15, and those of c-chase$_{\mathcal{M}^*}(\mathcal{N}(I^*))$ in Table 3.16. Note that c-chase$_{\mathcal{M}^*}(I^*)$ is a universal solution for I^*, while c-chase$_{\mathcal{M}^*}(\mathcal{N}(I^*))$ is a universal solution for $\mathcal{N}(I^*)$.

Let s-chase$_{\mathcal{M}^*}([\![I^*]\!])$ be the abstract target instance produced by *snapshot chase algorithm* , which chases the the snapshots of $[\![I^*]\!]$. Table 3.17 depicts the abstract target instance

Table 3.14: The relations *Emp* and *EmpPos* of the concrete target instance n-chase$_{\mathcal{M}^*}(I^*)$.

(a) *Emp*

Name	Company	Salary	Time
Ada	IBM	18000	$[2014, 2015)$
Ada	IBM	18000	$[2015, 2017)$
Ada	IBM	18000	$[2017, 2018)$
Bob	IBM	13000	$[2013, 2015)$

(b) *EmpPos*

Name	Company	Position	Time
Ada	IBM	Manager	$[2015, 2017)$
Bob	$N_2^{[2012,2013)}$	Consultant	$[2012, 2013)$
Bob	IBM	Consultant	$[2013, 2015)$

Table 3.15: The relations *Emp* and *EmpPos* of the concrete target instance c-chase$_{\mathcal{M}^*}(I^*)$.

(a) *Emp*

Name	Company	Salary	Time

(b) *EmpPos*

Name	Company	Position	Time
Ada	$N_1^{[2015,2017)}$	Manager	$[2015, 2017)$
Bob	$N_2^{[2012,2015)}$	Consultant	$[2012, 2015)$

Table 3.16: The relations *Emp* and *EmpPos* of the concrete target instance c-chase$_{\mathcal{M}^*}(\mathcal{N}(I^*))$.

(a) *Emp*

Name	Company	Salary	Time
Ada	IBM	18000	$[2014, 2018)$
Bob	IBM	13000	$[2013, 2015)$

(b) *EmpPos*

Name	Company	Position	Time
Ada	$N_1^{[2015,2017)}$	Manager	$[2015, 2017)$
Bob	$N_2^{[2012,2015)}$	Consultant	$[2012, 2015)$

s-chase$_{\mathcal{M}^*}(\llbracket I^* \rrbracket)$ by listing the facts of each of its snapshots.

By Proposition 1, we have that s-chase$_{\mathcal{M}^*}(\llbracket I^* \rrbracket)$ is a universal solution for $\llbracket I^* \rrbracket$ w.r.t. $\mathcal{M}^*$. It is now easy to verify that $\llbracket$c-chase$_{\mathcal{M}^*}(I^*)\rrbracket$ is *not* homomorphically equivalent to s-chase$_{\mathcal{M}^*}(\llbracket I^* \rrbracket)$. It follows that c-chase$_{\mathcal{M}^*}(I^*)$ is *not* semantically adequate for I^*. Further-

Table 3.17: Snapshots the abstract target instance s-chase$_{\mathcal{M}^*}(\llbracket I^* \rrbracket)$.

2012	$\{$*EmpPos*(Bob, N_3, Consultant)$\}$
2013	$\{$*Emp*(Bob, IBM, 13000), *EmpPos*(Bob, IBM, Consultant)$\}$
2014	$\{$*Emp*(Ada, IBM, 18000), *Emp*(Bob, IBM, 13000), *EmpPos*(Bob, IBM, Consultant)$\}$
2015	$\{$*Emp*(Ada, IBM, 18000), *EmpPos*(Ada, IBM, Manager)$\}$
2016	$\{$*Emp*(Ada, IBM, 18000), *EmpPos*(Ada, IBM, Manager)$\}$
2017	$\{$*Emp*(Ada, IBM, 18000)$\}$

more, it is not hard to show that if J and J' are universal solutions for I^* w.r.t. $\mathcal{M}^*$, then $\llbracket J \rrbracket$ and $\llbracket J' \rrbracket$ are homomorphically equivalent. Therefore, no concrete universal solution for I^* is semantically adequate for I^*. This proves the second part of the theorem. A similar argument with c-chase$_{\mathcal{M}^*}(\mathcal{N}(I^*))$ in place of c-chase$_{\mathcal{M}^*}(I^*)$ proves the third part of the theorem. □

3.2.3.2 Existence of Semantically Adequate Concrete Universal Solutions

Theorem 10 tells that in the temporal data exchange setting studied in [36], there are rather simple temporal schema mappings and temporal source instances for which no concrete universal solution is semantically adequate for these instances or for their normalized versions. A close scrutiny of the proof of Theorem 10 reveals that one of the root causes for this state of affairs appears to be the presence of two temporal atoms in the antecedent of the temporal target egd in Σ_t^*. Our next result identifies a class of temporal schema mappings for which normalized instances have concrete universal solutions that are also semantically adequate.

We first state a useful lemma.

Lemma 11. Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st}, \Sigma_t)$ be a temporal schema mapping such that the following conditions hold: (1) each concrete s-t tgd contains at most one temporal variable; (2) if an s-t tgd contains no temporal variables, then it is full; (3) if a concrete s-t tgd contains a temporal variable, then that temporal variable occurs in every atom of its consequent; (4) each concrete target egd contains at most one temporal atom in its antecedent. Let I be a normalized concrete source instance over $\mathbf{S}$ w.r.t. Σ_{st}, let K be the current concrete target instance over $\mathbf{T}$ in the run of the concrete chase algorithm, and let $\langle I, K\rangle \xrightarrow{\sigma, s} \langle I, K'\rangle$ be a chase step, where $\langle I, K'\rangle \neq \perp$. Assume that $\langle [\![I]\!], K^a\rangle$ is an abstract instance such that each snapshot of $\langle [\![I]\!], K^a\rangle$ satisfies σ' where σ' is obtained from σ by removing all occurrences of the temporal variable, and there exists a homomorphism h: $[\![\langle I, K\rangle]\!] \rightarrow \langle [\![I]\!], K^a\rangle$. Then there exists a homomorphism h': $[\![\langle I, K'\rangle]\!] \rightarrow \langle [\![I]\!], K^a\rangle$.

The proof of Lemma 11 is omitted as it is similar to the proof of Lemma 8. Note that Condition (4) in Lemma 11 implies that if σ is a target egd in Σ_t, then the current target instance K is automatically normalized w.r.t. σ.

Theorem 12. Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st}, \Sigma_t)$ be a temporal schema mapping such that the following conditions hold: (1) each s-t tgd contains at most one temporal variable; (2) if an s-t tgd contains no temporal variables, then it is full; (3) if an s-t tgd contains a temporal variable, then that temporal variable occurs in every atom of its consequent; (4) each target egd contains at most one temporal atom in its antecedent. If I is a concrete source instance, then the following statements hold:

1. If the concrete chase algorithm does not fail on $\mathcal{N}(I)$, then c-chase$_{\mathcal{M}}(\mathcal{N}(I))$ is a semantically adequate concrete universal solution for $\mathcal{N}(I)$ (in particular, $[\![$c-chase$_{\mathcal{M}}(\mathcal{N}(I))]\!]$ is

an abstract universal solution for $[\![I]\!]$).

2. If the concrete chase algorithm fails on $\mathcal{N}(I)$, then there is no solution for $[\![I]\!]$ w.r.t. $\mathcal{M}$.

Proof. The proof is in two parts.

Part 1 Assume that the concrete chase algorithm does not fail. The first key observation is that if a concrete schema mapping satisfies the hypotheses (1)-(4), then it satisfies the hypothesis (b) of Theorem 6. Therefore, from Theorem 6, we know that c-chase$_{\mathcal{M}}(\mathcal{N}(I))$ is a concrete universal solution for $\mathcal{N}(I)$ w.r.t. $\mathcal{M}$.

Let J be a concrete target instance in the run of the concrete chase algorithm on $\mathcal{N}(I)$ w.r.t. Σ_{st} and let G be an abstract target instance in the run of the *snapshot chase algorithm* on $[\![I]\!]$ w.r.t. Σ_{st}. Then the concrete chase algorithm applies each target egd in Σ_t to J and the *snapshot chase algorithm* applies each target egd in Σ_t to G. Let J' be the current concrete target instance in the run of the concrete chase algorithm and let G' be the current abstract target instance in the run of the *snapshot chase algorithm* ($J' \neq \perp$ and $G' \neq \perp$). Assume that applying a concrete chase step over $\sigma_t = \forall \mathbf{x} \forall t \big(\theta\big(\mathbf{x}, t\big) \rightarrow x_k = x_l\big)$ to J' results in J'' ($J'' \neq \perp$); and assume that applying a chase step over σ'_t for each snapshot of G' results in G'' (and $G'' \neq \perp$), where $\sigma'_t = \forall \mathbf{x} \big(\theta'\big(\mathbf{x}\big) \rightarrow x_k = x_l\big)$. We claim that if $[\![J']\!]$ and G' are homomorphically equivalent, that is, $h_1 : [\![J']\!] \rightarrow G'$ and $h_2 : G' \rightarrow [\![J']\!]$, then $[\![J'']\!]$ and G'' are homomorphically equivalent. We now explain why this claim holds. Since σ_t contains at most one temporal atom, it is easy to prove that the concrete chase step over σ_t either replaces a time-stamped null Δ by a constant a throughout J' or do nothing. Since $[\![J']\!]$ and G' are homomorphically equivalent, h_1 and h_2 preserves constants; and $h_1(a)$ is a labeled null if a is a labeled null in $[\![J']\!]$; and $h_2(a)$ is a labeled null if a is a labeled null in G'. Condition (4) states that there is at most one temporal atom in the antecedent of each target egd, hence J' is normalized w.r.t. σ_t. It follows that if there is an assignment s from variables $\mathbf{x}$ in the $\theta(\mathbf{x},t)$ to the active domain of J' such that $s(t) = [s,e)$, then there is an assignment s_j ($s \leq j < e$) from variables $\mathbf{x}$ in the $\theta'(\mathbf{x})$ to the active domain of the snapshot J'_{t_j}, such that $J'_{t_j}, s_j \models \theta'(\mathbf{x})$, where the assignment s agrees with the assignment s_j on $\mathbf{x}$; and vice versa. In addition, for the snapshot $[\![J']\!]_{t_j}$, if there is an assignment s_j from variables $\mathbf{x}$ in $\theta'(\mathbf{x})$ to the active domain of $[\![J']\!]_{t_j}$, there is an assignment s'_j from variables $\mathbf{x}$ in $\theta'(\mathbf{x})$ to the active domain of G'_{t_j}, where $s'_j(x) = s_j(x)$ if $s_j(x)$ is a constant; $s'_j(x) = h_1(s_j(x))$ and $s'_j(x)$ is a labeled null if $s_j(x)$ is a labeled null in $[\![J']\!]_{t_j}$. Assume that $s_j(x_k) = c$ and $s_j(x_l) = \Delta$. Hence, $s'_j(x_k) = c$ and $s'_j(x_l) = h_1(\Delta)$. It follows that c replaced the labeled null Δ throughout $[\![J']\!]_{t_j}$ in $[\![J'']\!]_{t_j}$, and c replaced the labeled null $h_1(\Delta)$ throughout G'_{t_j} in G''_{t_j}. Furthermore, if there is an assignment s'_j from variables $\mathbf{x}$ in $\theta'(\mathbf{x})$ to the active domain of G'_{t_j}, then there is an assignment s_j from variables $\mathbf{x}$ in $\theta'(\mathbf{x})$ to the active domain of $[\![J']\!]_{t_j}$, where $s_j(x) = s'_j(x)$ if $s'_j(x)$ is a constant and $s_j(x) = h_2(s'_j(x))$ if $s'_j(x)$ is a labeled null in G'_{t_j}. Assume that $s'_j(x_k) = c$ and $s'_j(x_l) = \Delta$. Hence, $s_j(x_k) = c$ and $s_j(x_l) = h_2(\Delta)$. It follows that c replaced the labeled null Δ throughout G'_{t_j} in G''_{t_j}, and c replaced the labeled null $h_2(\Delta)$ throughout $[\![J']\!]_{t_j}$ in $[\![J'']\!]_{t_j}$. Therefore, there is a homomorphism h_3 from $[\![J'']\!]_{t_j}$ to G''_{t_j}, where $h_3(a) = h_1(a)$ if a is a labeled null in $[\![J'']\!]_{t_j}$. This is true because h_1 is a homomorphism. Furthermore, there is a homomorphism h_4 from G''_{t_j} to $[\![J'']\!]_{t_j}$, where $h_4(a) = h_2(a)$ if a is a labeled null in G''_{t_j}. This is true because h_2 is a homomorphism. Therefore, $[\![J'']\!]$ and G'' are homomorphically equivalent. Hence, the claim holds.

We need to prove that if the concrete chase algorithm does not fail, then the *snapshot* *chase algorithm* does not fail, and hence there exists a solution for $[\![I]\!]$. With the identity mapping from $[\![\langle I,\emptyset\rangle]\!]$ to $\langle [\![I]\!], G\rangle$, it is easy to prove that there is a homomorphism from $[\![J]\!]$ to G by repeatedly using Lemma 11. In addition, by Proposition 1, G is an abstract universal solution for $[\![I]\!]$ w.r.t. Σ_{st}. Therefore, $[\![J]\!]$ is an abstract universal solution for $[\![I]\!]$ w.r.t. Σ_{st}, and hence $[\![J]\!]$ and G are homomorphically equivalent. In addition, by repeatedly using the aforementioned claim, $[\![J']\!]$ is homomorphically equivalent to G'. Towards a contradiction, assume that there is a target egd $\sigma_t = \forall \mathbf{x} \forall t(\theta(\mathbf{x},t) \rightarrow x_k = x_l)$ in Σ_t, such that the concrete chase algorithm does not fail on the chase step over σ_t for J' and the *snapshot chase algorithm* fails on a chase step over σ_t for G' ($\sigma'_t = \forall \mathbf{x}(\theta'(\mathbf{x}) \rightarrow x_k = x_l)$). That is to say, there is an assignment s from $\theta(\mathbf{x},t)$ to the active domain of J' such that $s(t) = [s,e)$ and $s(x_k) = s(x_l)$. Condition (4) states that there is at most one temporal atom in the antecedent of each target egd, hence J' is normalized w.r.t. σ_t. It follows that there is an assignment s_j ($s \leq j < e$) from variables $\mathbf{x}$ in the $\theta'(\mathbf{x})$ to the active domain of the snapshot J'_{t_j}, such that $J'_{t_j}, s_j \models \theta'(\mathbf{x})$, where the assignment s agrees with the assignment s_j on $\mathbf{x}$. In addition, since $[\![J']\!]$ is homomorphically equivalent to G', there is an assignment s'_j from $\mathbf{x}$ in $\theta'(\mathbf{x})$ to the active domain of the snapshot G'_{t_j}, such that $s'_j(x_l) = s_j(x_l)$ if $s_j(x_l)$ is a constant and $s'_j(x_k) = s_j(x_k)$ if $s_j(x_k)$ is a constant. Since the *snapshot chase algorithm* fails on a snapshot G'_{t_j} w.r.t. σ'_t, we have $s'_j(x_l)$ and $s'_j(x_k)$ are two different constants, which is a contradiction.

Let $\langle [\![I]\!], K^a\rangle$ be an arbitrary solution for $[\![I]\!]$ w.r.t. $\mathcal{M}$, which means that $\langle [\![I]\!], K^a\rangle$ satisfies Σ_{st} and K^a satisfies Σ_t. The identity mapping from $[\![\langle I,\emptyset\rangle]\!]$ to $\langle [\![I]\!], \text{s-chase}_{\mathcal{M}}(I)\rangle$ is a homomorphism. By repeatedly utilizing Lemma 11, we have that there is a homomorphism from $[\![\langle I, \text{c-chase}_{\mathcal{M}}(\mathcal{N}(I))\rangle]\!]$ to $\langle [\![I]\!], \text{s-chase}_{\mathcal{M}}(I)\rangle$. Therefore, $[\![\text{c-chase}_{\mathcal{M}}(\mathcal{N}(I))]\!]$ is an abstract universal solution. Furthermore, since $\text{s-chase}_{\mathcal{M}}([\![I]\!])$ is a universal solution for $[\![I]\!]$ w.r.t. $\mathcal{M}$, the abstract instance $[\![\text{c-chase}_{\mathcal{M}}(\mathcal{N}(I))]\!]$ is homomorphically equivalent to $\text{s-chase}_{\mathcal{M}}([\![I]\!])$. Moreover, $\text{c-chase}_{\mathcal{M}}(\mathcal{N}(I))$ is a concrete universal solution for $\mathcal{N}(I)$ w.r.t. $\mathcal{M}$. Therefore, $\text{c-chase}_{\mathcal{M}}(\mathcal{N}(I))$ is a semantically adequate concrete universal solution $\mathcal{N}(I)$.

Part 2 Assume that the concrete chase algorithm fails on $\mathcal{N}(I)$ w.r.t $\mathcal{M}$. Let $\langle \mathcal{N}(I), J\rangle \xrightarrow{\sigma,s} \perp$ be the last chase step of the failing chase. Suppose that σ is the target egd $\forall \mathbf{x} \forall t(\theta(\mathbf{x},t) \to x_k = x_l)$. Since, by the hypothesis about $\mathcal{M}$, each target egd contains at most one temporal atom, we know that the temporal variable t appears in exactly one atom of $\theta(\mathbf{x},t)$. Since the chase fails at this step, there exists an assignment s from the variables $\mathbf{x}$ and t to the active domain $adom(J)$ of J such that $s(x_l) \neq s(x_k)$. As discussed in the proof of Theorem 6, $s(x_l)$ and $s(x_k)$ are two distinct constants. Towards a contradiction, suppose that there exists an abstract solution J^a for $[\![I]\!]$ w.r.t. $\mathcal{M}$. Lemma 11 implies that there is a homomorphism g from $[\![\langle \mathcal{N}(I), J\rangle]\!]$ to $\langle [\![I]\!], J^a\rangle$. Then we must have that $g \circ s(x_l) = g \circ s(x_k)$. Since g is the identity on constants, it follows that $s(x_l) = s(x_k)$, which is a contradiction. Consequently, there is no solution for $[\![I]\!]$ w.r.t. $\mathcal{M}$. Next, suppose that σ is a standard target egd of the form $\forall \mathbf{x}(\theta(\mathbf{x}) \to x_k = x_l)$. With a similar reasoning as before, we can also conclude that there is no solution for $[\![I]\!]$ w.r.t. $\mathcal{M}$. $\square$

Remark 4. In Theorem 12, we identified a set of four conditions, (1)-(4), on schema mappings, such that if a schema mapping satisfies all of those four conditions, then, given a source instance I, the concrete chase algorithm produces a semantically adequate universal solution for the normalized instance $\mathcal{N}(I)$, provided the algorithm does not fail on $\mathcal{N}(I)$. It can be shown that this set of conditions is not necessary for the conclusion of Theorem 12. Actually, it can be shown that there exists a concrete schema mapping $\mathcal{M}'$ with at most one temporal variable in each constraint, such that the conclusion of Theorem 12 holds for $\mathcal{M}'$, yet $\mathcal{M}'$ violates conditions (2), (3), and (4). Specifically, let $\mathcal{M}' = (\mathbf{S}, \mathbf{T}, \Sigma'_{st}, \Sigma'_t)$ be the concrete schema mapping, where Σ'_{st} consists of the four s-t tgds

$$\sigma^1_{st} = \forall x, t(P(x,t) \rightarrow P'(x,t)), \;\; \sigma^2_{st} = \forall x, t(Q(x,t) \rightarrow Q'(x,t)),$$

$$\sigma^3_{st} = \forall x, t(P(x,t) \wedge Q(x,t) \rightarrow \exists y\, R(x,y)), \;\; \sigma^4_{st} = \forall x, y(E(x,y) \rightarrow \exists y\, R(x,y)),$$

and Σ'_t consists of the concrete target egd

$$\sigma_t = \forall x, y, t(R(x,y) \wedge P'(x,t) \wedge Q'(x,t) \rightarrow x = y).$$

Clearly, the concrete schema mapping $\mathcal{M}'$ violates conditions (2), (3), and (4). Yet, the conclusion of Theorem 12 holds for $\mathcal{M}'$ because it is not hard to verify that the following two statements hold: (i) given an arbitrary concrete source instance I, the concrete chase algorithm never fails on $\mathcal{N}(I)$ w.r.t. $\mathcal{M}'$, hence it produces a concrete universal solution; (ii) $[\![\text{c-chase}_{\mathcal{M}'}(\mathcal{N}(I))]\!]$ is an abstract universal solution for $[\![I]\!]$, hence $\text{c-chase}_{\mathcal{M}'}(\mathcal{N}(I))$ is semantically adequate for $\mathcal{N}(I)$.

It should be pointed out that there are a schema mapping $\mathcal{M}''$ that satisfies the hypothesis in Theorem 12 and a concrete source instance I'' such that no semantically adequate concrete universal solution for I'' w.r.t. $\mathcal{M}''$ exists. This is shown in the next proposition, which is the last result in this chapter.

Proposition 2. There are a temporal schema mapping $\mathcal{M}'' = (\mathbf{S}, \mathbf{T}, \Sigma''_{st}, \Sigma''_t)$ and a concrete source instance I'' with the following properties:

1. Σ''_{st} consists of two constraints: the first contains exactly one temporal variable, and that

temporal variable occurs in every atom of its consequent; the second contains no temporal variables and it is a full s-t tgd.

2. Σ''_t consists of a single constraint with exactly one temporal atom in its antecedent.

3. There exists a concrete universal solution for I'' w.r.t. $\mathcal{M}''$, but there is no concrete universal solution for I'' w.r.t. $\mathcal{M}''$ that is semantically adequate for I''.

Proof. Let $\mathcal{M}'' = (\mathbf{S}, \mathbf{T}, \Sigma''_{st}, \Sigma''_t)$ be the schema mapping where Σ''_{st} consists of the constraints

$$\forall n, s, c, t (E(n, c, t) \wedge S(n, s, t) \to \exists p Emp(n, c, s, p, t))$$

$$\forall n, p (P(n, p) \to EmpPos(n, p))$$

and Σ''_t consists of the constraint

$$\forall n, c, s, p_1, p_2, t (Emp(n, c, s, p_1, t) \wedge EmpPos(n, p_2) \to p_1 = p_2).$$

Let I'' be the concrete source instance whose relations are depicted in Table 3.18. After applying the semantic function on I'', we obtain the abstract source instance $[\![I'']\!]$ whose relations are depicted in Table 3.19.

Table 3.18: The relations E, S, and P of the concrete source instance I''.

(a) E

Name	Company	Time
Ada	IBM	$[2013, 2018)$
Bob	IBM	$[2012, 2015)$

(b) S

Name	Salary	Time
Ada	18000	$[2014, 2018)$
Bob	13000	$[2013, 2015)$

(c) P

Name	Position
Ada	Manager
Bob	Consultant

Let c-chase$_{\mathcal{M}''}(I'')$ be the concrete target instance produced by the concrete chase algorithm on I''. The relations of c-chase$_{\mathcal{M}''}(I'')$ are depicted in Table 3.20. By Theorem 4, we

Table 3.19: The relations E, S, and P of the abstract source instance $[\![I'']\!]$.

(a) E

Name	Company	Time
Ada	IBM	2013
Ada	IBM	2014
Ada	IBM	2015
Ada	IBM	2016
Ada	IBM	2017
Bob	IBM	2012
Bob	IBM	2013
Bob	IBM	2014

(b) S

Name	Salary	Time
Ada	18000	2014
Ada	18000	2015
Ada	18000	2016
Ada	18000	2017
Bob	13000	2013
Bob	13000	2014

(c) P

Name	Position
Ada	Manager
Bob	Consultant

have that c-chase$_{\mathcal{M}''}(I'')$ is a universal solution for I''.

Table 3.20: The relations *Emp* and *EmpPos* of the concrete target instance c-chase$_{\mathcal{M}''}(I'')$.

(a) *Emp*

Name	Company	Salary	Position	Time

(b) *EmpPos*

Name	Position
Ada	Manager
Bob	Consultant

Let s-chase$_{\mathcal{M}''}([\![I'']\!])$ be the abstract target instance produced by chasing the snapshots of $[\![I'']\!]$. Table 3.21 depicts this abstract target instance by listing the facts of each of its snapshots.

As shown in [36], s-chase$_{\mathcal{M}''}([\![I'']\!])$ is a universal solution for $[\![I'']\!]$ w.r.t. $\mathcal{M}''$. It is now easy to verify that $[\![$c-chase$_{\mathcal{M}''}(I'')]\!]$ is *not* homomorphically equivalent to s-chase$_{\mathcal{M}''}([\![I'']\!])$.

Table 3.21: The snapshots of the abstract target instance s-chase$_{\mathcal{M}''}(\llbracket I'' \rrbracket)$.

2013	$\{$*Emp*(Bob, IBM, 13000, Consultant), *EmpPos*(Ada, Manager), *EmpPos*(Bob, Consultant)$\}$
2014	$\{$*Emp*(Ada, IBM, 18000, Manager), *Emp*(Bob, IBM, 13000, Consultant), *EmpPos*(Ada, Manager), *EmpPos*(Bob, Consultant)$\}$
2015	$\{$*Emp*(Ada, IBM, 18000, Manager), *EmpPos*(Ada, Manager),*EmpPos*(Bob, Consultant)$\}$
2016	$\{$*Emp*(Ada, IBM, 18000, Manager), *EmpPos*(Ada, Manager),*EmpPos*(Bob, Consultant)$\}$
2017	$\{$*Emp*(Ada, IBM, 18000, Manager), *EmpPos*(Ada, Manager),*EmpPos*(Bob, Consultant)$\}$

From this it follows, that c-chase$_{\mathcal{M}''}(I'')$ is *not* semantically adequate for I''. Furthermore, it is not hard to show that if J and J' are universal solutions for I'' w.r.t. $\mathcal{M}''$, then $\llbracket J \rrbracket$ and $\llbracket J' \rrbracket$ are homomorphically equivalent. It follows that no concrete universal solution for I'' is semantically adequate for I''. $\square$

Chapter 4

Relational-to-RDF Temporal Data Exchange

In the previous chapter, we have discussed the data exchange problem for temporal relational databases. This part of our work is the fundamental theory for us to explore how to exchange temporal data from temporal relational databases into RDF with temporal components. Section 4.1 offers an overview of the related work, while Section 4.2 and Section 4.3 delve into the specifics of this part of work. Specifically, we take the notion of temporal annotation in RDF graphs proposed in paper [41] and use it to enrich RDF with temporal components (See Section 4.2) in such a way that we can define the notion of temporal RDF graph schema, and thus we can define the data exchange problem for transferring data from temporal relational databases into RDF with temporal components (See Section 4.3). Finally, Section 4.3 presents the chase algorithm, which is designed to address the relational-to-RDF temporal data exchange problem.

4.1 Related Work

RDFS [55] is a language used in practice for describing ontologies, see [3] and the Protege Ontology Library[1] for RDFS-based ontology examples. The need for temporal annotations and reasoning arises in many application domains; toward addressing the need, Gutiérrez et al. [41] defined temporal RDF and studied properties of inference in temporal graphs. Tappolet and Bernstein [59] presented another approach, which focuses on querying. For the RDFS layer, research has been done on inference of temporal properties in temporal ontologies, see, e.g., [69]. At the same time, the temporal aspect is usually not included in the practical development of domain ontologies; our proposed approach in this chapter is designed to bridge this gap. In particular, we preserve the temporal source semantics in the target ontology-compliant domain data, by enabling Allen's relations, which are mentioned in Section 2.2, such as `during` or `before`, in queries on the data.

Recently, considerable formal work has been done on temporal ontology-mediated query access (OMQA), see [17]. OMQA differs in its objectives from data exchange, on which we focus in this chapter, as the latter considers mainly materialization of exchanged data, while the former concentrates on certain answers in query processing. In addition, OMQA uses a single schema, rather than the source and target schemas, which are clearly separated in data exchange. Boneva et al. [22, 24] have undertaken a preliminary work on data exchange from relations to RDF, followed up by a proposal of a tool [23] for mapping elements of the source schemas into the target RDFS ontology. Boneva et al. [23, 22, 24] did not address temporal aspects of data exchange. To the best of our knowledge, temporal data exchange between relational schemas

[1] `https://protegewiki.stanford.edu/wiki/Protege_Ontology_Library`

and ontologies has not been studied formally.

4.2 RDF with temporal enrichment

4.2.1 RDF vocabularies

Resources indicates Web resources, such as the title, author, modification date, content, and copyright information of a Web page. In the RDF model, all items, such as web documents and real-world objects, are modeled as *resources* using universal resource identifier *(URIs)*. Assume an infinite set U of RDF URIs, an infinite blank-node set $B = \{N_j : j \in \mathbb{N}\}$, and an infinite set L of RDF literals. An *RDF triple* is a statement of the form $(s, p, o) \in (U \cup B) \times U \times (U \cup B \cup L)$, where s is called *subject,* which can be a blank node denoted by a labeled null N_j; p is called *predicate;* and o is called *object.* An *RDF graph G* is a set of RDF triples. The *universe* of an RDF graph G, denoted by *universe*(G), is the set of elements of $U \cup B \cup L$ occurring in the triples of G.

RDF Vocabulary A *class* is a group of resources. The elements of a class are known as *instances* of the class. Each class is an instance of “rdfs:Class”. For example, “rdfs:Literal” is a class of all literal values that is an instance of the class “rdfs:Class.” In general, there are two types of classes: built-in classes and domain-specific classes.

A *property* is a resource describing relationships between subject resources and object resources. Each property is an instance of the class “rdfs:Property”, which is an instance of “rdfs:Class”. That is to say, all properties are instances of “rdfs:Class”. Given a property p, for any two classes c and c' such that there are two triples $(p, \text{rdfs:domain}, c)$, $(p, \text{rdfs:range}, c')$, the property p describes the relationship between c and c'. The property "rdfs:domain" in the triple $(p, \text{rdfs:domain}, c)$ restricts the subject resources of property p to the class c, while property "rdfs:range" in the statement $(p, \text{rdfs:range}, c')$ restricts the object resources of p to the class c'. Hence, for any two values c_i and c'_i that are instances of c and c', respectively, the triple (c_i, p, c'_i) implies that the resource c_i has a property p, and the property value is c'_i. For example, in the triple $(https://www.w3schools.com/rdf,\ http://homepage,\ https://www.w3schools.com)$, the value of the property "http://homepage" is "https://www.w3schools.com". There are two types of properties, built-in properties and domain-specific properties. The most important built-in properties are "rdfs:range" (which we abbreviate as [range]), "rdfs:domain" [dom], "rdfs:type" [type], "rdfs:subClassOf" [sc], and "rdfs:subPropertyOf" [sp] [42].

Table 4.1 and Table 4.2 show RDF classes and properties in AMR domain, respectively.

Classes can be *instances* of other classes; the fact that class c is an instance of class c' is denoted by the statement (c, type, c') with the property "rdf:type" (which we abbreviate as [type]). Classes can also be *subclasses* of other classes; the fact that class c is a subclass of class c'' is denoted by the statement (c, sc, c''). Likewise, properties can have *subproperties;* the fact that property p is a subproperty of property p' is denoted by the statement (p, sp, p'). Via the property "rdfs:subPropertyOf," all the resources related by the subproperty are also related by the superproperty [26, 39].

The *reification* vocabulary provides a mechanism for making statements about statements, using class "rdfs:Statement" [Statement] and properties "rdfs:subject" [subj], "rdfs:object" [obj], and "rdfs:predicate" [pred]. Figure 4.1 shows an RDF graph stating that the triple (state-

Element	**Class of**	**rdfs:subClassOf**	**rdf:type**
rdfs:Resource	all resources	rdfs:Resource	rdfs:Class
rdfs:Class	all classes	rdfs:Resource	rdfs:Class
rdfs:Literal	literal values	rdfs:Resource	rdfs:Class
rdfs:Datatype	datatypes	rdfs:Class	rdfs:Class
rdf:XMLLiteral	XML literal values	rdfs:Literal	rdfs:Datatype
rdf:Property	properties	rdfs:Resource	rdfs:Class
rdf:Statement	statements	rdfs:Resource	rdfs:Class
rdf:List	lists	rdfs:Resource	rdfs:Class
rdfs:Container	containers	rdfs:Resource	rdfs:Class
rdf:Bag	unordered containers	rdfs:Container	rdfs:Class
rdf:Seq	ordered containers	rdfs:Container	rdfs:Class
rdf:Alt	containers of alternatives	rdfs:Container	rdfs:Class
rdfs:Container MembershipProperty	rdf:_1 ... properties expressing membership	rdf:Property	rdfs:Class
Farms	all farms	NULL	rdfs:Class
Animal	all animals	NULL	rdfs:Class
Antibiotic Drug	all antibiotic drugs	Antimicrobial Drug	rdfs:Class
Antimicrobial Drug	all antimicrobial Drug	NULL	rdfs:Class

Table 4.1: Classes in AMR RDF vocabulary.

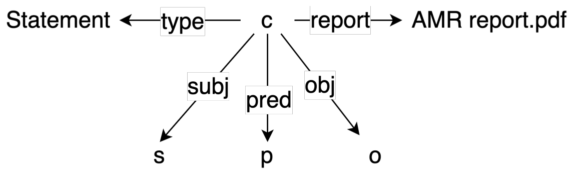


Figure 4.1: Given a triple $s \xrightarrow{p} o$, this is an example of reification being applied to the triple in an RDF graph.

Element	**Relates**	**rdfs:domain**	**rdfs:range**
rdfs:domain	restricts subjects	rdf:Property	rdfs:Class
rdfs:range	restricts objects	rdf:Property	rdfs:Class
rdf:type	instance of	rdfs:Resource	rdfs:Class
rdfs:subClassOf	subclass of	rdfs:Class	rdfs:Class
rdfs:subPropertyOf	subproperty of	rdf:Property	rdf:Property
rdfs:label	human readable label	rdfs:Resource	rdfs:Literal
rdfs:comment	human readable comment	rdfs:Resource	rdfs:Literal
rdfs:member	container membership	rdfs:Resource	rdfs:Resource
rdf:first	first element	rdf:List	rdfs:Resource
rdf:rest	rest of list	rdf:List	rdf:List
rdf:_1, rdf:_2, ...	container membership	rdfs:Container	rdfs:Resource
rdfs:seeAlso	further information	rdfs:Resource	rdfs:Resource
rdfs:isDefinedBy	definition	rdfs:Resource	rdfs:Resource
rdf:value	for structured values	rdfs:Resource	rdfs:Resource
rdf:object	object of statement	rdf:Statement	rdfs:Resource
rdf:predicate	predicate of statement	rdf:Statement	rdfs:Resource
rdf:subject	subject of statement	rdf:Statement	rdfs:Resource
livesIn	in which farm do animals live	Animal	Farms
usedOn	which Antibiotic drugs are used on animals	Antibiotic Drug	Animals

Table 4.2: Properties in AMR RDF vocabulary.

ment) (s, p, o) is reported in the file “AMR report.pdf.” (Here and in other Figures, we represent RDF graphs by representing each triple (s', p', o') in a given graph as $s' \xrightarrow{p'} o'$.)

4.2.2 RDF Temporal Enrichment, Temporal RDF Graph Schemas, and Instances

In this subsection, we review the notion of temporal annotations of RDF graphs defined in [41], and introduce the complementary concept of temporal markups in RDF graph schemas.

This allows us to define temporal RDF graph schemas and their instances, and to characterize the conditions under which they enable temporal annotations of data in a structured way [43], where time intervals represent temporal values carried by temporal annotations. The domain of time intervals is defined as a set $T_I = \{[s,e) : s < e \text{ and } s,e \in \mathbb{N}\}$.

In the remainder of this subsection, we will use the following reserved alphabets:

- x, y, $\ldots$ for non-temporal variables (set V) for values in $U \cup L$ and t for temporal variables (set T) for values in T_I in a formula;
- $a,b,c,\ldots$ for values in $U \cup L$ (values in L can be found only in the object positions of triples);
- $X,Y,Z,\ldots$ for blank nodes in B; and
- i for time intervals in T_I.

The above sets V, T, U, L, B, and T_I are all the infinite and disjoint from each other and from CONST.

A *temporal class* is a group of resources that represent temporal information. We consider temporal classes *TNode*, *Interval*, and *timeInterval*; here, *TNode* and *Interval* are "temporal-refication" classes, i.e., classes for statements about temporal statements. A *temporal property* is a property whose domain or range is a temporal class; we consider temporal properties *temporal*, *interval*, *validFor* (with values in *timeInterval*). The *temporal RDF vocabulary* is the result of adding these temporal classes and properties to the RDF vocabulary, see Tables 4.3 and 4.4. We call the set $W = \{\texttt{sc},\texttt{sp},\texttt{type},\texttt{dom},\texttt{range},\texttt{subj},\texttt{obj},\texttt{pred}\}$ the *RDFS vocabulary*, and call the set $W^* = W \cup \{\texttt{temporal, interval, validFor}\}$ the *temporal RDFS vocabulary*.

Element	**Class of**	**rdfs:subClassOf**	**rdf:type**
TNode	all temporal nodes	rdfs:Resources	rdfs:Class
Interval	all interval nodes	rdfs:Resources	rdfs:Class
timeInterval	all time intervals	rdfs:Resources	rdfs:Class

Table 4.3: The temporal classes in the temporal RDF vocabulary. Note that “TNode” and “Interval” are part of temporal reification.

Element	**Relates**	**rdfs:domain**	**rdfs:range**
temporal	provides temporal information	rdfs:statement	TNode
interval	provides interval information	TNode	Interval
validFor	provides valid time of intervals	Interval	timeIntervals

Table 4.4: The temporal properties in the temporal RDF vocabulary.

We now introduce the notions of temporal triples and of temporal RDF templates, which allow us to enable structural temporal annotation of triples via reification.

We call the set of triples $\mathcal{T}_t^{RDF} = \{(x, temporal, y), (y, interval, z), (z, \ validFor, \ t)\}$ the *temporal RDF template.* We say that $\mathcal{T}_t^{RDF}$ *is instantiated on triple* (s,p,o) *in a set of triples* G if there exists a substitution $\mu = \{x/k, y/l, z/m, t/i\}$, with $k,l,m \in U$ and $i \in T_I$, such that the result $\mathcal{T}_t^{RDF}|_\mu$ of applying μ to $\mathcal{T}_t^{RDF}$ is a subset of G, and so is the set of triples $Reif(s,p,o,\mu)$ $= \{(\mu(x), type, Statement), (\mu(x), subj, s), (\mu(x), pred, p), (\mu(x), obj, o)\}$. In this case, we say that the triple $(s,p,o) \in G$ *is temporally annotated with temporal-interval value i*, and call the set of triples $\mathcal{T}_t^{RDF}|_\mu \cup Reif(s,p,o,\mu)$ a *temporal-annotation structure* for (s,p,o) and i in G; the substitution μ here is said to *induce* the temporal annotation of (s,p,o) with i. (Notice that this notion of temporal annotation is simply a way to reify the triple (s, p, o) with a statement structure due to [41], whose objective is to "attach" the temporal-interval value *i* to the triple.) For any triple *r* in the result of applying any such substitution μ to the temporal RDF template $\mathcal{T}_t^{RDF}$, we say that *r* is a *temporal (RDF) triple.* For each graph *G* and for each triple (s, p, o) $\in G$ that is temporally annotated in *G* with some temporal-interval value *i*, with the annotation induced by a substitution μ, we say that the *temporal annotation of* (s, p, o) *with i is well typed* if *G* also contains the triples $(\mu(y), type, TNode)$, $(\mu(z), type, Interval)$, and $(i, type, timeInterval)$. For an illustration, see the subgraph shaded in green in Figure 4.2.

Example 6. As shown in Figure 4.2, the triples $(s, temporal, tn)$, $(tn, interval, ti)$, $(ti,\ validFor,$ $[1/1/2019, 1/5/2019])$ in green are temporal triples because they are the results of applying the substitution

$$mu = \{x/s, y/tn, z/ti, w/[1/1/2019, 1/5/2019]\}$$

to the temporal RDF template $\mathcal{T}_t^{RDF}$. These temporal triples and the triples $(s, type, Statement)$, $(s, subj, Ampicillin)$, $(s,\ pred,\ usedOn)$, $(s,\ obj,\ P1)$ compose a temporal-annotation structure for the RDF triple $(Ampicillin,\ usedOn,\ P1)$. This structure is also well typed as the triples $(tn, type, TNode)$, $(ti, type, Interval)$, $([1/1/2019, 1/5/2019],\ type,\ timeInterval)$ exist in *G*.

We call the set of triples

$$\begin{aligned}\mathcal{T}_t^{RDFS} = \{&(temporal, domain, Statement), (temporal, range, TNode),\\ &(interval, domain, TNode), (interval, range, Interval),\\ &(validFor, domain, Interval), (validFor, range, timeInterval)\}\end{aligned}$$

the *temporal RDFS markup set.* We say that $\mathcal{T}_t^{RDFS}$ *applies to a property n in a set of triples H* if H includes both $\mathcal{T}_t^{RDFS}$ (as a subset) and the triple $(Statement, pred, n)$ (as an element). In this case, we say that $\mathcal{T}_t^{RDFS} \cup \{(Statement, pred, n)\}$ *is the temporal markup for property n in H*; we also say that *property n in H admits temporal annotations.* For an illustration, see the subgraph shaded in blue in Figure 4.2.

A *temporal RDF graph* is a subset of $(U \cup B) \times U \times (U \cup B \cup L \cup T_I)$ that contains at least one temporal triple. The *universe* of a temporal RDF graph H, *universe*(H), is the set of elements of $U \cup B \cup L \cup T_I$ that occur in the triples of H. We say that such a graph H is a *well-formed temporal RDF graph* if for each temporal triple r_t in H, there exists a triple (s, p, o) $\in H$ and a temporal-interval value i such that r_t is an element of a temporal-annotation structure for (s, p, o) and i in H.

A *temporal RDF graph schema* O^T is a tuple $(\mathbf{C}, \mathbf{P}, G^T)$ in which $\mathbf{C}$ is a set of classes, $\mathbf{P}$ is a set of domain-specific properties, and G^T is an RDF graph that contains at least one temporal RDFS markup set $\mathcal{T}_t^{RDFS}$, called *temporal component* for simplicity, such that for each triple $(s, p, o) \in G^T$ the following holds:

- if p is *sc*, then $s \in \mathbf{C}$ and $o \in \mathbf{C}$;
- if p is *sp*, then $s \in \mathbf{P}$ and $o \in \mathbf{P}$;
- if p is *pred*, then s is $Statement$ and $o \in \mathbf{P}$;
- otherwise, $s \in \mathbf{P} \cup \{temporal, interval, validFor\}$; p is one of *domain* and *range*; and o $\in \mathbf{C}$.

We say that a temporal RDF graph schema $O^T = (\mathbf{C}, \mathbf{P}, G^T)$ is *well formed* if for each triple $r_t \in G^T$ such that r_t is an element of the temporal RDFS markup set $\mathcal{T}_t^{RDFS}$, r_t is an element of the temporal markup for some property n in G^T.

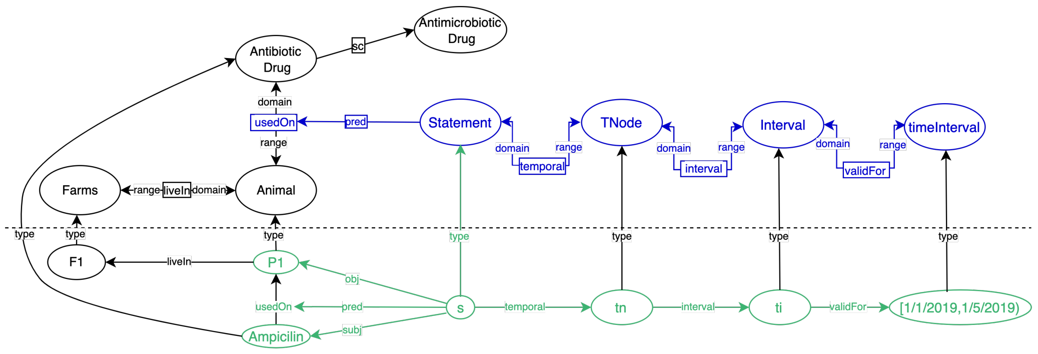


Figure 4.2: Example of a temporal RDF graph schema (above the dashed line) and its RDF instance (above and below the line); the notation we use originates from [8]. Here, an RDF triple *Ampicillin-[is]-usedOn-P1* is adorned with a temporal annotation shown in green. The annotation includes an instance of a (reified) statement *s* that links the triple to a temporal node *tn* characterized (via *ti*) as an interval, with (*validFor*) value [1/1/2019,1/5/2019). The schema (RDFS) layer indicates that the temporal-markup metadata for the annotation, in blue, contain the *Statement* class connected to the *TNode* class, which is, in turn, linked to classes *Interval* and *timeInterval.* Thus, both the temporal RDF graph schema and its instance are well formed.

Example 7. Consider a temporal RDF graph schema $O^T = (\mathbf{C}, \mathbf{P}, G^T)$ in the AMR domain. Here, $\mathbf{C} = \{$*AntimicrobialDrug, AntibioticDrug, Animal, Farm, Statement, TNode, Interval, timeInterval*$\}$; $\mathbf{P} = \{$*usedOn, liveIn*$\}$; and G^T is shown in the top half (above the dashed line) of Figure 4.2.

RDF Instances of Temporal RDF Graph Schemas Let $O^T = (\mathbf{C}, \mathbf{P}, G^T)$ be an RDF graph schema. Consider a temporal RDF graph H such that, for each triple $(a, p, b) \in H$,

- whenever the property p in (a, p, b) is `type`, then $b \in \mathbf{C}$; and
- whenever p in (a, p, b) is not `type`, then (i) G^T includes a triple $(p, domain, s)$ and a triple $(p, range, o)$, with $s, o \in \mathbf{C}$, and (ii) H includes triples $(a, \texttt{type}, s)$ and $(b, \texttt{type}, o)$.

Then we call the RDF graph $J = G^T \cup H$ an *RDF instance* of O. (Note that J includes the graph G^T of O.) We say that an RDF instance J of a well-formed RDF graph schema $O^T = (\mathbf{C}, \mathbf{P}, G^T)$ is *well formed* if (i) H is a well-formed temporal graph; (ii) each temporal annotation in H is well typed; and (iii) for each triple (s, p, o) that is temporally annotated in H, G^T has the triple $(Statement, pred, p)$.

Example 8. Fig. 4.2 shows a well-formed instance of the temporal RDF graph schema of Example 7.

An RDF graph schema can also be *atemporal.* In this case, the above definition is modified for $O^A = (\mathbf{C}, \mathbf{P}, G^A)$ to disallow in G^A (i) elements of the set $\mathcal{T}_t^{RDFS}$, and (ii) properties $temporal$, $interval$, and $validFor$ as subjects of triples. By definition, RDF instances of atemporal RDF graph schemas do not contain temporal triples; we call them *atemporal RDF instances.*

In this thesis, we focus on the temporal RDF graph schema since there has not been an extensive study of the data exchange problem for temporal information from relational databases into RDF data. For simplicity, in the remainder of the thesis, we will use $O = (\mathbf{C}, \mathbf{P}, G)$ to represent a temporal RDF graph schema, and we will assume that all the temporal RDF graph schemas and instances that we are given are well formed. Intuitively, well-formed temporal RDF graph schemas and instances enable temporal annotations on triples, for those triples whose predicates have temporal markups at the schema level. We follow here the approach of [41] of temporal annotations being done structurally; as explained in [43], reification is the only option for structural annotations.

4.3 Relational-to-RDF Temporal Schema Mappings

In this subsection, we formalize and study the *relational-to-RDF temporal data-exchange problem.* We focus on transferring relational data, with values in $\textsc{Const} \cup T_I$, into an RDF graph, with values in $U \cup B \cup L \cup T_I$. In the data transfer, we use *URI constructors*, which generate URIs from constants in relational sources or from RDF triples. We assume that we are given two constructor functions: the *class constructor* F_{uri}: $\textsc{Const} \times \cdots \times \textsc{Const} \rightarrow U$ and the *reification constructor* F_{ruri}: $(U \cup B) \times U \times (U \cup B \cup L \cup T_I) \rightarrow U \cup B$. We impose three requirements on the function F_{ruri}: (i) The value of F_{ruri} is a blank node iff at least one of its arguments is a blank node; (ii) F_{ruri} is injective.

4.3.1 Relational-to-RDF Temporal Schema Mappings

We now introduce the notion of *term* used in this chapter: (a) all variables are terms; (b) all properties and all classes are terms; (c) all constant symbols are terms, where a constant symbol is a URI in U, a literal in L; (d) whenever $\mathbf{T_x}$ is a set of terms and T_y, T_z, T_w are three distinct terms, then the URI constructors $F_{uri}(\mathbf{T_x})$ and $F_{ruri}(T_y, T_z, T_w)$ are also terms. Let

$O = (\mathbf{C}, \mathbf{P}, G)$ be a temporal RDF graph schema.

Let p, p', c, c', t be terms, where t is a temporal variable, $c, c' \in \mathbf{C}$ and $p, p' \in \mathbf{P}$. Let T_y be a term that is not a class or a property, and let T_x be a term in the form of $F_{uri}(\cdot)$ or $F_{ruri}(\cdot)$ (see (d) in the above paragraph). An *atomic formula over O* is an expression of one of the following forms:

1. (T_x, p', T_y)
2. (T_x, p'', T_y) for $p'' \in \{\texttt{temporal}, \texttt{interval}\}$ when T_x and T_y are in the form of $F_{ruri}(\cdot)$.
3. (T_x, p'', T_y) for $p'' \in \{\texttt{subj}\}$ when T_x in the form of $F_{ruri}(\cdot)$ and T_y is of the case (d).
4. $(t, \texttt{type}, \texttt{timeInterval})$
5. $(T_y, \texttt{type}, c)$ when T_y is not a temporal variable and c is not `timeInterval`
6. $(T_x, \texttt{pred}, p)$ when T_x is in the form of $F_{ruri}(\cdot)$
7. $(T_x, \texttt{validFor}, t)$ when T_x is in the form of $F_{ruri}(\cdot)$.

In each atomic formula (T_u, T_v, T_w) over O, we say that the term T_u is in the *position of subject*, the term T_v is in the *position of predicate*, and the term T_w is in the *position of object*.

Let $\mathbf{S}$ and O be a temporal source relational schema and a target temporal RDF graph schema, respectively.

Co-occuring atomic formulas : Let (T_1, p, T_2) be an atomic formula over O with $p \in \mathbf{P}$ which admits temporal annotations in O. We call the following atomic formulas the *co-occurring* atomic formulas for (T_1, p, T_2), where t is a temporal variable:

- $(F_{ruri}(T_1, p, T_2), \texttt{subj}, T_1)$
- $(F_{ruri}(T_1, p, T_2), \texttt{pred}, p)$
- $(F_{ruri}(T_1, p, T_2), \texttt{obj}, T_2)$
- $(F_{ruri}(T_1, p, T_2), \texttt{temporal}, F_{ruri}(F_{ruri}(T_1, p, T_2), \texttt{temporal}, t))$
- $(F_{ruri}(F_{ruri}(T_1, p, T_2), \texttt{temporal}, t), \texttt{interval},$
  $F_{ruri}(F_{ruri}(F_{ruri}(T_1, p, T_2), \texttt{temporal}, t), \texttt{interval}, t))$
- $(F_{ruri}(F_{ruri}(F_{ruri}(T_1, p, T_2), \texttt{temporal}, t), \texttt{interval}, t), \texttt{validFor}, t)$
- $(F_{ruri}(T_1, p, T_2), \texttt{type}, \texttt{Statement})$
- $(F_{ruri}(F_{ruri}(T_1, p, T_2), \texttt{temporal}, t), \texttt{type}, \texttt{TNode})$
- $(F_{ruri}(F_{ruri}(F_{ruri}(T_1, p, T_2), \texttt{temporal}, t), \texttt{interval}, t), \texttt{type}, \texttt{Interval})$
- $(t, \texttt{type}, \texttt{timeInterval})$

We use the notation $\mathcal{T}_t^{FL}$, called *co-occurring set* for (T_1, p, T_2), to denote the set that contains all the co-occurring atomic formulas of a given atomic formula (T_1, p, T_2) where p admits temporal annotations.

Definition 11. Relational-to-RDF full temporal s-t tgds In the context of the data exchange from the temporal relational database to data in RDF with temporal components, a *relational-to-RDF temporal full s-t tgd* is an expression of the form $\forall \mathbf{x}, \mathbf{t}(\varphi(\mathbf{x}, \mathbf{t}) \wedge \pi(\mathbf{t}) \rightarrow \psi(\mathbf{x}, \mathbf{y}, \mathbf{t}))$. Here, $\mathbf{t}$ are the only temporal variables; the formula $\varphi(\mathbf{x}, \mathbf{t})$ is a conjunction of atomic formulas with temporal variables over the schema $\mathbf{S}$; the formula $\pi(\mathbf{t})$ is a Boolean combination of *Allen atomic formulas* of the form $t_1 \ \rho \ t_2$ involving temporal variables in $\mathbf{t}$, with $\rho \in \{\mathsf{o}, \mathsf{d}, \prec, \mathsf{m}, \mathsf{s}, \mathsf{f}, =\}$; and $\psi(\mathbf{x}, \mathbf{y}, \mathbf{t})$ is a conjunction of atomic formulas over O.

Equation 4.6 in Example 9 illustrates an example of a relational-to-RDF temporal full s-t tgd.

In relational-to-relational temporal data exchange context, a *concrete full s-t tgd* is a concrete s-t tgd that contains no existential quantifier, and a *concrete GAV constraint* is a concrete full s-t tgd whose right-hand side is an atomic formula. Hence, concrete GAV constraint is a special case of concrete full s-t tgd. In the remaining part of this dissertation, we will use the term *relational-to-relational temporal full s-t tgds* to refer to concrete full s-t tgds, and we will use *relational-to-relational temporal GAV constraints* to refer to concrete GAV constraints. In addition, the term *relational-to-relational temporal GAV schema mappings* refers to the concrete schema mappings that each of its constraint is a relational-to-relational temporal GAV constraint.

Similarily, in the following definition, we will introduce the notion of relational-to-RDF temporal GAV constraints, which is also a special case of the relational-to-RDF temporal full s-t tgds.

Definition 12. Relational-to-RDF GAV temporal constraint In the context of the data exchange from the temporal relational database to data in RDF with temporal components, a *relational-to-RDF temporal GAV constraint*, which specifies how and what source data should appear in the target, is a first-order logic (FO) sentence of the form $\forall \mathbf{x}, \mathbf{t}(\varphi(\mathbf{x}, \mathbf{t}) \wedge \pi(\mathbf{t}) \rightarrow \psi(\mathbf{x}, \mathbf{y}, \mathbf{t}))$. Here, $\mathbf{t}$ are the only temporal variables; the formula $\varphi(\mathbf{x}, \mathbf{t})$ is a conjunction of atomic formulas with temporal variables over the schema $\mathbf{S}$; the formula $\pi(\mathbf{t})$ is a Boolean combination of *Allen atomic formulas* of the form $t_1\ \rho\ t_2$ involving temporal variables in $\mathbf{t}$, with $\rho \in \{\mathsf{o}, \mathsf{d}, \prec, \mathsf{m}, \mathsf{s}, \mathsf{f}, =\}$; and $\psi(\mathbf{x}, \mathbf{y}, \mathbf{t})$ is an atomic formula (T_1, p, T_2) over O where $p \in \mathbf{P}$ does not admit temporal annotation in O; or $\psi(\mathbf{x}, \mathbf{y}, \mathbf{t})$ is a conjunction of atomic formulas consisting of an atomic formula (T_1, p, T_2) over O with $p \in \mathbf{P}$ and all atomic formulas in the co-occurring set $\mathcal{T}_t^{FL}$ of (T_1, p, T_2) if p admit temporal annotation in O.

Note that each atomic formula over an RDF graph schema is analogous to an atomic formula over a temporal relational schema. However, unlike the relational databases where the conjunctions of any atomic formulas as the right-hand side of a relational-to-relational temporal s-t tgd is meaningful, the right-hand side of each relational-to-RDF temporal GAV constraint has extra restrictions to be meaningful. It requires that if there is an atomic formula (T_1, p, T_2) appearing in the right-hand side and the temporal RDFS markup set applies to p in O, then all other co-occurring atomic formulas of (T_1, p, T_2) also appear in the right-hand side of the constraint.

Definition 13. Relational-to-RDF temporal schema mappings A *relational-to-RDF temporal schema mapping* is a tuple $\mathcal{G} = (\mathbf{S}, O, \Sigma)$, where $\mathbf{S}$ is a temporal source relational schema, each O is a target temporal RDF graph schema, and Σ is a finite set of relational-to-RDF temporal GAV constraint.

In this work, we focus on the relational-to-RDF temporal data exchange problem that doesn't contain any existential quantifiers. This is because if the existential quantifiers introduced to the data exchange problem, the problem will be more complicated.

Example 9. Let **S** be a source schema with a temporal relation symbol

$$DrugUsage(farm, animal, drug, time)$$

and let O be a temporal RDF graph schema as below:

- $\mathbf{C} = \{Farms, Animals, AntibioticDrug, AntimicrobialDrug, Statement\}$;
- $\mathbf{P} = \{livesIn, usedOn\}$;
- Upper layer of Figure 4.2 is the G.

The following constraints are five relational-to-RDF temporal full s-t tgds from **S** to O^T is

$$\begin{aligned}
&\forall x_1, x_2, x_3, t \big(DrugUsage(x_1, x_2, x_3, t) \rightarrow \\
&\wedge(F_{uri}(x_3), \texttt{usedOn}, F_{uri}(x_2)) \\
&\wedge(F_{ruri}(F_{uri}(x_2), \texttt{usedOn}, F_{uri}(x_3)), \texttt{subj}, F_{uri}(x_3)) \\
&\wedge(F_{ruri}(F_{uri}(x_2), \texttt{usedOn}, F_{uri}(x_3)), \texttt{pred}, \texttt{usedOn}) \\
&\wedge(F_{ruri}(F_{uri}(x_2), \texttt{usedOn}, F_{uri}(x_3)), \texttt{obj}, F_{uri}(x_2)) \\
&\wedge(F_{ruri}(F_{uri}(x_2), \texttt{usedOn}, F_{uri}(x_3)), \texttt{temporal}, \\
&\quad F_{ruri}\big(F_{ruri}(F_{uri}(x_2), \texttt{usedOn}, F_{uri}(x_3)), \texttt{temporal}, t\big)) \\
&\wedge(F_{ruri}\big(F_{ruri}(F_{uri}(x_2), \texttt{usedOn}, F_{uri}(x_3)), \texttt{temporal}, t), \\
&\quad \texttt{interval}, F_{ruri}(F_{ruri}(F_{ruri}(F_{uri}(x_2), \texttt{usedOn}, F_{uri}(x_3)), \\
&\quad \texttt{temporal}, t\big), \texttt{interval}, t)) \\
&\wedge(F_{ruri}\big(F_{ruri}(F_{ruri}(F_{uri}(x_2), \texttt{usedOn}, \\
&\quad F_{uri}(x_3)), \texttt{temporal}, t), \texttt{interval}, t\big), \texttt{validFor}, t)\big).
\end{aligned} \tag{4.1}$$

$$\forall x_1, x_2, x_3, t\big(DrugUsage(x_1, x_2, x_3, t) \rightarrow (F_{uri}(x_2), \texttt{type}, \text{Animals}) \tag{4.2}$$

$$\forall x_1, x_2, x_3, t\big(DrugUsage(x_1, x_2, x_3, t) \rightarrow ((F_{uri}(x_1), \texttt{type}, Farms) \tag{4.3}$$

$$\forall x_1, x_2, x_3, t\big(DrugUsage(x_1, x_2, x_3, t) \rightarrow (F_{uri}(x_2), \text{livesIn}, F_{uri}(x_1)) \tag{4.4}$$

$$\forall x_1, x_2, x_3, t\big(DrugUsage(x_1, x_2, x_3, t) \rightarrow (F_{uri}(x_3), \texttt{type}, \text{AntibioticDrug}) \tag{4.5}$$

Note that with URI constructor F_{uri} and F_{ruri} constants from relational databases is assigned to a URI identifier.

For readability, we use d_1, d_2, and d_3 to represent terms using URI constructor F_{ruri} in the s-t tgd:

$$\begin{aligned}
d_1 =& F_{ruri}(F_{uri}(x_2), \texttt{usedOn}, F_{uri}(x_3)),\\
d_2 =& F_{ruri}(F_{ruri}(F_{uri}(x_2), \texttt{usedOn}, F_{uri}(x_3)), \texttt{temporal}, t)\\
=& F_{ruri}(d_1, \texttt{temporal}, t),\\
d_3 =& F_{ruri}\big(F_{ruri}(F_{ruri}(F_{uri}(x_2), \texttt{usedOn}, F_{uri}(x_3)), \texttt{temporal}, t), \texttt{interval}, t)\\
=& F_{ruri}(d_2, \texttt{interval}, t\big).
\end{aligned}$$

The s-t tgd in Equation 4.1 can be simply rewritten as follows:

$$\begin{aligned}
&\forall x_1, x_2, x_3, t\big(DrugUsage(x_1, x_2, x_3, t) \rightarrow\\
&\qquad ((F_{uri}(x_1), \texttt{type}, \text{Farms}) \wedge (F_{uri}(x_2), \texttt{type}, \text{Animals})\\
&\qquad \wedge (F_{uri}(x_2), \text{livesIn}, F_{uri}(x_1)) \wedge (F_{uri}(x_3), \texttt{type}, \text{AntibioticDrug})\\
&\qquad \wedge (F_{uri}(x_3), \texttt{usedOn}, F_{uri}(x_2)) \wedge (d_1, \texttt{subj}, F_{uri}(x_3))\\
&\qquad \wedge (d_1, \texttt{pred}, \texttt{usedOn}) \wedge (d_1, \texttt{obj}, F_{uri}(x_2)) \wedge (d_1, \texttt{temporal}, d_2)\\
&\qquad \wedge (d_2, \texttt{interval}, d_3) \wedge (d_3, \texttt{validFor}, t)\big).
\end{aligned} \tag{4.6}$$

4.3.2 Values in Source Instances and Target RDF Instances

In temporal data exchange from relational schemas to temporal RDF graph schema, the source instances contain values from a countable domain CONST of objects, called *constants*, as well as time intervals from the domain T_I.

As mentioned before, for our relational-to-RDF temporal data exchange problem we do not take existential quantifiers into account. Hence, values in the target RDF instances are URIs or RDF literals. Note that in reality, RDF literals have a special format, i.e., "$< string >$ "^^ $< datatype >$. For simplicity, in this thesis, we assume that L equals to CONST. That is to say, if a constant is transferred from a source instance into a target instance RDF to be an RDF literal, we directly use the constant as a literal instead of converting this constant into a value with the special format. As a result, in the data exchange, each constant in the temporal source relational instances would be transferred into URIs with the help of constructors or it would be directly populated into target RDF instances as RDF literals or time intervals. Consequently, a triple in the target RDF instance may contain values from the set $U \times U \times \{U \cup L \cup T_I\}$. The *universe* of an target RDF instance J, *universe*(J), is the set of elements of $\{U \cup L \cup T_I\}$ that occur in the triples of J.

Definition 14. Satisfaction We now spell out the semantics of the satisfaction for relational-to-RDF temporal full s-t tgds. Let $\mathbf{S}$ be a temporal relational source schema and let O be a temporal RDF graph schema. Let I be a source instance over $\mathbf{S}$. An RDF instance J is an RDF instance over O. Let $\sigma = \forall \mathbf{x}, \mathbf{t}(\varphi(\mathbf{x}, \mathbf{t}) \wedge \pi(\mathbf{t}) \rightarrow \psi(\mathbf{x}, \mathbf{t}))$ be a relational-to-RDF temporal full s-t tgd. For every assignment s from variables $\mathbf{x}$ and $\mathbf{t}$ to the active domain $adom(I)$ of I [2], if $s(\mathbf{x})$ is a tuple $\mathbf{a}$

[2] *Active domain* of an instance I, denoted by $adom(I)$, is the set of all values occurring in the relations of that

of constants and $s(\mathbf{x})$ is a tuple $\mathbf{i}$ of time intervals, such that $I \models \varphi(\mathbf{a},\mathbf{i}) \wedge \pi(\mathbf{i})$, then there is an assignment s' from terms that are in the positions of subject or object in $\psi(\mathbf{x},\mathbf{t})$ to the universe $universe(J)$ of J, and s' agrees with s on $\mathbf{x}$ and $\mathbf{t}$, such that the RDF instance J satisfies $\psi(\mathbf{x},\mathbf{t})$ with the assignment s', denoted as $J,s' \models \psi(\mathbf{x},\mathbf{t})$. The statement $J,s' \models \psi(\mathbf{x},\mathbf{t})$ indicates that

(1) if x is a variable occurring in $\psi(\mathbf{x},\mathbf{t})$ as a subject or an object of an atomic formula, and if $s(x) = a$, then $s'(x)$ is a;

(2) if t is a temporal variable occurring in $\psi(\mathbf{x},\mathbf{t})$ as a subject or an object of an atomic formula, and if $s(t) = i$, then $s'(t)$ is the time interval i;

(3) if c is in $P \cup W*$, $s'(c) = c = s(c)$;

(4) if $F_{uri}(\mathbf{x}')$ is a term occurring in $\psi(\mathbf{x},\mathbf{t})$, then $s'(F_{uri}(\mathbf{x}')) = F_{uri}(s(\mathbf{x}'))$, where $\mathbf{x}' \subseteq \mathbf{x}$;

(5) if $F_{ruri}(T_1,p,T_2)$ is a term occurring in $\psi(\mathbf{x},\mathbf{t})$, then
$s'(F_{ruri}(T_1,p,T_2)) = F_{ruri}(s'(T_1),p,s'(T_2))$, where $p \in \mathbf{P}$; and T_1, T_2 are two term, and all variables in each term are from $\mathbf{x}$ or $\mathbf{t}$;

(6) if (T_1,p,T_2) is an atomic formula in $\psi(\mathbf{x},\mathbf{t})$, then $s'((T_1,p,T_2)) = (s'(T_1),p,s'(T_2))$, and $(s'(T_1),p,s'(T_2))$ is a triple occurring in J, where the property $p \in \mathbf{P}$; and T_1, T_2 are terms and all variables in each term are from $\mathbf{x}$ or $\mathbf{t}$. We also say that $(I,J),s' \models \sigma$.

Definition 15. Logical implication Let $\mathcal{G}$ be a relational-to-RDF temporal schema mapping and let σ be a relational-to-RDF temporal full s-t tgd. We say that $\mathcal{G}$ logically implies σ if every (I,J) that satisfies each constraint of $\mathcal{G}$ also satisfies σ, where I is a source instance over the source schema in $\mathcal{G}$, and J is a target instance over the target schema.

Definition 16. Logical Equivalence Let $\mathcal{G}$ and $\mathcal{G}'$ be two relational-to-RDF temporal schema

instance.

mappings. We say that $\mathcal{G}$ is *logically equivalent* to $\mathcal{G}'$ if $\mathcal{G}'$ logically implies every constraint C in $\mathcal{G}$, and if $\mathcal{G}$ logically implies every constraint C' in $\mathcal{G}'$.

Definition 17. Homomorphisms between two RDF instances A *homomorphism* from an RDF instance J to an RDF instance J' is a mapping h from the active domain [3] of J to the active domain of J', such that the mapping preserves the value of URIs, RDF literals, and time intervals. In the other word, the following rules hold: a) $h(u) = u$ if $u \in U$, $h(l) = l$ if $u \in L$, and $h(i) = i$ if $i \in V$; b) for each blank node $b \in B$, we have $h(b) \in U \cup L \cup B$; c) if a triple (s, p, o) belongs to the RDF instance J, then $(h(s), h(p), h(o))$ belongs to the RDF instance J'.

Definition 18. Solutions and Universal Solutions Let $\mathcal{G} = (\mathbf{S}, O, \Sigma)$ be a relational-to-RDF temporal schema mapping and let I be a source instance over $\mathbf{S}$.

An RDF instance J is a *solution* of I w.r.t. $\mathcal{G}$ if $\langle I, J \rangle$ satisfies every relational-to-RDF temporal GAV constraint in Σ.

An RDF instance J is a *universal solution* for I w.r.t. $\mathcal{G}$ if J is a solution for I w.r.t. $\mathcal{G}$, and for every solution J' for I w.r.t. $\mathcal{G}$, there is a homomorphism from J to J'.

4.3.3 Chase for Relational-to-RDF Temporal Schema Mappings

Let $\mathcal{G} = (\mathbf{S}, O, \Sigma)$ be a temporal relational-to-RDF schema mapping. Assume a source instance I over $\mathbf{S}$ and target instance J over O. We call the tuple $\langle I, J \rangle$ over the schema $\mathbf{S} \cup O$ a *combined instance*. Sometimes, we will use the term *instances* to refer to combined instances. We design a chase algorithm to generate a solution for I w.r.t. $\mathcal{G}$. In the process of chasing the

[3]The *active domain* of a database is the set of all values occurring in the relations of that database.

instance, $K = (I,J)$ w.r.t. $\mathcal{G}$ (with J set initially to the graph G in O), for every constant in I mapped into an instance of a domain-specific class of O, the URI constructor F_{uri} generates the corresponding URI value in J.

Definition 19 (Chase step). For a given $\mathcal{G} = (\mathbf{S}, O, \Sigma)$, let I be a source instance, and let $K = (I, J_{curr})$ be a combined instance, where J_{curr} is the current target RDF instance.

Let σ: $\forall \mathbf{x}, \mathbf{t}\ (\varphi(\mathbf{x},\mathbf{t}) \wedge \pi(\mathbf{t}) \rightarrow \psi(\mathbf{x},\mathbf{t}))$ be a relational-to-RDF temporal GAV constraint in Σ^T, in which $|\mathbf{x}| = n$, $|\mathbf{y}| = l$, and $|\mathbf{t}| = m$, with $n,l,m \geq 0$.[4] Let h be a relational formula homomorphism for σ and I, such that $h(\mathbf{x})$ is a vector $\mathbf{c} = [c_1,\ldots,c_n]$ of values in CONST and $h(\mathbf{t})$ is a vector $\mathbf{i} = [i_1,\ldots,i_m]$ of values in T_I. Suppose that $I \models \varphi(\mathbf{c},\mathbf{i}) \wedge \pi(\mathbf{i})$, but $K \not\models \psi(\mathbf{c},\mathbf{i})$. Adding to J_{curr} the set of triples $\psi(\mathbf{c},\mathbf{i})$ gives rise to a new combined instance K'; by construction, $K' \models \varphi(\mathbf{c},\mathbf{i}) \wedge \pi(\mathbf{t}) \wedge \psi(\mathbf{c},\mathbf{i})$. We say that K' *is obtained from K via a chase step with* σ *and* h', and write $K \xrightarrow{\sigma,h} K'$.

Definition 20 (The relational-to-RDF chase algorithm). Let $\mathcal{G} = (\mathbf{S}, O, \Sigma)$, with $O = (\mathbf{C}, \mathbf{P}, G)$, be a relational-to-RDF temporal schema mapping, and I be a source instance. Let (I, G) be the combined instance K_0.

- We define a *chase sequence for I w.r.t* $\mathcal{G}$ as a sequence $K_0, K_1, K_2, \ldots$ in which, for all $j > 0$, $K_{j-1} \xrightarrow{\sigma,h_j} K_j$ for some $\sigma \in \Sigma$ and some homomorphism h_j.

- A *chase of I w.r.t* $\mathcal{G}$ is a finite chase sequence $K_0, K_1, K_2, \ldots, K_n$, such that $K_n \models \sigma$ for each σ in Σ^T. In this case, we say that $K_n = (I, J_n)$ is the *result of the chase,* and return J_n as a target RDF instance for I w.r.t $\mathcal{G}$.

[4]In those cases where $m = 0$, we take the conjunct $\pi(\mathbf{t})$ in the left-hand side of σ to be *true*.

Lemma 13. Let $\mathcal{G} = (\mathbf{S}, O, \Sigma)$, with $O = (\mathbf{C}, \mathbf{P}, G)$, be a relational-to-RDF temporal schema mapping, and I a source instance. Let $K_0 = (I, G)$. In a chase sequence $K_0, K_1, \dots$ for I w.r.t. $\mathcal{G}$, consider the chase step $K_{j-1} \xrightarrow{\sigma, h_j} K_j$ for an arbitrary $j > 0$. Let K be an instance such that: (1) $K \models \sigma$, and (2) there exists a homomorphism h: $K_{j-1} \to K$. Then there exists a homomorphism h': $K_j \to K$.

Proof. The GAV temporal constraint σ is of the form $\forall \mathbf{x}, \mathbf{t}\ (\varphi(\mathbf{x}, \mathbf{t}) \wedge \pi(\mathbf{t}) \to\ \psi(\mathbf{x}, \mathbf{t}))$. By definition of the chase step, there exists a relational formula homomorphism $h'_j : \varphi(\mathbf{x}, \mathbf{t}) \wedge \pi(\mathbf{t}) \to K_{j-1}$. Since composing homomorphisms yields homomorphisms, $h'_j \circ h : \varphi(\mathbf{x}, \mathbf{t}) \wedge \pi(\mathbf{t}) \to K$ is a homomorphism. In addition, since $K \models \sigma$, there exists a homomorphism $h'' : \varphi(\mathbf{x}, \mathbf{t}) \wedge \pi(\mathbf{t}) \wedge \psi(\mathbf{x}, \mathbf{t}) \to K$, such that h'' is $h'_j \circ h$, i.e., $h(h'_j(\mathbf{x})) = h''(\mathbf{x})$ and $h(h'_j(\mathbf{t})) = h''(\mathbf{t})$.

Consider now the homomorphism $h_j : \varphi(\mathbf{x}, \mathbf{t}) \wedge \pi(\mathbf{t}) \wedge \psi(\mathbf{x}, \mathbf{t}) \to K_j$. By definition of the chase step, we have that $h_j(\mathbf{x}) = h'_j(\mathbf{x})$ and $h_j(\mathbf{t}) = h'_j(\mathbf{t})$.

We define the homomorphism h' as follows: h' is the same as h when applied to each element of the vectors $h'_j(\mathbf{x})$ and $h'_j(\mathbf{t})$. Consider the vector V_{h_j} of triples obtained by applying the homomorphism h_j to the conjunction $\psi(\mathbf{x}, \mathbf{t})$, as well as the vector $V_{h''}$ of triples obtained by applying the homomorphism h'' to the same conjunction ordered in the same way. Consider an arbitrary value $u \in U \cup L \cup T_I$ that occurs in K_j but not in K_{j-1}. Then u must occur in the vector V_{h_j}, say in positions $p_1, \dots, p_r$. From $K \models \sigma$ and by the properties of the functions F_{uri} and F_{ruri}, we obtain that u also occurs in the vector $V_{h''}$ in (at least) the same positions $p_1, \dots, p_r$. By definition of the chase step, the triples in the set $h_j(\psi(\mathbf{x}, \mathbf{t}))$ are the only triples that occur in the instance K_j but not in the instance K_{j-1}. Thus, h' maps V_{h_j} onto $V_{h''}$, and maps (same as h) the rest of K_j into K. We conclude that h' is a homomorphism from the instance K_j to the instance K. □

Theorem 14. Let $\mathcal{G} = (S, O, \Sigma)$, with $O = (\mathbf{C}, \mathbf{P}, G)$, be a relational-to-RDF temporal schema mapping, and let I be a source instance. Then the target RDF instance J returned by the chase algorithm is a universal solution for I w.r.t. $\mathcal{G}$.

Proof. Let $K^{'}$ be an arbitrary solution for I w.r.t. $\mathcal{G}$; thus $K^{'}$ satisfies Σ. Observe that the identity mapping from (I, G) to $K^{'}$ is a homomorphism, and $K = (I, J)$ satisfies Σ. By repeatedly applying Lemma 13 at each chase step starting with $K_0 = (I, G)$, we obtain a homomorphism h from K to $K^{'}$. We conclude that J in K is a universal solution I w.r.t. $\mathcal{G}$. □

We call the target instance produced by the chase algorithm on I w.r.t. $\mathcal{G}$ the canonical universal solution.

Definition 21. Data Examples, **Universal Data Examples** Let $\mathcal{G} = \{\mathbf{S}, O, \Sigma\}$ be a relational-to-RDF temporal schema mapping.

- A *data example* is a pair (I, J), where I is a source instance over $\mathbf{S}$, and J is a target instance over O.
- Let (I, J) be a data example. We say that (I, J) is a *universal data example* if J is a canonical universal solution for I w.r.t. $\mathcal{G}$.
- Let E be a set of data examples. If every data example in E is a universal data example for a relational-to-RDF temporal schema mapping $\mathcal{G}$, then we say that $\mathcal{G}$ *fits* E.

Chapter 5

Active Learning of Temporal Schema Mappings

The learning framework casts schema-mapping identification as a computational learning problem aiming to identify a goal schema mapping by asking queries to various oracles that have access to the goal schema mapping. In paper [62], this problem of learning a GAV mapping is studied on the theoretical aspects, and the problem is solved by the algorithm EXACTGAV in Angluin's exact learning model [7]. In this model, schema mappings are regarded as concepts; the goal schema mapping is identified by asking *labeling* queries and *equivalence* queries. When presented with a source instance *I*, a labeling query returns the canonical universal solution *J* w.r.t. the goal schema mapping, while the equivalence query returns whether the hypothetical schema mapping is logically equivalent to the goal schema mapping or not. A labeling oracle and an equivalence oracle are utilized for answering labeling queries and equivalence queries. Building upon the theoretical result of the EXACTGAV algorithm, paper [64] developed the GAVLEARN algorithm to identify a relational-to-relational GAV schema mapping in practical scenarios with the help of conformance testing, which is a substitute of the equivalence oracle [6, 7].

The work presented in this chapter aims to identify a *goal* relational-to-RDF temporal schema mapping $\mathcal{G}$ through a learning framework that learns a relational-to-RDF temporal schema mapping $\mathcal{G}'$ that is logically equivalent to the goal schema mapping $\mathcal{G}$. To solve this problem, we consider the algorithm EXACTGAV and the algorithm GAVLEARN as our starting point and design a polynomial-time algorithm in Angluin's exact learning model [7].

In this context, according to Angluin's result [6], the labeling oracle could be implemented by a black box that has the knowledge of the behavior of the relational-to-RDF temporal schema mapping $\mathcal{G}'$. However, it is not easy to implement the equivalence oracle. For this reason, substitutes of the equivalence query have been investigated for years. Conformance testing as a substitute has been extensively used (please see the literature [67], for more details). In this approach, a finite set of membership tests will be used as the equivalence oracle.

Suppose a black box implementing the labeling oracle and performing data exchange according to some relational-to-RDF temporal schema mapping $\mathcal{G}'$, and the specification of the black box is unknown. We plan to design an algorithm that uses the black box and takes a finite set of data examples as input, constituting the conformance test during the algorithm execution. In the sequel, we will use the relational-to-RDF temporal schema mapping $\mathcal{G}'$ to refer to the black box. Consequently, our GAV learning problem could be identified as follows: Let **E** be a set of universal examples for $\mathcal{G}'$, design an algorithm that takes $\mathcal{G}'$ and **E** as input and returns a relational-to-RDF temporal schema mapping that is logically equivalent to $\mathcal{G}'$.

In this chapter, Section 5.1 provides some key definitions related to the problem of how to identify a goal relational-to-RDF temporal schema mapping through a learning framework.

In Section 5.2, we present how to produce a canonical constraint from a pair of a source instance and a target instance. In Section 5.3, we present our active learning algorithm. Then in Section 5.4, we describe a metadata generator we developed for generating benchmarks for relational-to-RDF temporal data exchange problem. Finally, in Section 5.5, we will present an experimental evaluation on our active learning algorithm and analyze the experimental results.

5.1 Symbolic Instances

This section will discuss how to represent a constraint in the form of a pair of a source instance and a target instance, called a *symbolic pair*. To achieve this, we will introduce the concept of symbolic instances. By doing so, a constraint can be regarded as a data example, allowing the chase algorithm to be applied to symbolic pairs with a set of constraints.

Definition 22 (Symbolic Instances for relational-to-RDF temporal full s-t tgds). Let C : $\varphi(\mathbf{x},\mathbf{t}) \wedge \pi(\mathbf{t}) \rightarrow \psi(\mathbf{x},\mathbf{t})$ be a relational-to-RDF temporal full s-t tgd.

The *symbolic source instance* $I_\varphi \wedge I_\pi$ of C consists of two parts with the variables $\mathbf{x}$ and $\mathbf{t}$ as non-temporal elements and temporal elements, respectively, where the first part consists of the conjuncts of $\varphi(\mathbf{x},\mathbf{t})$ as facts, (we will refer to this part as the φ-*part* of $I_\varphi \wedge I_\pi$), and the second part consists of the conjuncts of $\pi(\mathbf{t})$ as facts (we will refer to this part as the π-*part* of $I_\varphi \wedge I_\pi$), where for each conjunct $t \ \rho \ t'$ in $\pi(\mathbf{t})$, we produce a fact $R_\rho(t,t')$.

The *symbolic target instance* J_ψ of C consists of a set of tuples with the variables $\mathbf{x}$ and $\mathbf{t}$ as non-temporal elements and temporal elements, respectively, and the conjuncts of $\psi(\mathbf{x},\mathbf{t})$ as triples.

Example 10. Assume that

$$C : R_1(x_1,t_1) \wedge R_2(x_2,t_2) \wedge (t_1 \mathsf{m} t_2) \rightarrow (F_{uri}(x_1), \texttt{type}, Farms).$$

The *source symbolic instance* $I_\varphi \wedge I_\pi$ and the *target symbolic instance* J_ψ of C are as follows:

1. the φ-part of $I_\varphi \wedge I_\pi$ contains two facts $R_1(x_1,t_1)$, $R_2(x_2,t_2)$
2. the π-part of $I_\varphi \wedge I_\pi$ contains a fact $R_m(t_1,t_2)$,
3. J_ψ contains a tuple $(F_{uri}(x_1), \texttt{type}, Farms)$.

In the sequel, a relational-to-RDF temporal full s-t tgd C will often be identified as the pair $(I_\varphi \wedge I_\pi, J_\psi)$, called the *symbolic pair* of C, where $I_\varphi \wedge I_\pi$ is the symbolic source instance of C; and J_ψ is the symbolic target instance of C.

Definition 23 (Homomorphisms between two symbolic source (target) instances). Let C and C' be two relational-to-RDF temporal full s-t tgds. Let I and I' as symbolic representations of the left-hand sides of C and C', respectively, where I represents the source instance of C, and I' represents the source instance of C'.

A *homomorphism* from I to I' is a function from elements in I to elements in I', such that a) if v is a non-temporal element, then $h(v)$ is a non-temporal element; b) if t is a temporal element, then $h(t)$ is a temporal element; c) if $R(v_1,\ldots,v_n,t)$ is a fact in φ-part of I, then there exists a fact $R(h(v_1),\ldots,h(v_n),h(t))$ in φ-part of I'; d) if $R_\mathsf{p}(t,t')$ is a fact in π-part of I, then there exists a fact $R_\mathsf{p}(h(t),h(t'))$ in π-part of I'.

Example 11.

$$I = \{R_1(x_1,x_2,t_1), R_2(x_2,x_3,t_2), R_\mathsf{m}(t_1,t_2)\}$$

$$I' = \{R_1(y_1,y_2,t_1), R_2(y_2,y_3,t_2), R_3(y_3,t_3), R_{\mathsf{m}}(t_1,t_2)\}$$

We have the homomorphism $h : I \rightarrow I'$, where

$$h(x_1) = y_1,\ h(x_2) = y_2,\ h(x_3) = y_3,\ h(t_1) = t_1,\ h(t_2) = t_2.$$

I and I' are *homomorphically equivalent* if there is a homomorphism from I to I' and there is a homomorphism from I' to I. Note that homomoprhical equivalence for source symbolic instances are transitive.

In a symbolic target instance of a relational-to-RDF temporal full s-t tgd, *symbolic terms* have the following syntax:

- Every element E is a symbolic term;
- Every URI p is a symbolic term;
- If $\mathbf{T}$ is a set of symbolic terms, then $F_{uri}(\mathbf{T})$ is a symbolic term.
- If T_1, T_2, T_3 are symbolic terms, then $F_{ruri}(T_1, T_2, T_3)$ is a symbolic term.

Let J and J' be two symbolic target instances of C and C', respectively.

A *homomorphism h* from J to J' is a function from elements in J to elements in J', such that

1. if v is a non-temporal element, then $h(v)$ is a non-temporal element;
2. if t is a temporal element, then $h(t)$ is a temporal element;
3. if p is a URI, then $h(p) = p$;

4. if $\mathbf{T}$ is a set of symbolic term, then $h(F_{uri}(\mathbf{T})) = F_{uri}(h(\mathbf{T}))$;

5. if T_1, T_2, and T_3 are symbolic terms, then $h(F_{ruri}(T_1, T_2, T_3)) = F_{ruri}(h(T_1), h(T_2), h(T_3))$;

6. if $f = (s, p, o)$ is a tuple in J, where s, p, and o are symbolic terms, then J' contains a tuple $f' = (h(s), h(p), h(o))$.

Example 12. Assume that J contains exactly one tuple f and J' contains two tuples f' and f'', and

$$f = (F_{uri}(x_1), \texttt{usedOn}, F_{uri}(x_2)),$$

$$f' = (F_{uri}(y_1), \texttt{usedOn}, F_{uri}(x_1)),$$

$$f'' = (F_{uri}(x_1), \texttt{liveIn}, F_{uri}(x_2)).$$

We have the homomorphism $h : J \to J'$, where

$$h(x_1) = y_1,\ h(\texttt{usedOn}) = \texttt{usedOn},\ h(x_2) = x_1.$$

Via the definition of homomorphism between symbolic instances, we can define the homomorphism between two relational-to-RDF temporal full s-t tgds.

Definition 24 (Homomorphisms between two relational-to-RDF temporal full s-t tgds). Let C and C' be two relational-to-RDF temporal full s-t tgds. Assume that (I, J) is the symbolic pair of C and that (I', J') is the symbolic pair of C'.

A *homomorphism h* from C to C' is a function from symbolic pair in C to symbolic pair in C', such that h is a homomorphism from I to I' and h is a homomorphism from J to J'.

Assignments from a symbolic source (target) instance to a source (target) instance Let $\mathbf{S}$ be a temporal relational schema and let O be a temporal RDF graph schema. Assume a relational-to-RDF temporal full s-t tgd C. Let I be a symbolic source instance of C and let I' be a source instance over $\mathbf{S}$. Let J be a symbolic target instance of C and let J' be a target instance over O.

- An *assignment* from I to I' is a function from elements in I to the active domain of I', such that a) if v is a non-temporal element, then $h(v)$ is a constant; b) if t is a temporal element, then $h(t)$ is a time interval; c) if there is a fact $R(v_1,\ldots,v_n,t)$ in φ-part of I, then there exists a fact $R(h(v_1),\ldots,h(v_n),h(t))$ in I'; d) if there is a fact $R_{\mathsf{m}}(t_1,t_2)$ in the π-part of I, then $h(t_1)\ \mathsf{m}\ h(t_2)$.

  Example 13.

  $$I = \{R_1(x_1,x_2,t_1), R_2(x_2,x_3,t_2), R_{\mathsf{m}}(t_1,t_2)\} \text{ and } I' = \{R_1(a,b,i), R_2(b,c,i')\}$$

  We have the assignment $h : I \to I'$, where

  $$h(x_1) = a,\ h(x_2) = b,\ h(t_1) = i,\ h(x_3) = c,\ h(t_2) = i'.$$

- An *assignment* from J to J' is a function from symbolic terms in J to values occurring in J', such that a) if v is a non-temporal element, then $h(v)$ is an RDF literal; b) if t is a temporal element, then $h(t)$ is a time interval; c) if p is a URI indicating an RDF property, then $h(p) = p$; if s is a URI indicating an RDF class, then $h(s) = s$; and if o is a literal, then $h(o) = o$; d) if T is a symbolic term in the form of $F_{uri}(\mathbf{x})$, then,

  d.i) $h(T)$ is an arbitrary URI if $h(x)$ is unknown;

d.ii) otherwise, its URI value $h(F_{uri}(\mathbf{x})) = F_{uri}(h(\mathbf{x}))$;

e) if $F_{ruri}(T_1, T_2, T_3)$ is a symbolic term as a subject or an object, where T_1, T_2, and T_3 are symbolic terms appearing in J as a subject or an object, then $h(F_{ruri}(T_1, T_2, T_3))$ is the URI value of $F_{ruri}(h(T_1), h(T_2), h(T_3))$; f) if $F_{ruri}(T_1, T_2, T_3)$ is a symbolic term as a subject or an object, where T_1, T_2, and T_3 are symbolic terms, but at least one of them does not appear in J as a subject or an object, then $h(F_{ruri}(T_1, T_2, T_3))$ is an arbitrary URI; g) if $f = (s, p, o)$ is a tuple in J, where s, p, and o are symbolic terms, then there exists a triple $f' = (h(s), h(p), h(o))$ in J'.

Example 14. Consider a symbolic target instance J containing a tuple f, and a target instance J' containing three triples f', f'', and f''':

$$f = (F_{ruri}(F_{uri}(x_1), \texttt{usedOn}, F_{uri}(x_2)), \texttt{subj}, F_{uri}(x_2)),$$

$$f' = (a, \texttt{subj}, b),\ f'' = (a, \texttt{pred}, \texttt{usedOn}),\ f''' = (a, \texttt{obj}, c).$$

We have the assignment $h : J \to J'$, where

$$h(F_{ruri}(F_{uri}(x_1), \texttt{usedOn}, F_{uri}(x_2)) = a,\ h(\texttt{subj}) = \texttt{subj},\ h(F_{uri}(x_2)) = b.$$

Example 15. Consider a symbolic target instance J containing two tuples f and f', and a target instance J' containing two triples f'' and f'''.

$$f = (F_{ruri}(F_{uri}(x_1), \texttt{usedOn}, F_{uri}(x_2)), \texttt{subj}, F_{uri}(x_2)),$$

$$f' = (F_{uri}(x_1), \texttt{usedOn}, F_{uri}(x_2)),$$

$$f'' = (a, \texttt{subj}, b),\ f''' = (b, \texttt{usedOn}, c),$$

where $a = F_{ruri}(b, \texttt{usedOn}, c)$.

We have the assignment $h : J \to J'$, where

$$h(F_{uri}(x_1)) = b,\ h(\texttt{usedOn}) = \texttt{usedOn},\ h(F_{uri}(x_2)) = c,$$

$$h(F_{ruri}(F_{uri}(x_1), \texttt{usedOn}, F_{uri}(x_2)) = a,\ h(\texttt{subj}) = \texttt{subj}.$$

An assignment h from $C = (I, J)$ to a data example (I', J') is a function from the elements in (I, J) to the elements in (I', J'), such that h is an assignment from I to I' and h is an assignment from J to J'.

Example 16.

$$I = \{R_1(x_1, x_2, t_1), R_2(x_2, x_3, t_2), R_{\mathsf{m}}(t_1, t_2)\},$$

$$J = \{(F_{ruri}(F_{uri}(x_1), \texttt{usedOn}, F_{uri}(x_2)), \texttt{subj}, F_{uri}(x_2)\}$$

and

$$I' = \{R_1(a, b, [1, 3)), R_2(b, c, [3, 4))\},$$

$$J' = \{(u_a, \texttt{subj}, u_b),\ (u_a, \texttt{pred}, \mathit{usedOn}),\ (u_a, \texttt{obj}, u_c)\}.$$

We assume that

$$u_b = F_{uri}(b),$$

$$u_a = F_{uri}(F_{uri}(a), \texttt{usedOn}, F_{uri}(b)) = F_{ruri}(F_{uri}(a), \texttt{usedOn}, u_b).$$

We have the assignment $h : (I, J) \to (I', J')$, where

$$h(x_1) = a,\ h(x_2) = b,\ h(t_1) = [1, 3),\ h(x_3) = c,\ h(t_2) = [3, 4), h(F_{uri}(x_2)) = u_b,$$

$$
\begin{aligned}
&h(F_{ruri}(F_{uri}(x_1), \texttt{usedOn}, \texttt{F}_{\texttt{uri}}(\texttt{x}_2)))\\
=&F_{ruri}(F_{uri}(h(x_1)), \texttt{usedOn}, \texttt{F}_{\texttt{uri}}(\texttt{h}(\texttt{x}_2)))\\
=&F_{ruri}(F_{uri}(a), \texttt{usedOn}, \texttt{F}_{\texttt{uri}}(\texttt{b}))\\
=&u_a.
\end{aligned}
\tag{5.1}
$$

5.2 Canonical Relational-to-RDF Temporal Full S-t tgd

In the above section, we demonstrated that constraints can be represented by symbolic pairs, in such a way that they are regarded as a pair of a source instance and a target instance. Conversely, this section will discuss how to convert a source instance and a target instance into a *canonical constraints*, which will help us derive a constraint in the active learning algorithm (see Section 5.3).

Let I be a temporal source instance I over **S** and let J be a target RDF instance over O.

5.2.1 Canonical Conjunctions

Definition 25 (**Canonical Conjunction of atomic formulas for a temporal relational instance).** A *canonical conjunctive* query I_φ for I is a Boolean conjunctive query with a renaming of the temporal elements of I as temporal variables $t_1, t_2, \ldots$ and with a renaming of the non-temporal elements of I as non-temporal variables $x_1, x_2, \ldots$, and the facts of I as conjuncts, where a fact of I is an expression $R_i(a_1, \ldots, a_m, i)$ such that $(a_1, \ldots, a_m, i) \in R_i$ conjunctive query.

Definition 26 (**Canonical Conjunctions of Allen's atomic formulas for a temporal relational instance).** Let I_φ be a canonical conjunction of atomic formula for I. We say that I_π is a *canonical* *conjunction of Allen's atomic formulas for I* w.r.t. I_φ if for any pair of time intervals i and i' in I, there is an Allen's atomic formula in the form of $R_\rho(t,t')$, where the two intervals i and i' are in the Allen's relation $i\rho i'$, and i is renamed as t and i' is renamed as t' in I_φ.

We call $I_\varphi \wedge I_\pi$ the *canonical conjunctions* for the instance I.

Example 17. Assume an instance I over some temporal relational schema $\mathbf{S}$ and the facts of I are shown as follows:

$$f_1 = R_1(a,b,[1,3))$$

$$f_2 = R_1(a,c,[1,5))$$

$$f_1 = R_2(c,d,[3,5))$$

We have that

$$I_\varphi = R_1(x_1,x_2,t_1) \wedge R_1(x_1,x_3,t_2) \wedge R_2(x_3,x_4,t_3)$$

and

$$I_\varphi = t_1 \mathsf{\ s\ } t_2 \wedge t_1 \mathsf{\ m\ } t_3 \wedge t_3 \mathsf{\ f\ } t_2.$$

5.2.2 Canonical Constraints

Given a pair (I,J) of a source instance over $\mathbf{S}$ and a target instance over O, we will transform it into a relational-to-RDF temporal full s-t tgd $\sigma = (I_\varphi \wedge I_\pi, J_\psi)$ with the canonical conjunction of I as the left-hand side and a conjunction of atomic formulas over O as the right-hand side, where each atomic formula is transformed from each triple in J and one atomic formula for each triple in J, in such a way that there is an assignment s from the active domain of I to elements in $I_\varphi \wedge I_\pi$, and thus $(I,J), s \models (I_\varphi \wedge I_\pi, J_\psi)$. In order to generate such a constraint from (I,J), we analyze the challenges and design Algorithm 1 to generate such a constraint from (I,J) with the help of Algorithm 2 - Algorithm 6.

Challenges: We can easily convert the temporal relational source instance I into canonical conjunctions for I by assigning each constant a fresh variable and by assigning each time interval a fresh temporal variable. This process generates an assignment table that contains assignments for all variables and terms. However, when generating the conjunction of RDF atomic formulas for triples from J, we classify the type of values from J into three groups, and we generate terms for those values accordingly: a) for a value that is a property or a class, called keyword for simplicity, defined in Section 4.2, we construct a term (the keyword itself is a term); b) for a value that is directly transferred from I, the variable that is assigned to the value in the assignment table becomes the term of the value; c) for a value that is generated by functions F_{uri} or F_{ruri} with parameters from values in I, keywords, and other URI values generated by functions F_{uri} and F_{ruri}, we face certain challenges that need to be addressed in order to map those values into terms.

Challenge 1: Identify generating functions for terms in the right-hand side of a canonical relational-to-RDF temporal full s-t tgd

Note that the function F_{ruri} is used to generate instances of the RDF classes,

$$\{\texttt{Statement}, \texttt{TNode}, \text{ and } \texttt{Interval}\}.$$

According to the definition of temporal RDF schemas and instances, we know that i) the subjects of the triples whose predicates are `temporal`, `obj`, `pred`, or `subj` are instances of the class

`Statement`; ii) objects of the triples whose predicates are `temporal` and subjects of the triples whose predicates are `interval` are instances of the class `TNode`; iii) objects of the triples whose predicates are `interval` and subjects of the triples whose predicates are `validFor` are instances of the class `Interval`. Therefore, given a value that is generated by the two functions F_{uri} and F_{ruri}, we are able to identify if the value is generated by the function F_{ruri} or not by identifying if the above cases apply to the value. If none of the cases can be applied, then the value is generated by F_{uri}. For example, assume a triple $(a, \texttt{interval}, b)$. As case ii) applies to the value a and case iii) applies to the value b, both a and b are produced by the function F_{ruri}.

Challenge 2: After identifying the function to be used for a value a, parameters of the function should be calculated reversely with the help of the function and the value a. There are several types of parameters of the function F_{ruri}. Some parameters may come from I directly, but some parameters are generated by another function F_{ruri} or F_{uri}. The latter case makes the term of the value complicated, and this term contains functions that may contain nested functions.

i) If it is F_{uri}, parameters are directly generated through reversing function $F_{uri}^{-1}(a)$.

ii) If it is F_{ruri}, the case becomes complicated for the parameters of the function because some parameters of the function are directly from I, some parameters are generated by F_{uri}, some other parameters are generated from F_{ruri}. For the last two cases, the term generated for the value a should be a function containing another nested function as its parameters. In general, those parameters also appear in other triples as subjects or objects. Therefore, it will be easier if first identify the terms of those parameters of the function F_{ruri}. To this end, for the value a, which will be a term in the form of $F_{ruri}(\cdot)$, we first generate atomic formulas for other triples with the parameters as the subjects or the objects. Hence, for the triples that contain values that will be converted into terms with nested functions, we will handle them later; for the triples whose values could be used as parameters of the functions F_{ruri} and F_{uri}, we handle them earlier. As a result, it matters in which order we will transfer triples into RDF atomic formulas. After careful observation, we assign each property appearing in O a priority. Specifically, a) properties in **P** have the first priority; b) properties in $Snd = \{subj, obj, pred\}$ have the second priority; c) properties in $Thrd = \{temporal, interval, validFor\}$ have the third priority; d) properties in $Lst = \{sc, dom, ran, sp, type\}$ have the least priorities, where predicates in the sets Snd, $Thrd$, and Lst are listed in descending order of priority. Hence, we are able to classify triples in J into four sets $FstTriples$, $SndTriples$, $TrdTriples$, and $LstTriples$, where $FstTriples$ only contains the triples whose predicates are in **P**; the sets $SndTriples$, $TrdTriples$, and $LstTriples$ only contain triples whose predicates are in Snd, $Thrd$, and Lst, respectively. Sort the triples in $SndTriples$, $TrdTriples$, and $LstTriples$, respectively, according to the priorities of the properties in each set.

We first calculate the RDF atomic formulas for triples whose predicate is in **P** (see Algorithm 3), then for the triples in the order of $SndTriples$ (see Algorithm 4), $TrdTriples$ (see Algorithm 5), and $LstTriples$ (see Algorithm 6). Moreover, Algorithm 4 - Algorithm 6 carefully provide the details of how to identify generating functions for terms in the right-hand side of a canonical relational-to-RDF temporal full s-t tgd and how to convert triples in a target instance into terms in the right-hand side of a canonical relational-to-RDF temporal full s-t tgd.

For example, let $f = DrugUsage(a, b, c, [1, 2))$ be a fact in I. Let $(c_{uri}, \texttt{usedOn}, b_{uri})$ and $(d_{ruri}, \texttt{subj}, c_{uri})$ be two triples in J. The fact f is transferred into a relational atom $DrugUsage(x_1, x_2, x_3, t)$ and the assignment table is $a : x_1, b : x_2, c : x_3, [1, 2) : t$.

First, we transfer the triple $(c_{uri}, \texttt{usedOn}, b_{uri})$ into an atomic formula since, as mentioned above, we must calculate the RDF atomic formula for the triple $(c_{uri}, \texttt{usedOn}, b_{uri})$ before doing that for $(d_{ruri}, \texttt{subj}, c_{uri})$. We choose the function F_{uri} for c_{uri} and b_{uri} since there is no assignment for c_{uri} and b_{uri} in the assignment table and F_{ruri} cannot be selected based on the predicate of the triple. Then we calculate the parameters by reversing the function F_{uri}, such that we have $F_{uri}^{-1}(c_{uri}) = c$. Hence, c_{uri} is transferred into the term $F_{uri}(x_3)$ since c is assigned to the variable x_3 in the assignment table. As a result, $(c_{uri}, \texttt{usedOn}, b_{uri})$ is converted into the atomic formula $(F_{uri}(x_3), \texttt{usedOn}, F_{uri}(x_2))$. Next, we can proceed to handle the triple $(d_{ruri}, \texttt{subj}, c_{uri})$. For the value d_{ruri}, the function F_{ruri} will be applied because the predicate is `subj`. Then we calculate the parameter values of the function F_{ruri} by reversing the function $F_{ruri}^{-1}(d_{ruri})$ which consists of three parameters values $\{c_{uri}$, `usedOn`, $b_{uri}\}$. Since terms translated from c_{uri} and b_{uri} are $F_{ruri}($ $F_{uri}(x_3)$, `usedOn`, $F_{uri}(x_2))$ and $F_{uri}(x_3)$, the atomic formula is $(F_{ruri}($ $F_{uri}(x_3)$, `usedOn`, $F_{uri}(x_2))$, `subj`, $F_{uri}(x_3))$.

Algorithm 1 Convert a data example to a canonical relational-to-RDF temporal full s-t tgd

```
Require: $(I,J)$ - a universal data example for $\mathcal{G}$.
Ensure: a relational-to-RDF temporal full s-t tgd $\sigma$ that is satisfied by $(I,J)$;
        a mapping *map* from a triple in $J$ to an RDF atomic formula
 1: Calculate $I_\varphi$ and $I_\pi$ by using Algorithm 2
 2: Initialize a mapping *map* from a triple in $J$ to an RDF atomic formula to be empty
 3: Initialize a vector $J_\psi$ to be empty
 4: $J'_\psi, map' \leftarrow$ result generated by Algorithm 3.
 5: $J_\psi \leftarrow J_\psi \cup J'_\psi$
 6: $map \leftarrow map \cup map'$
 7: $J''_\psi, map'' \leftarrow$ result generated by Algorithm 4.
 8: $J_\psi \leftarrow J_\psi \cup J''_\psi$
 9: $map \leftarrow map \cup map''$
10: $J'''_\psi, map''' \leftarrow$ result generated by Algorithm 5.
11: $J_\psi \leftarrow J_\psi \cup J'''_\psi$
12: $map \leftarrow map \cup map'''$
13: $J''''_\psi, map'''' \leftarrow$ result generated by Algorithm 6.
14: $J_\psi \leftarrow J_\psi \cup J''''_\psi$
15: $map \leftarrow map \cup map''''$
16: **return** $(I_\varphi \wedge I_\pi, J_\psi)$ and *map*
```

We call the constraint $C = (I_\varphi \wedge I_\pi, J_\psi)$ generated by Algorithm 1 the *canonical constraint* for (I,J), where we first use Algorithm 2 to convert the source instance I into a canonical conjunction; and then, we use Algorithm 3 - Algorithm 6 to convert the target instance J into a conjunction of RDF atomic formulas. Note that the canonical relational-to-RDF temporal full s-t tgd has the following properties:

1. $I_{\varphi} \wedge I_{\pi}$ is a canonical conjunction for the instance I;

2. J_{ψ} is a conjunction of atomic formulas over O.

3. each variable in J_{ψ} comes from $I_{\varphi} \wedge I_{\pi}$, and each atomic formula in J_{ψ} corresponds to a triple in J;

4. for each triple (s, p, o) with a time interval i such that J contains its temporal annotation, if the triple is transformed to the atomic formula (T_1, p, T_2) in J_{ψ}, then J_{ψ} contains the co-occurring set of (T_1, p, T_2), each of which corresponds to a triple in the temporal annotation.

If C is a relational-to-RDF temporal GAV constraint, then we say that C is a *canonical relational-to-RDF temporal GAV constraint.*

Algorithm 2 Convert a source instance to a canonical conjunction

```
Require: $(I, J)$ - a universal data example for $\mathcal{G}$.
Ensure: a canonical conjunction $I_{\varphi} \wedge I_{\pi}$ of $I$
1: $I_{\varphi} \leftarrow$ the canonical conjunction of atomic formulas for $I$
2: $I_{\pi} \leftarrow$ the canonical conjunction of Allen's atomic formulas for $I$ w.r.t. $I_{\varphi}$
3: return $I_{\varphi} \wedge I_{\pi}$
```

Algorithm 3 Transfer a data example to a canonical relational-to-RDF temporal full s-t tgd for triples of the first priority.

```
Require: $I_\varphi \wedge I_\pi$ - a canonical conjunction of a relational instance $I$;
         $FstTriples$ - subset of $J$, where it contains all triples whose predicates in $\mathbf{P}$.
Ensure: a relational-to-RDF temporal full s-t tgd $\sigma$ that is satisfied by $(I,J)$;
 1: while for each triple $f = (s,p,o)$ in $FstTriples$ do
 2:     create a conjunct $(T_1, p, T_2)$
 3:     $T_1 \leftarrow F_{uri}(F_m(F_{uri}^{-1}(s)))$
 4:     if $o$ is an RDF URIs then
 5:         $T_2 \leftarrow F_{uri}(F_m(F_{uri}^{-1}(o)))$
 6:     else
 7:         $T_2 \leftarrow F_m(o)$
 8:     end if
 9:     $map[(s,p,o)] = (T_1, p, T_2)$
10:     Add $(T_1, p, T_2)$ into $J_\psi$
11: end while
12: return $map$ and $J_\psi$
```

Algorithm 4 Convert a data example to a canonical relational-to-RDF temporal full s-t tgd- for triples of the second priority.

```
Require: $I_\varphi \wedge I_\pi$ - a canonical conjunction of a relational instance $I$
        $SndTriples$ - subset of $J$, where it contains all triples whose predicates in $Snd$, and the triples
    are sorted according to the order of their predicates appearing in $Snd$.
Ensure: a relational-to-RDF temporal full s-t tgd $\sigma$ that is satisfied by $(I,J)$;
 1: while for each triple $f = (s,p,o)$ in $SndTriples$ do
 2:     create a conjunct $(T_1, p, T_2)$
 3:     if $p$ is $pred$ then
 4:         $s' \leftarrow$ find the object of the tuple that has $s$ as subject and has $subj$ as predicate
 5:         $T_s \leftarrow F_{uri}(F_m(F_{uri}^{-1}(s')))$
 6:         $o' \leftarrow$ find the object of the tuple that has $s$ as subject and has $obj$ as predicate
 7:         if $o'$ is an RDF Literal, then $T_o$ is $F_m(o')$; and if $o'$ is an RDF URI, then $T_o$ is $F_{uri}(F_m(F_{uri}^{-1}(o')))$
 8:         $T_1 \leftarrow F_{ruri}(T_s, o, T_o)$
 9:         $T_2 \leftarrow o$
10:     else if $p$ is $subj$ then
11:         $p' \leftarrow$ find the object of the tuple that has $s$ as subject and has $pred$ as predicate.
12:         $o' \leftarrow$ find the object of the tuple that has $s$ as subject and has $obj$ as predicate
13:         if $o'$ is an RDF Literal, then $T_o$ is $F_m(o')$; and if $o'$ is an RDF URI, then $T_o$ is $F_{uri}(F_m(F_{uri}^{-1}(o')))$
14:         $T_2 \leftarrow F_{uri}F_m(F_{uri}^{-1}(o))$
15:         $T_1 \leftarrow F_{ruri}(T_2, p', T_o)$
16:     else
17:         $p' \leftarrow$ find the object of the tuple that has $s$ as subject and has $pred$ as predicate.
```

Algorithm 4 Convert a data example to a canonical relational-to-RDF temporal full s-t tgd- for triples of the second priority (continued).

```
18:         $s' \leftarrow$ find the object of the tuple that has $s$ as subject and has $subj$ as predicate
19:         $T_s \leftarrow F_{uri}(F_m(F_{uri}^{-1}(s')))$
20:         if $o$ is an RDF Literal $T_2$ is $F_m(o)$; and if $o$ is an RDF URI, then $T_2$ is $F_{uri}(F_m(F_{uri}^{-1}(o)))$
21:         $T_1 \leftarrow F_{ruri}(T_s, p', T_2)$
22:     **end if**
23: **end while**
24: $map[(s,p,o)] = (T_1, p, T_2)$
25: Add $(T_1, p, T_2)$ into $J_\psi$
26: **return** $map$ and $J_\psi$
```

Algorithm 5 Convert a data example to a canonical relational-to-RDF temporal full s-t tgd-for the triples of the third priority.

```
Require: $I_\varphi \wedge I_\pi$ - a canonical conjunction of a relational instance $I$;
        $ThrdTriples$ - a subset of $J$, where it contains all triples whose predicates in $Thrd$, and the
    triples are sorted according to the order of their predicates appearing in $Thrd$;
        $SndTriples$ - a subset of $J$, where it contains all triples whose predicates in $Snd$, and the triples
    are sorted according to the order of their predicates appearing in $Snd$;
        $map$ - a map from triples in $SndTriples$ to RDF atomic formulas
Ensure: a relational-to-RDF temporal full s-t tgd $\sigma$ that is satisfied by $(I,J)$;
 1: while for each triple $f = (s,p,o)$ in $ThrdTriples$ do
 2:     create a conjunct $(T_1, p, T_2)$
 3:     if $p$ is $temporal$ then
 4:         $(s', p', o') \leftarrow$ find a triple from $SndTriples$ that has $s$ as subject and has $subj$ as predicate
 5:         $o'' \leftarrow$ find the object of the triple from $ThrdTriples$ that has $o$ as subject and has $interval$ as
    predicate
 6:         $i \leftarrow$ find the object of the triple from $ThrdTriples$ that has $o''$ as subject and has $validFor$ as
    predicate
 7:         $T_1 \leftarrow$ subject of $map[(s', p', o')]$
 8:         $T_2 \leftarrow F_{ruri}(T_1, temporal, F_m(i))$
 9:     end if
```

Algorithm 5 Convert a data example to a canonical relational-to-RDF temporal full s-t tgd-for the triples of the third priority (continued).

```
10:     if $p$ is $interval$ then
11:         $(s', p', o') \leftarrow$ find a triple that has $s$ as object and has $temporal$ as predicate
12:         $i \leftarrow$ find the object of the triple that has $o'$ as subject, has $validFor$ as predicate
13:         $T_1 \leftarrow$ the object of $map[(s', p', o')]$
14:         $T_2 \leftarrow F_{ruri}(T_1, interval, F_m(i))$
15:     end if
16:     if $p$ is $validFor$ then
17:         $(s', p', o') \leftarrow$ find a triple that has $s$ as object and has $interval$ as predicate
18:         $T_1 \leftarrow$ the object of $map[(s', p', o')]$
19:         $T_2 \leftarrow i$
20:     end if
21:     $map[(s, p, o)] = (T_1, p, T_2)$
22:     Add $(T_1, p, T_2)$ into $J_\psi$
23: end while
24: return $map$ and $J_\psi$
```

Algorithm 6 Convert a data example to a canonical relational-to-RDF temporal full s-t tgd-least priority triples.

```
Require: $I_\varphi \wedge I_\pi$ - a canonical conjunction of a relational instance $I$;
    $LstTriples$ - a subset of $J$, where it contains all triples whose predicates in $Lst$, and the triples
  are sorted according to the order of their predicates appearing in $Lst$.
Ensure: a relational-to-RDF temporal full s-t tgd $\sigma$ that is satisfied by $(I,J)$;
 1: while for each triple $f = (s,p,o)$ in $LstTriples$ do
 2:     create a conjunct $(T_1,p,T_2)$
 3:     if $p$ is $type$ then
 4:         create a conjunct $(T_1,p,T_2)$
 5:         $T_2 \leftarrow o$
 6:         if $o$ is $timeInterval$ then $T_1 \leftarrow F_m(s)$
 7:         else if datatype of $s$ is a string literal then $T_1 \leftarrow F_m(s)$
 8:         else if $o$ is $Statement$ then
 9:             $(s',p',o') \leftarrow$ find a triple that has $s$ as subject and has $subj$ as predicate
10:             $T_1 \leftarrow$ subject of $map[(s',p',o')]$
11:         else if $o$ is $Interval$ then // not instance of
12:             $(s',p',o') \leftarrow$ find a triple that has $s$ as subject and has $interval$ as predicate
13:             $T_1 \leftarrow$ subject of $map[(s',p',o')]$
```

Algorithm 6 Convert a data example to a canonical relational-to-RDF temporal full s-t tgd-least priority triples (continued).

```
14:         else if $o$ is $TNode$ then
15:             $(s', p', o') \leftarrow$ find a triple that has $s$ as subject and has $validFor$ as predicate
16:             $T_1 \leftarrow$ subject of $map[(s', p', o')]$
17:         else
18:             $T_1 \leftarrow F_{uri}(F_m(F_{uri}^{-1}(s))$
19:         end if
20:     else
21:         $T_1 \leftarrow s$
22:         $T_2 \leftarrow o$
23:     end if
24:     $map[(s, p, o)] = (T_1, p, T_2)$
25:     Add $(T_1, p, T_2)$ into $J_\Psi$
26: end while
27: return $map$ and $J_\Psi$
```

5.3 Active Learning Algorithm

Let $\mathcal{G} = (\mathbf{S}, O, \Sigma)$ be a goal schema mapping and let E be a set of universal data examples of $\mathcal{G}$. This section introduces an active learning algorithm (see Algorithm 7) based on a given relational-to-RDF temporal schema mapping $\mathcal{G}$ as a black box and E.

5.3.1 The Learning Algorithm

In this section, we will state and prove several results needed for the Algorithm 7.

Lemma 15. Assume a relational-to-RDF temporal schema mapping $\mathcal{G} = (\mathbf{S}, \mathcal{O}, \Sigma)$. Let I be a source instance over $\mathbf{S}$ and let J be a target RDF instance over $\mathcal{O}$, which contains a set $\mathbf{f}$ of triples. If $\mathbf{f}$ contains exactly one triple (s, p, o) where p is a non-temporal property that does not admit temporal annotation, or if $\mathbf{f}$ consists of a triple (s, p, o) with p admitting temporal annotations and a temporal annotation of (s, p, o) and a time interval, then we have that the canonical constraint $(I_\varphi \wedge I_\pi, \mathbf{f}_\psi)$ of $(I, \mathbf{f})$ is a relational-to-RDF temporal GAV constraint.

Proof. According to the definition of relational-to-RDF temporal GAV constraint, there are two cases for the right-hand side of a constraint. The first case is that the right-hand side is an atomic formula over $\mathcal{O}$ where the predicate of the atomic formula does not admit temporal annotations. Therefore, if $\mathbf{f}$ contains exactly one triple and if its predicate does not admit temporal annotations, then the predicate of the atomic formula does not admit temporal annotation since canonical constraints preserve the predicate of the original triple according to the properties that canonical constraints hold. In addition, $\mathbf{f}_\psi$ contains exactly one atomic formula transformed from the only triple from $\mathbf{f}$. Hence, $(I_\varphi \wedge I_\pi, \mathbf{f}_\psi)$ is a relational-to-RDF temporal GAV constraint.

The second case is that the right-hand side is a conjunction of atomic formulas which consists of an atomic formula (T_1, p, T_2) with $p \in \mathbf{P}$ and all atomic formulas in the co-occurring set of (T_1, p, T_2) if p admits temporal annotation in $\mathcal{O}$. Now, we assume that $\mathbf{f}$ consist of a triple (s, p, o) with p admits temporal annotations and a temporal annotation of the triple and a time interval as mentioned in the lemma. Then according to the property that canonical constraints hold, $\mathbf{f}_\psi$ consists of the atomic formula (T_1, p, T_2) transformed from (s, p, o) and its co-occuring set. Thus the canonical constraint of $(I, \mathbf{f})$ is a GAV constraint according to the property of canonical constraint. $\square$

Lemma 16. Assume a relational-to-RDF temporal schema mapping $\mathcal{G} = (\mathbf{S}, O, \Sigma)$. Let I be a source instance over $\mathbf{S}$ and let J be a target RDF instance over O, which contains a set $\mathbf{f}$ of triples, where $\mathbf{f}$ contains exactly one triple (s, p, o) where p is a non-temporal property that does not admit temporal annotation; or $\mathbf{f}$ consists of a triple (s, p, o) with p is a non-temporal property admitting temporal annotations and a temporal annotation of (s, p, o) and a time interval. For all canonical constraints $C = (I_\varphi \wedge I_\pi, \mathbf{f}_\psi)$ of $(I, \mathbf{f})$, where $I_\varphi \wedge I_\pi$ is the canonical conjunctions for I, the following statements are equivalent:

(1) $\mathcal{G}$ logically implies C;

(2) there is a homomorphism from C' to C for some C' in Σ;

(3) a canonical universal solution for I w.r.t. Σ contains $\mathbf{f}$.

Proof. The proof is similar to that of the Lemma 3.1 in paper [62]. We will prove our lemma in a round-robin fashion: (1) implies (3), which implies (2), which, in turn, implies (1).

Note that C is canonical relational-to-RDF temporal GAV constraint according to Lemma 16.

- The implication from (1) to (3): Let $\textit{can-sol}_{\mathcal{G}}(I)$ be a canonical universal solution for I w.r.t. $\mathcal{G}$. It follows that $(I, \textit{can-sol}_{\mathcal{G}}(I))$ satisfies all constraints in $\mathcal{G}$. By the statement of (1), we know that $(I, \mathit{can\text{-}sol}_{\mathcal{G}}(I))$ satisfies C, which means the triples $\mathbf{f}$ exist in $\mathit{can\text{-}sol}_{\mathcal{G}}(I)$. This completes the proof of the implication from (1) to (3).

- The implication from (3) to (2): If $\mathbf{f}$, as stated in (3), occurs in a canonical universal solution for I w.r.t. $\mathcal{G}$ (denoted by $\mathit{can\text{-}sol}_{\mathcal{G}}(I)$), then according to the definition of relational-to-RDF chase algorithm, there exists a GAV constraint $C' = (I'_{\varphi} \wedge I'_{\pi}, \mathbf{f}'_{\psi})$ ($I'_{\varphi} \wedge I'_{\pi}$ and $\mathbf{f}'_{\psi}$ are the source symbolic instance of C' and the target symbolic instance of C', respectively) in $\mathcal{G}$, such that there exists an assignment from C' to I. It follows that there is a homomorphism h from $I'_{\varphi} \wedge I'_{\pi}$ to $I_{\varphi} \wedge I_{\pi}$. In addition, C is a GAV constraint, and that means $\mathbf{f}$ either contains exactly one triple or it consists of a triple and one temporal annotation for this triple. Therefore, all triples in $\mathbf{f}$ must be generated by the same GAV constraint C' in Σ according to the definition of GAV constraint and the chase algorithm. It implies that each triple in $\mathbf{f}$ is generated with one of atomic formulas in $\mathbf{f}'_{\psi}$. This is because $\mathbf{f}'_{\psi}$ consists of one atomic formula (T_1, p, T_2) if p does not admit temporal annotation, otherwise it consists of (T_1, p, T_2) and it co-occurring set, which corresponds to $\mathbf{f}$ consisting of exactly one triple if the predicate of it does not admit temporal annotation, or it consists of a triple and one temporal annotation for this triple, and thus the size of $\mathbf{f}'_{\psi}$ equals to that of $\mathbf{f}_{\psi}$. Hence, h is the homomorphism from $\mathbf{f}'_{\psi}$ to $\mathbf{f}_{\psi}$. Thus, by the definition of homomorphism between two constraints, there exists a homomorphism from $C' = (I'_{\varphi} \wedge I'_{\pi}, \mathbf{f}_{\psi})$ to $C = (I_{\varphi} \wedge I_{\pi}, \mathbf{f}_{\psi})$. This completes the proof of the implication from (3) to (2).

- The implication from (2) to (1): Let (I'', J'') be any pair consisting of a source instance and a target instance, satisfying all constraints in $\mathcal{G}$. We will show that (I'', J'') satisfies

C, i.e., if $I_\varphi \wedge I_\pi$ is satisfied by I'', then $\mathbf{f}_\psi$ is satisfied by J''. Assume that there is an assignment g from the symbolic source instance $I_\varphi \wedge I_\pi$ of C to the source instance I''. Let $C' = (I'_\varphi \wedge I'_\pi, \mathbf{f'}_\psi)$ and let h be a homomorphism from the constraint C to the constraint C' ($h : I_\varphi \wedge I_\pi \rightarrow I'_\varphi \wedge I'_\pi$ and $h : \mathbf{f}_\psi \rightarrow \mathbf{f'}_\psi$). It follows that there is a composition assignment $h \circ g$ from $I_\varphi \wedge I_\pi$ to I''. Since (I'', J'') satisfies $I'_\varphi \wedge I'_\pi$ (it is easy to prove that composing a homomorphism and an assignment is an assignment), and thus, (I'', J'') satisfies C, we have that the assignment $h \circ g : \mathbf{f}_\psi \rightarrow J''$. Therefore, we have that g is the assignment from $\mathbf{f}_\psi$ to J'', which follows from the fact that h is the homomophism from $\mathbf{f}_\psi$ to $\mathbf{f'}_\psi$. This completes the proof of the implication from (2) to (1).

$\square$

The notion of critically sound Let $\mathcal{G}$ be a relational-to-RDF temporal schema mapping. A relational-to-RDF temporal GAV constraint is *sound* with respect to $\mathcal{G}$ if $\mathcal{G}$ logically implies $C = (I_\varphi \wedge I_\pi, \mathbf{f}_\psi)$, that is, for every data example (I, J) that satisfies $\mathcal{G}$, we have that (I, J) also satisfies C. Given a conjunct $R(\mathbf{x}, t)$ in I_φ, we call every conjunct in I_π that contains the temporal variable t *the related conjuncts* of $R(\mathbf{x}, t)$. A relational-to-RDF temporal GAV constraint is *critically sound* with respect to $\mathcal{G}$ if i) C is sound with respect to $\mathcal{G}$; and ii) for every relational-to-RDF temporal GAV constraint C' obtained either by removing one of the conjuncts of I_φ and its related conjuncts in I_π or by removing one of the conjuncts of I_π, we have that $\mathcal{G}$ does not logically imply C'.

Definition 27 (Direct Product). Let $\mathbf{S}$ be a temporal relational schema and let I and I' are two instances over $\mathbf{S}$. Assume that $I_\varphi \wedge I_\pi$ and $I'_\varphi \wedge I'_\pi$ are canonical conjunctions of I and I'

respectively. *The direct-product* $I_\varphi \wedge I_\pi \times I'_\varphi \wedge I'_\pi$ of $I_\varphi \wedge I_\pi$ and $I'_\varphi \wedge I'_\pi$ is the a conjunction of atomic formulas with a pair $< x, x' >$ as each variable. $I_\varphi \wedge I_\pi \times I'_\varphi \wedge I'_\pi$ consists of two parts, $I_\varphi \times I'_\varphi$ and $I_\pi \times I'_\pi$, where the part $I_\varphi \times I'_\varphi$ contains all atoms $R(< x_1, x'_1 >, \ldots, < x_k, x'_k >)$ (with $k = arity(R)$) such that $R(x_1, \ldots, x_k)$ is in I_φ and $R(x'_1, \ldots, x'_k)$ is in I'_φ; and the part $I_\varphi \times I'_\varphi$ contains all atoms $R_\rho(< t_1, t'_1 >, < t_2, t'_2 >)$ such that a) both $< t_1, t'_1 >$ and $< t_2, t'_2 >$ are in $I_\varphi \times I'_\varphi$; b) $R_\rho(t_1, t_2)$ is in I_π and $R_\rho(t'_1, t'_2)$ is in I'_π.

Let O be an temporal RDF graph schema and let J and J' are two RDF instances over O. Assume that J_ψ and J'_ψ are canonical conjunctions of J and J' respectively. The *direct-product* $J_\psi \times J'_\psi$ of J_ψ and J'_ψ is the a conjunction of RDF atomic formulas with a pair $< T, T' >$ as each term. I_d contains all RDF atoms $(< T_1, T'_1 >, p, < T_2, T'_2 >)$ such that (T_1, p, T_2) is in J_ψ and (T'_1, p, T'_2) is in J'_ψ.

Direct-product of two relational-to-RDF temporal full s-t tgds Let $I_\varphi \wedge I_\pi \to J_\psi$ and $I'_\varphi \wedge I'_\pi \to J'_\psi$ are two relational-to-RDF temporal GAV constraints. The direct-product of the two relational-to-RDF temporal GAV constraints σ is $I_\varphi \wedge I_\pi \times I'_\varphi \wedge I'_\pi \to J_\psi \times J'_\psi$.

Let (T_1, p, T_2) be an RDF atomic formula, where T_1 is an RDF term and T_2 is an RDF term, and T_1 or T_2 is either a term with some variables in functions or just a variable. In the following, we will use $T_1(x_1, \ldots, x_n, t)$ to indicate that the variables that involved in term T_1 are $x_1, \ldots, x_n, t$, and use $T_2(x'_1, \ldots, x'_m, t)$ to indicate that variables involved in the term T_2 are $x'_1, \ldots, x'_m, t$. By the definition of relational-to-RDF temporal GAV constraint, we observed that T_1 and T_2 contain the same temporal variable if both of them contain a temporal variable.

Lemma 17. Let C, C_1, and C_2 be relational-to-RDF temporal GAV constraints with homomorphisms $h_1 : C \to C_1$ and $h_2 : C \to C_2$. Then $C_1 \times C_2$ is well defined, and it is a GAV constraint; moreover, $C \to C_1 \times C_2$; $C_1 \times C_2 \to C_1$ and $C_1 \times C_2 \to C_2$.

Proof. To prove the lemma, we will consider two cases: 1) the right-hand side of C contains exactly one atomic formula; 2) the right-hand side of C consists of an atomic formula $(T_1(x_1,\dots,x_n,t), p, T_2(x'_1,\dots,x'_m,t))$ with p admits temporal annotations and all atomic formulas in its co-occurring set, where $(T(x_1,\dots,x_n,t)$ indicates a term containing one or more variables in $\{T(x_1,\dots,x_n,t\}$.

For the first case, let $(T_1(x_1,\dots,x_n,t), p, T_2(x'_1,\dots,x'_m,t))$ be the right-hand side of C. It follows that a) p does not admit temporal annotations; b) the right-hand side of C_1 is $(h_1(T_1), p, h_1(T_2))$; c) and the right-hand side of C_2 is $(h_2(T_1), p, h_2(T_2))$. Thus, the right-hand side of $C_1 \times C_2$ is

$$\big(T_1(<h_1(x_1),h_2(x_1)>,\dots,<h_1(x_n),h_2(x_n)>,<h_1(t),h_2(t)>),\ p,$$
$$T_2(<h_1(x'_1),h_2(x'_1)>,\dots,<h_1(x'_m),h_2(x'_m)>,<h_1(t),h_2(t)>)\big).$$

It is easy to identify that $C_1 \times C_2$ is a GAV constraint since both C_1 and C_2 are GAV constraints. It also suffices to show that each pair of variables $<h_1(x_i),h_1(x_i)>$ $(<h_2(x'_i),h_2(x'_i))>$ or $<h_1(t),h_2(t)>$) occurs in the left-hand side of $C_1 \times C_2$. This is indeed the case: since C is well defined, x_i, x'_i, or t occurs in some source atom, and applying the homomorphisms h_1 and h_2 to this atom yields an atom that belongs to $C_1 \times C_2$.

For the second case, let the conjunction of $(T_1(x_1,\dots,x_n,t), p, T_2(x'_1,\dots,x'_m,t))$ and atomic formulas in its co-occuring set T_t^{FRMLA} be the right-hand side of C. It follows that a) p admits temporal annotations; b) the right-hand side of C_1 consists of the atomic formula

$(h_1(T_1), p, h_1(T_2))$ and the set $h_1(T_t^{FRMLA})$ of atomic formulas generated by applying h_1 to every atomic formulas in T_t^{FRMLA}; c) and the right-hand side of C_2 consists of the atomic formula $(h_2(T_1), p, h_2(T_2))$ and the set $h_2(T_t^{FRMLA})$ of atomic formulas generated by applying h_2 to every atomic formulas in T_t^{FRMLA}. Thus, the right-hand side of $C_1 \times C_2$ is $(T_1(< h_1(x_1), h_2(x_1) >, \ldots, < h_1(x_n), h_2(x_n) >, < h_1(t), h_2(t) >), p, T_2(< h_1(x'_1), h_2(x'_1) >, \ldots, < h_1(x'_m), h_2(x'_m) >, < h_1(t), h_2(t) >))$ and the set of atomic formulas generated by applying h_1 and h_2 to each variable x, such that the variable x becomes $< h_1(x), h_2(x) >$. It is easy to identify that $C_1 \times C_2$ is a GAV constraint since both C_1 and C_2 are GAV constraint. It also suffices to show that each pair of variables $< h_1(x_i), h_1(x_i) > (< h_2(x'_i), h_2(x'_i)) >$ or $< h_1(t), h_2(t) >$) occurs in the left-hand side of $C_1 \times C_2$. This is indeed the case: since C is well defined, x_i, x'_i, or t occurs in some source atom, and applying the homomorphisms h_1 and h_2 to this atom yields an atom that belongs to $C_1 \times C_2$. $\square$

Algorithm 7 The relational-to-RDF temporal schema mapping learning algorithm

Require: $\mathcal{G}$- goal mapping (as a labeling oracle); E - a set of universal examples for $\mathcal{G}$.

Ensure: a schema mapping $\mathcal{H}$ that fit E.

```
1: $\mathcal{H} \leftarrow \emptyset$
2: while $True$ do
3:     if each $(I,J) \in E$ is canonical universal for $\mathcal{H}$ then
4:         return $\mathcal{H}$
5:     end if
6:     Choose an $(I,J) \in E$ such that $J \neq can\text{-}sol_{\mathcal{H}}(I)$
7:     $f \leftarrow$ choose a fact $f \in J/can\text{-}sol_{\mathcal{H}}(I)$
8:     Calculate canonical relational-to-RDF temporal full s-t tgd $(I_\varphi \wedge I_\pi, J_\psi)$ for $(I,J)$ by
   Algorithm 1
9:     if $J$ contains $f$ and its temporal annotation, collect all the triples which are part of the
   temporal annotation of $f$.
10: $\mathbf{f} \leftarrow f \cup$ temporal annotation of $f$
11:     Obtain $(I_\varphi \wedge I_\pi, \mathbf{f}_\psi)$ for $(I, \mathbf{f})$ from $(I_\varphi \wedge I_\pi, J_\psi)$
12:     if $\mathcal{G}$ logically implies $(I_\varphi \wedge I_\pi, \mathbf{f}_\psi) \times C$ for some $C \in \mathcal{H}$ then
13:         Choose $C \in \mathcal{H}$ such that $\mathcal{G}$ logically implies $(I_\varphi \wedge I_\pi, \mathbf{f}_\psi) \times C$
14:         $\mathcal{H} \leftarrow (\mathcal{H}/\{C\}) \cup \{Crit_{\mathcal{G}}((I_\varphi \wedge I_\pi, \mathbf{f}_\psi) \times C)\}$
15:     else
16:         $\mathcal{H} \leftarrow \mathcal{H} \cup \{Crit_{\mathcal{G}}(I_\varphi \wedge I_\pi, \mathbf{f}_\psi)\}$
17:     end if
18: end while
```

Challenge: Calculation of critically sound constraints Given a constraint as shown in line 13 of Algorithm 7, we need to generate a constraint that is critically sound w.r.t. $\mathcal{G}$. In GAVLEARN, we only need to remove each relational atom, and check if $\mathcal{G}$ logically implies the new constraint. Different from GAVLEARN, here, the constraint also contains conjunctions of Allen's relations; therefore, identifying critical Allen's relations adds complexity to the algorithm because the size of the conjunction of Allen's relation usually is the polynomial of the size of the conjunction of relational atoms.

Given a constraint $(I_\varphi \wedge I_\pi, \mathbf{f}_\phi)$, we first remove each Allen's relation until we find all critical Allen's relations. Then, we try to remove each relational atom and its corresponding Allen's relations that involve variables coming from the removed relational atom.

5.3.2 Soundness and Correctness of the Active Learning Algorithm

Lemma 18. Let $\mathcal{G}$ be a relational-to-RDF temporal schema mapping and let E be a finite set of universal data examples w.r.t. $\mathcal{G}$. Recall that the labeling oracle $LABEL_\mathcal{G}$ takes as input a source instance I and returns the target instance $can\text{-}sol_\mathcal{G}(I)$. If $C = (I_\varphi \wedge I_\pi, \mathbf{f}_\psi)$ is a relational-to-RDF temporal GAV constraint that is logically implied by $\mathcal{G}$, then, with access to $LABEL_\mathcal{G}$ and the set E of data examples, we can construct in polynomial time a relational-to-RDF temporal GAV constraint $Crit_\mathcal{G}(C) = (I'_\varphi \wedge I'_\pi, \mathbf{f}_\psi)$, where $I'_\varphi \wedge I'_\pi$ is the canonical conjunction for an instance I' and $I' \subseteq I$ such that $Crit_\mathcal{G}(C)$ is critically sound w.r.t. $\mathcal{G}$.

Proof. To compute $Crit_\mathcal{G}(C)$, we start with $I_\varphi \cup I_\pi$, which is the symbolic instance of $I_\varphi \wedge I_\pi$, and then try to repeatedly remove each fact in I_π, as long as $can\text{-}sol_\mathcal{G}(I_\varphi \cup I'_\pi)$ contains $\mathbf{f}$, where I'_π is the sub-instance obtained from I_π; and $\mathbf{f}$ is the symbolic instance of $\mathbf{f}_\psi$. We stop when a minimal sub-instance $I_\varphi \cup I'_\pi$ of $I_\varphi \cup I_\pi$ is reached. Note that there are $|I_\pi|$ many atoms to be removed, hence at most $|I_\pi|$ many iterations are needed.

We then check fact in $I_\varphi \cup I'_\pi$ by trying to repeatedly remove each fact in I_φ and its related facts in I'_π as long as *can-sol*$_{\mathcal{G}}(I'_\varphi \cup I''_\pi)$ contains the facts $\mathbf{f}$, where I'_φ is the sub-instance obtained from I_φ; I''_π is the sub-instance obtained from I'_π; and $\mathbf{f}$ is the symbolic instance of $\mathbf{f}_\psi$. By construction, the constraint $(I'_\varphi \wedge I''_\pi, \mathbf{f}_\psi)$ is critically sound with respect to $\mathcal{G}$. Note that there are at most $|I_\varphi \cup I'_\pi|$ many facts to be removed, hence at most $|I_\varphi \cup I'_\pi|$ many iterations are needed, where in each iteration we check, by the labeling oracle $LABEL_{\mathcal{G}}$, if current *can-sol*$_{\mathcal{G}}(I'_\varphi \wedge I''_\pi)$ contains $\mathbf{f}$. Therefore, there are at most $|I_\varphi| + 2 \times |I_\pi|$ iterations in total since $|I'_\pi| \leq |I_\pi|$. $\square$

Lemma 19. Let $\mathcal{G}$ be a relational-to-RDF temporal schema mapping and let $C = (I_\varphi \wedge I_\pi, J_\psi)$ be a relational-to-RDF temporal GAV constraint. If $\mathcal{G}$ logically implies a relational-to-RDF temporal GAV constraint $C' = (I_\varphi, J_\psi)$, then $\mathcal{G}$ logically implies the constraint C.

Proof. Let *can-sol*$_{\mathcal{G}}(I)$ be a canonical universal solution for I w.r.t. $\mathcal{G}$. It follows that $(I,$*can-sol*$_{\mathcal{G}}(I))$ satisfies all constraints in $\mathcal{G}$. In addition, since the relational-to-RDF temporal schema mapping $\mathcal{G}$ logically implies C', it follows that $(I,$*can-sol*$_{\mathcal{G}}(I))$ satisfies $C = (I_\varphi \wedge I_\pi, J_\psi)$. It is esay to prove that $(I,$*can-sol*$_{\mathcal{G}}(I))$ also satisfies $C = (I_\varphi, J_\psi)$, where $I_\varphi \wedge I_\pi = I_\varphi \wedge I_\pi$. $\square$

Lemma 20. Let $\mathcal{G}$ be a relational-to-RDF temporal schema mapping and let $C = (I_\varphi, J_\psi)$ be a critically sound relational-to-RDF temporal GAV constraint w.r.t. $\mathcal{G}$. If $C' = (I'_\varphi, J_\psi)$ is a relational-to-RDF temporal GAV constraint, where $I_\varphi \subseteq I'_\varphi$, then $\mathcal{G}$ logically implies C'.

Proof. We show, by induction, that for every superset I'_φ of I_φ ($I_\varphi \subseteq I'_\varphi$), $\mathcal{G}$ logically implies (I'_φ, J_ψ). To begin with, we assume an atom $R(\mathbf{x}, \mathbf{t})$ (R is a relation symbol in the source schema of $\mathcal{G}$) and assume that $I'_\varphi = I_\varphi \wedge R(\mathbf{x},\mathbf{t})$. Let $\textit{can-sol}_{\mathcal{G}}(I)$ be a canonical universal solution for I w.r.t. $\mathcal{G}$. Since C is critically sound w.r.t. $\mathcal{G}$, we have $\mathcal{G}$ logically implies C. It is easy to prove that $(I,\ \textit{can-sol}_{\mathcal{G}}(I))$ satisfies (I'_φ, J_ψ). Let I'_φ be a superset of I_φ. By induction hypothesis, assume that $\mathcal{G}$ logically implies (I'_φ, J_ψ), and thus, $(I,\ \textit{can-sol}_{\mathcal{G}}(I))$ satisfies (I'_φ, J_ψ). Let I''_φ is a conjunction of I'_φ and an atom. We have to show that (I''_φ, J_ψ) has the desired properties. It is easy to prove that $(I,\ \textit{can-sol}_{\mathcal{G}}(I))$ also satisfies (I''_φ, J_ψ). $\square$

Claim 1. For every $i \geq 0$ such that $\mathcal{H}_i$ is defined, $\mathcal{H}_i$ consists of relational-to-RDF temporal GAV constraints that are sound w.r.t. $\mathcal{G}$. In particular, in the line 6 of Algorithm 7, it must be the case that $\textit{can-sol}_{\mathcal{H}_i}(I) \not\subseteq J$.

Proof. By induction on i. This claim is trivially true for $i = 0$, since $\mathcal{H} = \emptyset$. Now, for the case of $i+1$, let $(I,J) \in E$ be the data example chosen in line 3. Note that $J = \textit{can-sol}_{\mathcal{G}}(I)$. By the induction hypothesis, $\mathcal{H}_i$ is logically implied by $\mathcal{G}$, and hence, $\textit{can-sol}_{\mathcal{H}_i}(I) \subseteq \textit{can-sol}_{\mathcal{G}}(I) = J$. Let $f \in J/\textit{can-sol}_{\mathcal{H}_i}(I)$ as chosen in line 7. It follows from Lemma 16 that $\mathcal{G}$ logically implies the constraint $(I_\varphi \wedge I_\pi, \mathbf{f}_\psi)$. In other words, $(I_\varphi \wedge I_\pi, \mathbf{f}_\psi)$ is sound w.r.t. $\mathcal{G}$. It follows by Lemma 18 and Lemma 17 that the constraint added to $\mathcal{H}_{i+1}$ in line 14 or 16 is also sound w.r.t. $\mathcal{G}$. We conclude that $\mathcal{G}$ logically implies $\mathcal{H}_{i+1}$. $\square$

Claim 2. Let $i \geq 0$ such that $\mathcal{H}_i$ is defined. For every $C \in \mathcal{H}_i$ there is a $C^* \in \mathcal{G}$ such that $C^* \to C$. Furthermore, for each $C^* \in \mathcal{G}$, the set $\mathcal{H}_i(C^*) = \{C \in \mathcal{H}_i | C^* \to C\}$ is either empty or is a singleton set.

Proof. The first part of the claim follows from the Claim 1 and Lemma 16. The proof for the second part of this claim proceeds by induction on i. The base case, for $i = 0$, is trivially true.

For $i > 0$, consider any $C^* \in \mathcal{G}$. Let C'_i be the constraint added to $\mathcal{H}$ in the i-th iteration of the algorithm either line 13, 14 or 15, 16. If $C^* \not\to C'_i$, the result immediately follows from the induction hypothesis. Therefore, in what follows, we will assume that $C^* \to C'_i$. We distinguish two cases: (i) $\mathcal{H}_{i-1}(C^*) = \emptyset$. In this case, C'_i is the only constraint in $\mathcal{H}_i$ into which C^* maps; hence the result holds trivially. (ii)$\mathcal{H}_{i-1}(C^*) \neq \emptyset$. By the induction hypothesis, there is a unique $C_j \in \mathcal{H}_{i-1}$ such that $C^* \to C_j$. Note that, in this case, we have that $C^* \to C_j \times (I_\varphi \wedge I_\pi, \mathbf{f}_\psi)$ (by the property of direct product), and hence the if-condition in line 12 of the algorithm is satisfied. Moreover, we can show that C_j is, in fact, the constraint C chosen by the algorithm in line 13. This follows immediately from the induction hypothesis, because C^* homomorphically maps both to C_j and to C'_i, which, in turn, by Lemma 18 and the property of direct product, homomorphically maps to C, since $C'_i = Cirt_{\mathcal{G}}(C \times I_\varphi \wedge I_\pi, \mathbf{f}_\psi)$. Since C is explicitly removed from $\mathcal{H}$ in line 14, it follows that $\mathcal{H}_i(C^*)$ is again a singleton set. This concludes the proof for this claim. □

Claim 3. The size of the schema mapping $\mathcal{H}$ produced by Algorithm 3 is at most the size of $\mathcal{G}$.

Proof. By Claim 2 we have that, for each constraint C_i in $\mathcal{H}$ there is a constraint C_i^* in $\mathcal{G}$ such that $C_i^* \to C_i$, and, moreover, for $i \neq j$, we have that $C_i^* \neq C_j^*$. In other words, the constraint of $\mathcal{H}$ stands in a one-to-one correspondence with a subset of the constraint of $\mathcal{G}$. Furthermore, it is easy to see that each homomorphism in question must be surjective, otherwise the constraint C_i would not be *critically* sound w.r.t. $\mathcal{G}$. Therefore, each constraint C_i is of size at most the size of C_i^*. The claim immediately follows. □

Claim 4. For every $i > 0$ such that $\mathcal{H}_i$ is defined, we have $s_i > s_{i-1}$, and $s_i \leq n$, where $n =$

$\Sigma_{C \in \mathcal{G}} nvar(C)$.

Proof. Let T be a constraint in $\mathcal{G}$ and let $s_i = \Sigma_{T \in \mathcal{G}} s^T$, where $s_i^T = 0$ indicates that there is no constraint T in $\mathcal{G}$ such that $T \rightarrow C_{i+1}$; and s_i^T is the number of variables occurring in the unique element of $\mathcal{H}_i(T)$ (the corresponding constraint C in $\mathcal{H}_i$ to T, such that $T \rightarrow C_i$), otherwise (this is well defined by Claim 2). Let $nvar(C)$ be the number of variables occurring in C. We first show that $s_{i+1} > s_i$. Let C_{i+1} be the constraint added to $\mathcal{H}_{i+1}$ in the $i+1$-th iteration. Since C_{i+1} is critically sound with respect to $\mathcal{G}$, there is a T in $\mathcal{G}$ such that $T \rightarrow C_{i+1}$. By Claim 2, C_{i+1} is the unique element of $\mathcal{H}_{i+1}(T)$. It follows that s_{i+1}^T is the domain size of C_{i+1}. First, consider the case where $\mathcal{H}_i(T)$ is empty. In this case, C_{i+1} cannot be of the form $Cirt_{\mathcal{G}}((I_\varphi \wedge I_\pi, \mathbf{f}_\psi)) \times C$ with $C \in \mathcal{H}_i$, because this would imply that $C_{i+1} \rightarrow C$ and hence $C \in \mathcal{H}_i(T)$. In other words, C_{i+1} must be of the form $Cirt_{\mathcal{G}}((I_\varphi \wedge I_\pi, \mathbf{f}_\psi))$, and must have been added to $\mathcal{H}_{i+1}$ in line 16 of the algorithm. It immediately follows that $s_{i+1} > s_i$.

Next, consider the case where $\mathcal{H}_i(T)$ is non-empty, and let C_i be its unique element. It follows from Claim 2 that C_i must be the constraint removed from $\mathcal{H}_i$ by the algorithm in the $i+1$-th iteration and hence $C_{i+1} = Crit_{\mathcal{G}}(C_i \times (I_\varphi \wedge I_\pi, \mathbf{f}_\psi))$ for some I and f. In particular, we have that $C_{i+1} \rightarrow C_i$ by the property of direct product since $C_{i+1} \leftarrow C_i \times (I_\varphi \wedge I_\pi, \mathbf{f}_\psi)$ and $C_{i+1} \rightarrow (I_\varphi \wedge I_\pi, \mathbf{f}_\psi)$. In fact, the homomorphism h from the left-hand side of C_{i+1} to the left-hand side of C_i must be surjective; in other words, C_i must be the *h-image* of C_{i+1}, for otherwise, we could obtain a non-surjective homomorphism from the left-hand side of T to the left-hand side of C_i (namely the composition of h with the homomorphism $T \rightarrow C_{i+1}$), contradicting, via Lemma 16, the fact that C_i is critically sound w.r.t. $\mathcal{G}$. We also know that C_{i+1} is not isomorphic to C_i, because $C_{i+1} \rightarrow (I_\varphi \wedge I_\pi, \mathbf{f}_\psi)$ whereas $C_i \not\rightarrow (I_\varphi \wedge I_\pi, \mathbf{f}_\psi)$. It follows that the domain size of C_{i+1} is larger than that of C_i (otherwise h would be an isomorphism) and this claim is proved.

Next, we show that each s_i is at most n. Assume, for the sake of a contradiction, that $s_i > n$, for some well defined s_i. Then there is a constraint $C \in \mathcal{H}_i$ and $C^* \in \mathcal{G}$ with $\mathcal{H}_i(C_i) = \{C\}$ such that $nvar(C) > nvar(C^*)$. This means that there is a non-surjective homomorphism from C^* to C, resulting in a contradiction to the fact that C is critically sound w.r.t. $\mathcal{G}$. □

Theorem 21. Algorithm 7 is an optimal Occam algorithm for learning relational-to-RDF temporal schema mapping. Specifically, given a labeling oracle for a relational-to-RDF temporal schema mapping $\mathcal{G}$, as well as a set of universal examples for $\mathcal{G}$, Algorithm 7 returns a relational-to-RDF temporal schema mapping $\mathcal{H}$ such that E is, universal for $\mathcal{H}$ and the size of $\mathcal{H}$ and the size of $\mathcal{H}$ is at most the size of $\mathcal{G}$.

Proof. Let $\mathcal{H}_i$ denote the value of the variable $\mathcal{H}$ after the i-th iteration of the while loop of Algorithm 7, where $\mathcal{H}_i = \emptyset$, and $\mathcal{H}_i$ is undefined if i is larger than the total number of iterations of while loop by Claim 1- Claim 4. □

Theorem 22. For every set E of universal examples w.r.t. $\mathcal{G}$, Algorithm 7 returns a relational-to-RDF temporal schema mapping that is logically implied by $\mathcal{G}$.

Proof: The result follows directly from the Claim 1.

Theorem 23. Algorithm 7 asks at most

$$size(\mathcal{G})^2 \cdot maxsize(E^2)$$

many labeling queries, where $size(\mathcal{G})$ is the size of the goal schema mapping, and $maxsize(E)$ is the maximum number of facts in an input data example.

Proof. We will show a slightly stronger bound: Algorithm 7 asks at most

$$n_{\mathcal{G}} \cdot (|\mathcal{G}|) + k_{\mathcal{G}} \cdot k_E^2$$

many labeling queries, where $n_{\mathcal{G}} = \Sigma_{C \in \mathcal{G}} nvar(C)$, $|\mathcal{G}|$ is the number of constraints of $\mathcal{G}$, $k_{\mathcal{G}}$ is the maximum number of atoms in the left-hand side of a constraint of $\mathcal{G}$, and $k_E = max_{(I,J)\in E}|I|$ is the maximum number of source facts of a data example in E.

By Claim 4, the number of iterations of the algorithm is bounded by $n_{\mathcal{G}}$. In each iteration, the labeling oracle is called in line 12 (for testing logical implication) at most $|\mathcal{G}|$ times, and in line 14 and 16 (for computing the critically sound subconstraint) at most $k_{\mathcal{G}} \times (k_E)^2$, since the number of atoms in φ-part of the left-hand side of the product of $(I_\varphi \wedge I_\pi, \mathbf{f}_\psi) \times C$ is bounded by $k_{\mathcal{G}} \times k_E$ and the number of π-part of the left-hand side of the product is bounded by $k_{\mathcal{G}} \times (k_E)^2$. Note that the fact that the left-hand side of the constraint $C \in \mathcal{H}$ cannot contain more atoms than the left-hand side of a constraint in $\mathcal{G}$ (see the proof of Claim 2). □

5.4 Benchmark

5.4.1 Temporal D2RQ

D2RQ[21] is an open source software that implements W3C's *direct mapping* [53], which defines a simple transformation of content from a relational database to RDF.

We developed *Temporal D2RQ* by adding some features to *D2RQ* such that it can transform temporal information in a temporal relational database into RDF data with temporal components.

Let $T(A_1,\ldots,A_n)$ be a relation symbol, where A_n is a temporal attribute; let k be an integer between 1 and $n-1$ inclusively s.t. $(A_1,\ldots,A_k)$ is the primary key of T. Let T also be an interpretation of the relation symbol $T(A_1,\ldots,A_n)$. Assume a fact $T(a_1,\ldots,a_k,\ldots,a_{n-1},i)$ in a relation T.

Temporal D2RQ firstly generates a class T_{cls} for the relation T by assigning a URI for the class T_{cls}. Then, for each attribute A in $\{A_1,\ldots,A_n\}$, *Temporal D2RQ* generate a property A_p for it by assigning a URI for the attribute A.

To transfer the data from relational databases into RDF, we use the two functions, F_{uri} and F_{ruri} (see Section 4.3), to generate URIs that are used to identify each fact in a relation, values of attributes, and values of the classes `Statement`, `TNode`, and `Interval`. The function F_{uri} generates a URI for each fact in T, while F_{ruri} is used to generate instances of the classes `Statement`, `TNode`, and `Interval`.

- $F_{uri}(x_1,\ldots,x_k)$: Generate an URI for the fact $T(a_1,\ldots,\ a_k,\ldots,a_{n-1},i)$ in T, where the attribute set $\{A_1,\ldots,\ A_k,A_t\}$ is the primary key of the relation T, and A_t is the temporal attribute.

- $F_{ruri}(S,P,a)$: Given an URI S, an URI P (A_p or properties $temporal$, $interval$, $validFor$), and a value a appearing in a fact of the relation T, generate an URI as a value of the class `Statement`, `TNode`, or `Interval`.

Note that we take values of the primary key from T as the input for F_{uri}, while F_{ruri} not only takes values from T but also takes values generated by F_{uri} and F_{ruri}.

Temporal D2RQ will generate the following triples for the fact

$$T(a_1,\ldots,a_k,\ldots,a_{n-1},i):$$

- Non-temporal information in $T(a_1,\ldots,a_k,\ldots,a_{n-1},i)$: $(F_{uri}(a_1,\ldots,a_k),A_p^i,a_j)$ for any j between 1 and $n-1$ inclusively,
- Temporal information in $T(a_1,\ldots,a_k,\ldots,a_{n-1},i)$: We add a temporal component for each value of a non-temporal attribute A_i that are not part of the primary key (i.e., $A_{k+1},\ldots,A_n$).
  - $(F_{ruri}(F_{uri}(x_1,\ldots,x_k),A_p^i,x_i)),$
    $\texttt{subj},F_{uri}(x_1,\ldots,x_k)),$
  - $(F_{ruri}(F_{uri}(x_1,\ldots,x_k),A_p^i,x_i)),\texttt{obj},x_i),$
  - $(F_{ruri}(F_{uri}(x_1,\ldots,x_k),A_p^i,x_i)),\texttt{pred},A_p^i),$
  - $(F_{ruri}(F_{uri}(x_1,\ldots,x_k),A_p^i,x_i)),\texttt{temporal},$
    $F_{ruri}(F_{ruri}(F_{uri}(x_1,\ldots,x_k),A_p^i,x_i),\texttt{temporal},t)),$
  - $(F_{ruri}(F_{ruri}(F_{uri}(x_1,\ldots,x_k),A_p^i,x_i),\texttt{temporal},t),$
    $\texttt{interval},F_{ruri}(F_{ruri}(F_{ruri}(F_{uri}(x_1,\ldots,x_k),$
    $A_p^i,x_i),\texttt{temporal},t),\texttt{interval},t)),$
  - $(F_{ruri}(F_{ruri}(F_{ruri}(F_{uri}(x_1,\ldots,x_k),A_p^i,x_i),$
    $\texttt{temporal},t),\texttt{interval},t)),\texttt{validFor},t),$
  - $(F_{ruri}(F_{uri}(x_1,\ldots,x_k),A_p^i,x_i)),\texttt{type},\texttt{Statement}),$
  - $(F_{ruri}(F_{ruri}(F_{uri}(x_1,\ldots,x_k),A_p^i,x_i),$
    $\texttt{temporal},t),\texttt{type},\texttt{TNode}),$

- $(F_{ruri}(F_{ruri}(F_{ruri}(F_{uri}(x_1,\ldots,x_k),A_p^i,x_i),$

  $\texttt{temporal},t),\texttt{interval},t)),\texttt{type},\texttt{Interval}),$

- $(t,\texttt{type},\texttt{timeInterval}),$

- $(F_{uri}(x_1,\ldots,x_k),\texttt{type},T_{cls})$

where $k+1 \leq i \leq n-1$.

5.4.2 Temporal iBench

iBench[15] is the first metadata generator that can generate schema mappings for the context of relational-to-relational data exchange and their source instances. It can be used to evaluate data-exchange-related tasks. *iBench* generates schema mappings of different characteristics by using primitives and parameters of primitives. In particular, there are four primitives that are relevant for GAV constraints.

- *Copy* primitive (copying): $R(a,b,c,d) \rightarrow T(a,b,c,d)$
- *Delete* primitive (projection): $R(a,b,c,d) \rightarrow T(a,b)$
- *Merging* primitive (join multiple source relational atoms to create one target relational atom): $R_1(a,b,c,d) \wedge R_2(f,g,h,c) \wedge R_3(i,j,k,d) \rightarrow T(a,b,f,g,h,i,j,k,c,d)$
- *SelfJoin* primitive (self join a source relational atoms to create one target relational atom): $R(a,b,c,d) \rightarrow T_1(a,c,d)$ and $R(a,b,c,d) \wedge R(b,e,f,h) \rightarrow T_2(a,b)$

1. *JoinSize* is one of the critical parameters for the above primitives, which is the number of relational atoms per *Merging* primitive.

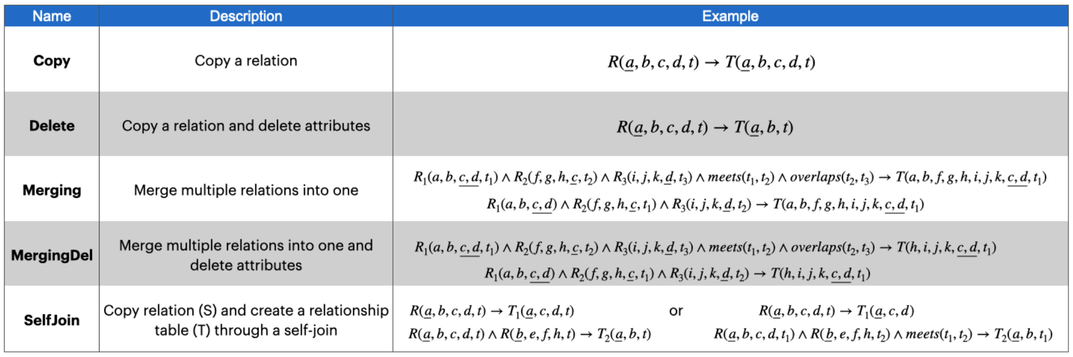
2. *SourceShare* (*TargetShare*) is another critical parameter that specifies the percentage of constraints that share source (target) relations. For instance, $R(a,b,c,d) \rightarrow T(a,b,c,d)$ and the constraint $R(a,b,c,d) \rightarrow T(a,b)$ share the same source relation. In addition, $R_1(a,b,c,d) \rightarrow T(a,b,c,d)$ and $R_2(a,b,c,d,e) \rightarrow T(a,b,c,d)$ share the same target relation. These two parameters ensure that a (source/target) relation may involve multiple constraints.

3. If *JoinKind* is the type of joins (start or chain) for *Merging* primitive and *MergingDel* primitive.

In general, *iBench* generates a constraint for each parameterized primitive except *SelfJoin* primitive, which generates two constraints as shown above.

5.4.2.1 Generation of Relational-to-Relational Temporal Schema Mappings

We developed *Temporal iBench* by adding some features to *iBench* to generate relational-to-RDF temporal schema mappings and their universal data examples with the help of *Temporal D2RQ*. *Temporal iBench* can generate relational-to-RDF temporal schema mappings. To do that, it first generates the relational-to-relational temporal GAV schema mappings. Then it directly converts those schema mappings into relational-to-RDF temporal schema mappings according to some conversion rules which will be presented later in this section. In the design of how to generate relational-to-RDF temporal schema mappings, we redefine the above four primitives and introduce a new primitive *MergingDel*. Those five newly defined primitives are listed in Figure 5.1.



Name	Description	Example
Copy	Copy a relation	$R(\underline{a}, b, c, d, t) \rightarrow T(\underline{a}, b, c, d, t)$
Delete	Copy a relation and delete attributes	$R(\underline{a}, b, c, d, t) \rightarrow T(\underline{a}, b, t)$
Merging	Merge multiple relations into one	$R_1(a, b, \underline{c, d}, t_1) \wedge R_2(f, g, h, \underline{c}, t_2) \wedge R_3(i, j, k, \underline{d}, t_3) \wedge meets(t_1, t_2) \wedge overlaps(t_2, t_3) \rightarrow T(a, b, f, g, h, i, j, k, \underline{c, d}, t_1)$ $R_1(a, b, \underline{c, d}) \wedge R_2(f, g, h, \underline{c}, t_1) \wedge R_3(i, j, k, \underline{d}, t_2) \rightarrow T(a, b, f, g, h, i, j, k, \underline{c, d}, t_1)$
MergingDel	Merge multiple relations into one and delete attributes	$R_1(a, b, \underline{c, d}, t_1) \wedge R_2(f, g, h, \underline{c}, t_2) \wedge R_3(i, j, k, \underline{d}, t_3) \wedge meets(t_1, t_2) \wedge overlaps(t_2, t_3) \rightarrow T(h, i, j, k, \underline{c, d}, t_1)$ $R_1(a, b, \underline{c, d}) \wedge R_2(f, g, h, \underline{c}, t_1) \wedge R_3(i, j, k, \underline{d}, t_2) \rightarrow T(h, i, j, k, \underline{c, d}, t_1)$
SelfJoin	Copy relation (S) and create a relationship table (T) through a self-join	$R(\underline{a}, b, c, d, t) \rightarrow T_1(\underline{a}, c, d, t)$ $R(\underline{a}, b, c, d, t) \wedge R(\underline{b}, e, f, h, t) \rightarrow T_2(\underline{a}, b, t)$ or $R(\underline{a}, b, c, d, t) \rightarrow T_1(\underline{a}, c, d)$ $R(\underline{a}, b, c, d, t_1) \wedge R(\underline{b}, e, f, h, t_2) \wedge meets(t_1, t_2) \rightarrow T_2(\underline{a}, b, t_1)$

Figure 5.1: Exemplary temporal primitives

Given a primitive listed in Figure 5.1, the generation of a relational-to-relational temporal GAV constraint of the primitive is controlled via those parameters in *iBench* and two extra parameters $(isTemporal, isPrimary)$, which we added for the generation of temporal variables.

- *isTemporal* is used to determine if the GAV constraint contains temporal variables. Specifically, *isTemporal* determines the appearance of temporal variables in the relational atoms that occurs in the left-hand side of the GAV constraint, such that it controls the generation of its constraints in the following ways: (a) If *isTemporal* is set to 0, then no temporal variables exist in the aforementioned relational atoms in the left-hand side of the GAV constraint; (b) if *isTemporal* is set to 1, then there must be at least one temporal variable in those relational atoms; (c) if *isTemporal* is set to -1, then the existence of a temporal variable in each of those relational atoms is randomly determined. In fact, in *Copy* primitive, *Delete* primitive, or *SelfJoin* primitive, since each of them involves exactly one relation symbol in the left-hand side of its constraint, the value of *isTemporal* indicates whether the relation symbol contains a temporal variable or not. However, in the *Merging* primitive and *MergingDel* primitive, the situation of the occurrence of temporal variables is more complicated. This is because each constraint of *Merging* primitive or *MergingDel* primitive involves multiple source relations in its left-hand side, and each source relation symbol appears only once. Therefore, if *isTemporal* is 1, a further determination of temporal variables for each relational atom is required. Specifically, when generating a constraint for *Merging* primitive or *MergingDel* primitive:

(a) if *isTemporal* is 0, then we generate GAV constraints in the same way as *iBench*.

(b) if *isTemporal* is 1, then we add a temporal attribute to each source relation symbol involved in the constraint so that there is a fresh temporal variable in each source relational atom. Hence, the number of temporal variables equals the number of source relational atoms in the left-hand side of a constraint.

(c) if *isTemporal* is -1, then for each source relation symbol involved in the left-hand side of the constraint, we randomly determine if it contains a temporal attribute. Hence, the number of temporal variables in the left-hand side of a constraint is randomly determined.

The relational atom in the right-hand side of a constraint contains one of the temporal variables from the left-hand side (for simplicity, we choose the first temporal variable from the left-hand side).

- *isPrimary* is used to determine the following questions per *SelfJoin* primitive, where there are two constraints, deleting constraint (the first one in Figure 5.1) and self-join constraint (the second one in Figure 5.1), to be created per *SelfJoin* primitive: 1) if the temporal variable will occur in the right-hand side of the first constraint, 2) if there is multiple temporal variables in the second constraint, and 3) which temporal variable in the left-hand side of a constraint will occur in the right-hand side of the second constraint. Recall Section 3.2, when we investigate the semantic adequacy, we observed that whether or not a constraint contains multiple temporal variables matters and that whether or not a constraint contains a temporal variable in its right-hand side matters. Therefore, without lose generality, *isPrimary* ensures that *Temporal iBench* can cover all important types of constraints.

(a) If *isPrimary* is 1 (the left example in Figure 5.1), i) the right-hand side of the deleting constraint contains the temporal variable from the left; ii) the left-hand side of the self-join constraints contains exactly one temporal variable; iii) the right-hand side of the self-join constraints contains the temporal variable that is from the left.

(b) If *isPrimary* is 0 (the right example in Figure 5.1), i) the right-hand side of the deleting constraint doesn't contain any temporal variable; ii) the left-hand side of the self-join constraints contains two temporal variables; iii) the right-hand side of the self-join constraints contains the temporal variable that is from the first relational atom in the left-hand side.

Note that in each generated schema, temporal variables will never be part of a primary key of a relation.

If a constraint contains multiple temporal variables in its left-hand side, then we randomly generate a conjunction of Allen's atoms. Assume there are *m* temporal variables in a constraint. We begin by randomly generating an integer that is less than or equal to m^2 as the number of Allen's atoms. Then we create each Allen's atom by randomly selecting two temporal variables from all temporal variables in the left-hand side and by randomly picking up one Allen's relation for the two temporal variables. In general, it is possible that the randomly generated conjunctions of Allen's atoms are unsatisfiable, which makes the generated constraint meaningless. Therefore, we verify if the generated conjunction of Allen's atoms is satisfiable, and if not, *Temporal iBench* keeps generating a new conjunction until a satisfiable one is found. The verifier first transfers each conjunction of Allen's atoms into an Integer Linear Programming (IP) problem and then we can use a powerful tool, such as *Gurobi* [40], to check the satisfiability.

Given a conjunction of Allen's atoms, we can convert it into an IP problem as follows: Check the feasibility of a set of inequalities of the form

$$\text{Maximize } x_1 + x_2 + \cdots + x_n$$

$$x_i - x_j \leq -1$$

$$l_i \geq x_i$$

$$i \neq j,$$

where $i, j = 1, \ldots, n$, and $x_i, x_j, l_i \in \mathbb{N}$. Specifically, given an Allen's atom of the form $t_1 \rho t_2$, for each temporal variable t we generate two temporal variables t_s and t_e for time points; t_s indicates the starting point of t, and t_e indicates the ending point of the temporal variable t, which represents an unknown time interval. Given $t_1 \rho t_2$, it results in four variables t_s^1, t_e^1, t_s^2, t_e^2 for time points. Then based on the definition of Allen's relations in Section 2.2, each linear relation in the definition of the Allen's relation ρ is translated, with the four variables, into an inequality of the first type in the above inequalities. Assume a linear relation $t'' < t'$. It is converted into the inequality $t'' - t' \leq -1$. If the linear relation is $t'' = t'$, we replace t'' by t' throughout all inequalities. In addition, for each variable t' in inequalities, we add a constraint $0 \leq t'$ since time points should start from 0. Finally, we limit our variables to integers. As a result, a conjunction of Allen's atoms is translated into an IP problem. For example, t_1 `meets` $t_2 : t_1^s < t_e^1 = t_s^2 < t_e^2$ is translated into the following inequalities:

$$t_s^1 - t_e^1 \leq -1,$$

$$t_e^1 - t_e^2 \leq -1,$$

$$0 \geq t_s^1,$$

$$0 \geq t_e^1,$$

$$0 \geq t_e^2,$$

where $t_s^1, t_e^1, t_e^2 \in \mathbb{N}$.

The above IP problem have four characteristics: a) each inequality of the first type contains two variables; b) the coefficients of x_i and x_j are of opposite sign (called monotone system); c) every variable is bounded; d) coefficients are in $\{1, -1\}$.

In the IP problem, there is an important concept called *totally unimodular*. A matrix A is totally unimodular if every square submatrix of A has determinant -1, 0, or $+1$ [57]. The importance of the concept stems from Lemma 24.

Lemma 24. Let A be an integer matrix such that the following conditions hold: (1) each entry is in $\{0, 1, -1\}$; (2) each row contains at most two non-zero entries; (3) if a row contains exactly two non-zero entries, then they are in the opposite signs.

Then for the determinant of the matrix A, denoted by $detA$, we have $|det\ A| \leq 1$, and the matrix A is totally unimodular.

Please refer to Chapter 19 in the book [57] for the proof of Lemma 24;

According to Lemma 24, the coefficient matrix of the IP problem described above is totally unimodular. This is a very important property for an IP problem since each linear program with integral input data and a totally unimodular coefficient matrix has an integral solution. Linear programming problems are known to be solvable in polynomial time, so the IP problem mentioned above could be solved in polynomial time.

SourceShare/TargetShare: In *Temporal iBench*, we implemented the functionality that allows the *Copy* primitive, *Delete* primitive, *Merging* primitive, and *MergingDelete* primitive to either reuse source relations from all generated source relations or reuse target relations from all the generated target relations. However, for the *SelfJoin* primitive, we only allow it to reuse source relations (target relations) that are source relations (target relations) generated by the *SelfJoin* primitive, *Copy* primitive, and *Delete* primitive.

Keys and Foreign Keys: Note that in *iBench* and *Temporal iBench*, the generation of schema mappings is based on the keys and foreign keys of each relations (for details please refer to paper [15] or our source code in github [1]). *Temporal iBench* defines a primary key for each source relation symbol and for each target relation symbol. It also defines foreign keys for some of those source relation symbols that are involved in *Merging* primitive, *MergingDel* primitive, and *SelfJoin* primitive.

[1]https://github.com/Evelynchengusa/SchemaMappingGeneration.git

5.4.2.2 Generation of Relational-to-RDF Temporal Schema Mappings

Given a temporal source relational schema **S**, a temporal target relational schema **T**, and a set Σ of GAV constraints between **S** and **T**, *Temporal iBench* first generates source instances over **S** based on the keys and foreign keys, and then generate universal solutions for the generated instances w.r.t. Σ. To generate source instance over a temporal source schema, *Temporal iBench* adds an extra data type Date to ToXgene (a data generator used by *iBench*), such that it can generate time intervals. Furthermore, *Temporal iBench* also implemented the function (which was not implemented in *iBench*) that helps with generating an instance of a relation that is involved in a self-join constraint such that 1) the instance satisfies keys and foreign keys of the relation; 2) there exists at least one assignment from the left-hand side of the self-join constraint to the instance of the relation. In addition, given a relational-to-relational temporal schema mapping $\mathcal{M}$ and a source instance I, a canonical universal solution J can be generated by *Temporal iBench* using the concrete chase algorithm described in Section 3.1. Hence, the pair (I, J) is a universal data example for $\mathcal{M}$. Taking **T** and J as inputs, we generate a temporal target RDF graph instance J' by applying *Temporal D2RQ*.

Transfer a relational-to-relational temporal GAV schema mapping $\mathcal{M}$ into a relational-to-RDF GAV schema mapping $\mathcal{M}'$. By considering each target relational atom as a relational fact, we are able to process the conversion from a target relational atom into a set of RDF atoms in the same way as *Temporal D2RQ*. Therefore, each constraint in $\mathcal{M}$ is converted into a set of constraints in $\mathcal{M}'$. For example, the relational-to-relational temporal GAV constraint $R(\underline{a}, b, t_1) \to T(\underline{a}, b, t_1)$ is transferred into the relational-to-RDF GAV constraints listed in Figure 5.2.

$$
\begin{aligned}
&R(\underline{a}, b, t_1) \rightarrow \big(\mathit{Furi}(a), \text{http://www.w3.org/1999/02/22-rdf-syntax-ns\#type}, \text{file:///vocab/T}\big)\\
&R(\underline{a}, b, t_1) \rightarrow \big(\mathit{Furi}(a), \text{file:///batchoboactivelearnm12\_1.nt\#/T\_thank}, a\big)\\
&R(\underline{a}, b, t_1) \rightarrow \big(\mathit{Furi}(a), \text{file:///batchoboactivelearnm12\_1.nt\#/T\_stick}, b\big)\\
&\qquad \wedge \big(\mathit{Fruri}(\mathit{Furi}(a), \mathit{T\_stick}, b), \text{http://www.w3.org/1999/02/22-rdf-syntax-ns\#subject}, \mathit{Furi}(a)\big)\\
&\qquad \wedge \big(\mathit{Fruri}(\mathit{Furi}(a), \mathit{T\_stick}, b), \text{http://www.w3.org/1999/02/22-rdf-syntax-ns\#object}, b\big)\\
&\qquad \wedge \big(\mathit{Fruri}(\mathit{Furi}(a), \mathit{T\_stick}, b), \text{http://www.w3.org/1999/02/22-rdf-syntax-ns\#predicate}, \text{file:///batchoboactivelearnm12\_1.nt\#/T\_stick}\big)\\
&\qquad \wedge \big(\mathit{Fruri}(\mathit{Furi}(a), \mathit{T\_stick}, b), \text{file:///vocab/P\_temporal}, \mathit{Fruri}(\mathit{Fruri}(\mathit{Furi}(a), \text{T\_stick}, b)), \mathit{P\_temporal}, t_1)\big)\\
&\qquad \wedge \big(\mathit{Fruri}(\mathit{Fruri}(\mathit{Furi}(a), \mathit{T\_stick}, b)), \mathit{P\_temporal}, t_1), \text{file:///vocab/P\_interval}, \mathit{Fruri}(\mathit{Fruri}(\mathit{Fruri}(\mathit{Furi}(a), \text{T\_stick}, b)), \mathit{P\_temporal}, t_1)), \mathit{P\_interval}, t_1)\big)\\
&\qquad \wedge \big(\mathit{Fruri}(\mathit{Fruri}(\mathit{Fruri}(\mathit{Furi}(a), \text{T\_stick}, b)), \mathit{P\_temporal}, t_1)), \mathit{P\_interval}, t_1), \text{file:///vocab/P\_validFor}, t_1\big)\\
&\qquad \wedge \big(\mathit{Fruri}(\mathit{Fruri}(\mathit{Furi}(a), \text{T\_stick}, b)), \mathit{P\_temporal}, t_1), \text{http://www.w3.org/1999/02/22-rdf-syntax-ns\#type}, \text{file:///vocab/TNode}\big)\\
&\qquad \wedge \big(\mathit{Fruri}(\mathit{Fruri}(\mathit{Fruri}(\mathit{Furi}(a), \text{T\_stick}, b)), \mathit{P\_temporal}, t_1)), \mathit{P\_interval}, t_1), \text{http://www.w3.org/1999/02/22-rdf-syntax-ns\#type}, \text{file:///vocab/Interval}\big)\\
&\qquad \wedge \big(t_1, \text{http://www.w3.org/1999/02/22-rdf-syntax-ns\#type}, \text{file:///vocab/timeInterval}\big)\\
&\qquad \wedge \big(\mathit{Fruri}(\mathit{Furi}(a), \mathit{T\_stick}, b), \text{http://www.w3.org/1999/02/22-rdf-syntax-ns\#type}, \text{file:///vocab/Statement}\big)
\end{aligned}
$$

Figure 5.2: Relational-to-RDF GAV constraints converted from $R(\underline{a}, b, t_1) \rightarrow T(\underline{a}, b, t_1)$.

5.5 Experiments

In this section, we will present an experimental evaluation of our active learning algorithm. Initially, we describe our data preparation process, utilizing *Temporal iBench* to create both goal relational-to-RDF temporal schema mappings and canonical universal data examples. Subsequently, we will present our evaluation methodology, which draws inspiration from the experimental methodology employed in paper [64]. Lastly, we present an in-depth analysis of the experimental results obtained from conducting experiments on the data generated by *Temporal iBench* using our active learning algorithm.

5.5.1 Data Preparation and Experimental Methodology

Schema-mapping Generation We use *Temporal iBench* to create three types of relational-to-RDF temporal schema mappings: simple, moderate, and complex. This is implemented via generating relational-to-relational temporal GAV schema mappings, where each s-t tgd is a relational-to-relational temporal GAV constraint. Specifically, as mentioned above, we first generate five simple relational-to-relational temporal GAV schema mappings, five moderate relational-to-relational temporal GAV schema mappings, and five complex relational-to-relational temporal GAV schema mappings.

1. A simple relational-to-relational temporal GAV schema mapping consists of one Copy constraint, one Delete constraint, one Merging constraint, one MergingDel constraint, and one SelfJoin constraint.

2. A moderate relational-to-relational temporal GAV schema mapping consists of two copy constraints, two Delete constraints, two Merging constraints, two MergingDel constraints, and two SelfJoin constraints.

3. A complex relational-to-relational temporal GAV schema mapping consists of three copy constraints, three Delete constraints, three Merging constraints, three MergingDel constraints, and three SelfJoin constraints.

Then *Temporal iBench* transforms each relational-to-relational temporal GAV constraint into a set of relational-to-RDF temporal GAV constraints. For example, 1 relational-to-relational temporal GAV constraints could be converted into 57 relational-to-RDF temporal GAV constraints.

We set the *JoinKind* with its default value (star join). We set different values for *JoinSize*, $numOfJoinAttributes$ (denoted by *#JoinAttrs*), and *PrimaryKeySize* (denoted by *#PKSize*) for each of two types of mappings. Statistics of schema mappings of those three types are shown in Table 5.1, where *Avg. source relations* indicates the average number of source relations in generated schema mappings of type X; *Avg. arity* indicates the average number of attributes in each source relation; *Avg. #constraint* indicates the average number of constraints in generated schema mappings of type X; *Avg. source temporal relations* indicates the average number source temporal relations in generated schema mapping of type X.

Table 5.1: Statistics for different mapping types.

	JoinSize	*#JoinAttrs*	*#PKSize*	Avg. source relations	Avg. arity	Avg. #constraints	Avg. source temporal relations
Simple	3	2	2	9	7.2	58.4	4.2
Moderate	6	2	2	26	9.2	123.2	12.6
Complex	9	1	1	69.4	606.2	200	38

Data Example Generation *Temporal iBench* randomly generates source instances for a schema mapping $\mathcal{G}$ according to an integer value $NumofElem$ specifying the number of facts generated in each relation.

For each raw source instance I with $NumOfElem = m$ generated by *Temporal iBench*, there are m assignments from the left-hand side of each constraint to the source instance I. Here in our experiment, we set $NumOfElem$ to 5. To avoid uniformity, as in GAVLearn paper [64], we introduce a parameter α, the value of which is from 0 to 1. We will generate a sub-instance I' from the raw source instance I. Given a source instance I and α, each fact in I has α chance to be in I'.

Once *Temporal iBench* creates a source instance I' for a schema mapping relational-to-RDF schema mapping $\mathcal{G}$, we could use *Temporal iBench* to generate a canonical universal data example (I', J'). This way, a set E of canonical universal data examples for $\mathcal{G}$ are obtained. *ExNum* is a parameter that we use to adjust the number of data examples in our experiments.

Evaluation Methodology We evaluate the performance of our proposed learning algorithm on the three different types of schema mappings. For each type, we create five schema mappings and measure the learning algorithm's average precision, recall, F-score, and running time of the learning algorithm. Given a source instance I, the precision of $\mathcal{H}$ on I w.r.t. a goal mapping $\mathcal{G}$ is the fraction of triples in $can-sol_{\mathcal{H}}(I)$ that are contained in $can-sol_{\mathcal{G}}(I)$, while the recall of $\mathcal{H}$ is the fraction of triples of $can-sol_{\mathcal{G}}(I)$ that are included in $can-sol_{\mathcal{H}}(I)$. The F-score of $\mathcal{H}$ on a data set is computed using the formula $\frac{2*\overline{Recall}*\overline{Precision}}{\overline{Recall}+\overline{Precision}}$ that combines average precision and average recall on the data set.

Let $\alpha = \{0.1, 0.3, 0.5\}$ and let $\mathbf{ExNum} = \{10, 30, 50, 70, 90\}$. Given a type X and a schema mapping of type X, for each pair of $(\alpha, ExNum)$, we do the following: a) generate a set of data examples as described above (e.g., with $\alpha = 0.5$ and $ExNum = 10$);
b) split data examples into a training set $E(50\%)$ and a test set $V(50\%)$; c) run our proposed learning algorithm on E to obtain a schema mapping $\mathcal{H}$. d) evaluate $\mathcal{H}$ on V and collect the precision, recall, and F-score information. We call c) and d) an active learning task with the parameter setting $(\alpha, ExNum)$.

Comprehensiveness and Representativeness The *comprehensiveness* and the *representativeness* are two measures for helping with analysis on the experiments results: the *comprehensiveness* of a training set E w.r.t. a goal schema mapping $\mathcal{G}$, denoted by $Comp_{\mathcal{G}}(E)$, indicates whether the training set E can be used to learn different constraints in $\mathcal{G}$; The *representativeness* of a training set E w.r.t. a test set V and a goal schema mapping $\mathcal{G}$, denoted by $Rep_{\mathcal{G},V}(E)$, indicates how well a training set can cover the constraints triggered in a testing set (we say a constraint σ is triggered if there is a data example $(I,J) \in E$ such that there is a homomorphism from the left-hand side of σ to I).

Let $\mathcal{G} = \{\mathbf{S}, O, \Sigma\}$ be a goal relational-to-RDF temporal schema mapping. Let E and V be a training set and a test set created for $\mathcal{G}$, respectively. The comprehensiveness and the representativeness are defined as follows:

$$Comp_{\mathcal{G}}(E) = \frac{|\Sigma^E|}{|\Sigma|} \text{ and } Rep_{\mathcal{G},V}(E) = \frac{|\Sigma^E \cap \Sigma^V|}{|\Sigma^V|},$$

where Σ^X denotes the subset of Σ that is triggered in the set X. Generally, a high $Rep_{\mathcal{G},V}(E)$ indicates that the training set E provides sufficient information such that the learning algorithm can learn a high-quality schema mapping. For more details, please see [64].

We observe that if the $Comp_{\mathcal{G}}(E)$ is 1, then $Rep_{\mathcal{G},V}(E)$ is 1.

Implementation The metadata generator *Temporal iBench* and the learning algorithm are implemented in JAVA. We used PostgreSQL 2.5 with default settings. All experiments were run on a Linux machine with an Intel 2.60 CPU (GHz) and 63GB RAM. The source code in this chapter can be found online [2].

5.5.2 Experimental Result Analysis

The experiment results are provided in Table 5.2, 5.3, and 5.4 (in Table 5.4, we only set α to be 0.1). Each row of those two tables corresponds to a particular active learning task with a specific parameter setting $(\alpha, ExNum)$. In the remainder of this section, we will use $(\alpha, ExNum)$ to refer to the corresponding active learning task. Recall that we have five schema

[2]https://github.com/Evelynchengusa/SchemaMappingGeneration

mappings of each type X. We collect the comprehensiveness and representativeness for each of them and report the average values (denoted by $\overline{Comp}$ and $\overline{Rep}$) and standard deviation values for comprehensiveness and representativeness, respectively. We run an active learning task $(\alpha, ExNum)$ for each schema mapping, and average the recall value and time it costs for each run. Furthermore, the F_s score is calculated by the formula $\frac{2*\overline{Recall}*\overline{Precision}}{\overline{Recall}+\overline{Precision}}$, where $\overline{Precision}$ is 1. This is because according to Theorem 22, we know that $(I, can-sol_{\mathcal{G}}(I))$ satisfies $\mathcal{H}$. Hence, $can-sol_{\mathcal{H}}(I)$ is a subset of $can-sol_{\mathcal{G}}(I)$. From the definition of precision, we know that *precision* is always 1. For more details, please see [64]. We also record the precision in our experiment, and it turns out that all active learning tasks have a precision of 1. Hence, precision is not reported in Table 5.2, 5.3, and 5.4.

Table 5.2: Results of active learning tasks on schema mappings of simple type.

α	$ExNum$	$\|E\|$	$\overline{Comp}$	$\overline{Rep}$	$\overline{Recall}$	F_s	$\overline{Time}$
0.1	10	5	$0.662 \pm 11.7\%$	$1.000 \pm 0\%$	$1.000 \pm 0\%$	1.000	0.526s
	30	15	$0.833 \pm 19.0\%$	$0.833 \pm 0\%$	$0.994 \pm 1.0\%$	0.997	0.753s
	50	25	$0.833 \pm 19.0\%$	$1.000 \pm 0\%$	$0.995 \pm 0.9\%$	0.998	0.783s
	70	35	$0.952 \pm 19.7\%$	$1.000 \pm 0\%$	$0.994 \pm 0.7\%$	0.997	0.881s
	90	45	$0.833 \pm 19.0\%$	$1.000 \pm 0\%$	$0.998 \pm 0.4\%$	0.999	0.869s
0.3	10	5	$0.833 \pm 26.0\%$	$1.000 \pm 0\%$	$0.951 \pm 4.8\%$	0.975	5.537s
	30	15	$0.967 \pm 9.6\%$	$1.000 \pm 0\%$	$1.000 \pm 0\%$	1.000	9.261s
	50	25	$0.967 \pm 9.6\%$	$1.000 \pm 0\%$	$1.000 \pm 0\%$	1.000	9.201s
	70	35	$1.000 \pm 9.5\%$	$1.000 \pm 0\%$	$1.000 \pm 0\%$	1.000	9.311s
	90	45	$1.000 \pm 9.6\%$	$1.000 \pm 0\%$	$1.000 \pm 0\%$	1.000	9.444s
0.5	10	5	$0.933 \pm 15.2\%$	$0.967 \pm 0\%$	$0.955 \pm 8.9\%$	0.977	39.540s
	30	15	$0.967 \pm 9.6\%$	$1.000 \pm 0\%$	$1.000 \pm 0\%$	1.000	42.144s
	50	25	$0.967 \pm 9.6\%$	$1.000 \pm 0\%$	$1.000 \pm 0\%$	1.000	42.464s
	70	35	$0.357 \pm 9.6\%$	$1.000 \pm 0\%$	$1.000 \pm 0\%$	1.000	42.630s
	90	45	$0.967 \pm 9.6\%$	$1.000 \pm 0\%$	$1.000 \pm 0\%$	1.000	43.212s

Table 5.3: Results of active learning tasks on schema mappings of moderate type.

α	$ExNum$	$\lvert E\rvert$	$\overline{Comp}$	$\overline{Rep}$	$\overline{Recall}$	F_s	$\overline{Time}$
0.1	10	5	$0.264 \pm 8.5\%$	$0.990 \pm 1.8\%$	$0.941 \pm 7.4\%$	0.970	14.434s
	30	15	$0.268 \pm 8.5\%$	$0.993 \pm 0.9\%$	$0.972 \pm 2.3\%$	0.986	15.093s
	50	25	$0.270 \pm 8.4\%$	$1.000 \pm 0\%$	$0.982 \pm 0.7\%$	0.991	15.627s
	70	35	$0.270 \pm 8.4\%$	$1.000 \pm 0\%$	$0.994 \pm 0.5\%$	0.997	16.034s
	90	45	$0.270 \pm 8.4\%$	$1.000 \pm 0\%$	$0.994 \pm 1.2\%$	0.997	16.656s
0.3	10	5	$0.350 \pm 16.5\%$	$0.906 \pm 18.7\%$	$0.980 \pm 4.0\%$	0.990	154.742s
	30	15	$0.386 \pm 14.8\%$	$0.951 \pm 9.8\%$	$0.998 \pm 0.3\%$	0.999	178.678s
	50	25	$0.386 \pm 14.8\%$	$0.835 \pm 20.3\%$	$0.996 \pm 0.5\%$	0.998	174.069s
	70	35	$0.395 \pm 17.4\%$	$1.000 \pm 0.0\%$	$1.000 \pm 0.0\%$	1.000	374.913s
	90	45	$0.476 \pm 17.3\%$	$0.807 \pm 23.6\%$	$0.994 \pm 0.8\%$	0.997	523.266s
0.5	10	5	$0.556 \pm 23.9\%$	$0.859 \pm 17.1\%$	$0.949 \pm 5.4\%$	0.974	1870.702s
	30	15	$0.645 \pm 24.7\%$	$0.862 \pm 18.1\%$	$0.985 \pm 1.6\%$	0.992	3118.607s
	50	25	$0.673 \pm 24.5\%$	$0.909 \pm 17.8\%$	$0.994 \pm 0.7\%$	0.997	3525.101s
	70	35	$0.609 \pm 10.1\%$	$0.999 \pm 0.2\%$	$0.996 \pm 0.5\%$	0.998	3742.805s
	90	45	$0.711 \pm 18.1\%$	$1.000 \pm 0.0\%$	$0.999 \pm 0.0\%$	0.999	4049.661s

Correctness of Active Learning Algorithm In general, $\overline{Rep}$ values are more correlated to $\overline{Recall}$ than $\overline{Comp}$ values. The high $\overline{Comp}$ value indicates that the training set E covers a variety of constraints in a goal schema mapping, such that the active learning task can produce relational-to-RDF temporal schema mappings that are closer to the goal schema mapping. The $\overline{Rep}$ value of 1 indicates that the information about all the constraints covered by a testing set V

Table 5.4: Results of active learning tasks on schema mappings of complex type.

α	*ExNum*	$\|E\|$	$\overline{Comp}$	$\overline{Rep}$	$\overline{Recall}$	F_s	$\overline{Time}$
0.1	10	5	$0.227 \pm 5.7\%$	$0.994 \pm 0.8\%$	$0.804 \pm 16.2\%$	0.892	1863.437s
	30	15	$0.229 \pm 5.7\%$	$0.995 \pm 0.7\%$	$0.981 \pm 2.4\%$	0.990	1943.565s
	50	25	$0.230 \pm 5.6\%$	$1.000 \pm 0\%$	$0.982 \pm 2.9\%$	0.991	2112.356s
	70	35	$0.230 \pm 5.6\%$	$1.000 \pm 0\%$	$0.989 \pm 1.1\%$	0.994	2262.333s
	90	45	$0.230 \pm 5.6\%$	$1.000 \pm 0\%$	$0.990 \pm 0.8\%$	0.995	2446.097s

is provided by the corresponding training set E. Hence, a good learning algorithm is expected to generate higher $\overline{Recall}$ values when $\overline{Rep}$ values increase. In particular, if $\overline{Rep} = 1.0$, then $\overline{Recall}$ is expected to be 1.0.

As shown in Table 5.3, when α values are fixed, the higher the $\overline{Rep}$ values are, the higher the $\overline{Recall}$ values are. This is consistent with what we stated earlier. For instance, in Table 5.3, $\overline{Recall}$ value for $(0.1, 10)$ is 0.941; the $\overline{Recall}$ value for $(0.1, 30)$ is 0.972; and the $\overline{Recall}$ value for $(0.1, 50)$ is 0.982. On the other hand, we also observe that when $\overline{Rep}$ is less than 1, the $\overline{Recall}$ value is less than 1. This observation is consistent with our earlier statements.

However, different from what we expected, from Table 5.2 and Table 5.3, and 5.4, we observe that when the $\overline{Rep}$ is 1, the maximum value of $\overline{Recall}$ can be 1. In Table 5.2, the active learning tasks of $(0.1, 50)$, $(0.1, 70)$, $(0.1, 90)$, and $(0.3, 10)$ have $\overline{Recall}$ values less than 1, whereas their $\overline{Rep}$ values are 1. And we call *exceptions* these cases that are inconsistent with what we expect. For other settings, $\overline{Recall} = 1$ when $\overline{Rep} = 1$. We call *normal* cases these cases that are consistent with what we expect. In addition, Table 5.3 also has exceptions such as $(0.1, 50)$, $(0.1, 70)$, $(0.1, 90)$, $(0.5, 90)$.

Two possible reasons cause such exceptions. The first one relates to the characteristics of each data example in the training set, which is the main reason that causes the exceptions in our experiments. We will illustrate this by using the following two examples. Note that in our temporal iBench, each relational-to-RDF temporal schema mapping is converted from a relational-to-relational temporal GAV schema mapping, and each relational-to-relational temporal GAV constraint corresponds to a set of relational-to-RDF temporal full s-t tgds. Hence, for simplicity, we will use relational-to-relational temporal GAV schema mappings as our examples:

Let $\mathcal{M}_1$ be a relational-to-RDF temporal schema mapping consisting of a copy constraint:

$$R(x,y,t) \rightarrow T(x,y,t).$$

Consider a training example (I_1,J_1) where $I_1 = \{R(a,a,[2010,2012))\}$. When we run the active learning algorithm, it will return a schema mapping $\mathcal{M}_1'$ consisting of the constraint $\sigma = R(x,x,t) \rightarrow T(x,x,t)$. Assume a testing data example (I_1',J_1'), where $I_1' = \{R(c,d,[2011,2013))\}$ and $J_1' = \{T(c,d,[2011,2013))\}$. When applying the chase algorithm, it will return an empty universal solution. Therefore, the recall is not 1 in this case, even though the representativeness is 1.

Let $\mathcal{M}_2$ be a relational-to-RDF temporal schema mapping consisting of a self-join constraint: $R(x,y,z,t_1) \wedge R(y,u,v,t_2) \rightarrow T(x,y,t_1)$. Consider a training example (I_2,J_2) where $I_2 = \{R(a,a,c,\ [2010,2012))\}$. When we run the active learning algorithm, it will return a schema mapping $\mathcal{M}_2'$ consisting of a constraint $R(x,x,y,t_1) \rightarrow T(x,x,t_1)$. Assume a testing data example (I_2',J_2'), where $I_2' = \{R(a,b,c,\ [2011,2013)),\ R(b,c,d,[2012,2013))\}$ and $J_2' = \{T(a,b,[2011,2013))\}$. For this example, the recall value of the active learning task is less than

1 even though the representativeness is 1. We leave the details to the reader.

Given a set E of data examples, we say that a schema mapping $\mathcal{M}$ fits the set E if for each data example (I,J) the target instance J is a universal solution for I w.r.t. $\mathcal{M}$. Our active learning algorithm aims to generate the most *general fitting schema mapping* (for the definition of the concept, please see [1]). Therefore, when the schema mapping produced by the active learning algorithm is not the most general fitting schema mapping of a training set, the recall value is less than 1 even though the representativeness is 1. Furthermore, we observe that in both tables, there are more exceptions when $\alpha = 0.1$ than when $\alpha = 0.3$ and $\alpha = 0.5$. In the examples described above, it is evident that the values of the first and the second attributes of the relation R are the same. We call such fact the fact with *special characteristics*. When α is small, a relation in the training set is more likely to only contain facts with special characteristics. Hence, the schema mapping learned by the active learning algorithm will be strongly impacted by the facts with the special characteristics since the constraints can only be learned through those facts with the special characteristics.

The second one is caused by the design of the active learning algorithm, which only generates one critically sound constraint for each candidate constraint. At the same time, there could be multiple critically sound constraints. We refer the reader to Section 4.1 in [64] for details.

Efficiency of Active Learning Algorithm The running time of each active learning task is recorded in Table 5.2 and Table 5.3. The efficiency of the active learning algorithm is highly related to the size of the training set. The tables show that the running time increases with both the α value and the number of data examples increasing. In Table 5.2, the running time ranges from $0.526s$ to $43.212s$. In Table 5.3, the running time ranges from $14.434s$ to $4049.661s$ (which is 1.12 hour). Furthermore, from Table 5.4, we observe that the time it takes to run the active learning algorithm becomes very high for learning schema mappings of type complex that contains 200 relational-to-RDF temporal GAV constraints. It takes around 30 minutes even when $|E| = 5$. ten Cate et al. [64] conducted experiments on the active learning algorithm for the relational-to-RDF temporal schema mappings, and the running time of the active learning algorithm ranges from $0.3s$ to $15m5s$. Such a difference in the running time is caused by the number of constraints we have in each schema mapping and Allen's relations that we considered in our learning process. In that paper, there are at most 30 constraints in each relational-to-relational temporal GAV schema mapping. Table 5.2 recorded experiments on five schema mappings of the type simple, where on average, there are 58.4 relational-to-RDF temporal GAV constraints in each schema mapping. Table 5.3 recorded experiments on five schema mappings of the type moderate, where on average, there are 123.2 relational-to-RDF temporal GAV constraints in each schema mapping. More information is reported in Table 5.1. Most importantly, the computation of critically sound constraints is the bottleneck for the active learning algorithm. In our active learning algorithm for relational-to-RDF temporal schema mapping, we first add all possible Allen's relations in each candidate constraint, which causes the size of the candidate constraint to increase by polynomial. Assume that there are m temporal atoms in the left-hand side of the candidate constraint. Now we will add m^2 number of possible Allen's relations to the left-hand side. Thus the size of the left-hand side of the candidate constraint becomes $O(n^2)$. Hence, the time for calculating a critically sound constraint increases significantly.

Summary Our experiment results show that $\overline{Recall}$ values are higher when $\overline{Rep}$ values increase. The value of $\overline{Recall}$ could reach 0.999, and the value of F_s could also reach 0.999. It indicates that our active learning algorithm is a good learning algorithm for the relational-to-RDF temporal schema mapping. Moreover, we observe that the running time depends on the size of the training set and the size of the goal schema mappings. Specifically, the running time of the learning algorithm is higher with larger source instances controlled by the parameter α, higher numbers of data examples, and increased complexity of the goal schema mappings. It is worth mentioning that the computation of critically sound constraints with large training set and large goal schema mapping might lead to bottlenecks. However, in practice, the number of data examples and the number of constraints in goal schema mappings are typically within reasonable limits. Therefore, the running time of our active learning algorithm remains reasonable.

Chapter 6

Concluding Remarks

In the first part of our work, we contributed to the development of relational-to-relational temporal data exchange, an area that had remained largely unexplored. Our main emphasis has been on the properties of universal solutions in the concrete model of time. We first formalized the concept of a concrete universal solution and presented a version of the chase algorithm, called the concrete chase algorithm that produces concrete universal solutions, provided the algorithm does not fail. After that, we identified certain sufficient conditions that guarantee that if the concrete chase algorithm fails, then no concrete solutions exist. We then focused on the pursuit of semantically adequate concrete universal solutions. We showed that such solutions may not exist even for temporal schema mappings with a single temporal variable. Furthermore, we identified sufficient conditions that guarantee the existence of semantically adequate concrete universal solutions for normalized concrete source instances.

Drawing from our investigation into temporal data exchange between relational databases, we embarked on a study concerning the problem of data exchange for temporal data from relational databases into RDF-expressed ontologies. Within this context, we defined the notion of a full schema mapping and a GAV schema mapping, which takes a *temporal RDF graph schema* as its target schema. Subsequently, we designed a chase algorithm to produce a universal solution for a given temporal relational database w.r.t. a given schema mapping.

The last part of our work mainly contributes to the derivation of relational-to-RDF temporal schema mappings by an active learning algorithm based on a labeling oracle and a set of universal data examples. Most existing related work focuses on deriving schema mappings for the relational-to-relational data exchange problem, which do not consider temporal information. We developed an active learning algorithm by applying conformance testing, which takes a labeling oracle and a set of universal data examples of a goal schema mapping as inputs and returns a relational-to-RDF temporal schema mapping which is logically equivalent to the goal schema mapping. To verify the effectiveness and efficiency of our active learning algorithm, we developed *Temporal iBench* as a tool to generate relational-to-RDF temporal schema mappings and universal data examples of the schema mappings. Finally, we carried out an experimental evaluation and demonstrated the effectiveness and efficiency of our active learning algorithm.

We conclude by describing three directions for further research in the area of relational-to-relational temporal data exchange: semantic adequacy for temporal schema mappings with multiple temporal variables, temporal data exchange with target tuple-generating dependencies, and temporal data exchange with existentially quantified temporal variables.

Semantic Adequacy for Temporal Schema Mappings with Multiple Temporal Variables

In the relational-to-relational temporal data exchange problem, we studied the existence of semantically adequate concrete universal solutions for schema mappings with at most one temporal variable in each constraint (see Section 3.2.3). The next step would be to investigate the existence of semantically adequate concrete universal solutions for temporal schema mappings with multiple temporal variables.

Recall that in Section 3.2.3, every concrete schema mapping $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st}, \Sigma_t)$ gives rise to a schema mapping that is meaningful in the abstract model of time, called an abstract schema mapping $\mathcal{M}^a = (\mathbf{S}, \mathbf{T}, \Sigma_{st}^a, \Sigma_t^a)$, where Σ_{st}^a and Σ_t^a are converted from Σ_{st} and Σ_t, respectively. Let I be a concrete source instance. We have that a concrete target instance J is semantically adequate for I if the abstract target instance $[\![J]\!]$ is a universal solution for $[\![I]\!]$ w.r.t. $\mathcal{M}^a$. We showed that if a schema mapping contains at most one temporal variable per constraint, then it is meaningful in both the concrete model of time and the abstract model of time without changing the constraints in the schema mapping, i.e., $\Sigma_{st} = \Sigma_{st}^a$, $\Sigma_t = \Sigma_t^a$, and hence $\mathcal{M} = \mathcal{M}^a$.

However, in the presence of multiple temporal variables, such a schema mapping is only meaningful in the abstract model of time with changing the constraints in the schema mapping. Therefore, we are facing three challenges:

Challenge 1: Concrete s-t tgds and concrete target egds must be converted to "essentially equivalent" abstract s-t tgds and to abstract target egds, respectively, because the concrete ones involve Allen's relations while the abstract ones involve suitable formulas of first-order logic over time points compared with the $<$ relation. The conversion requires thorough exploration and careful design. If σ is a concrete s-t tgd or a concrete target egd, then we write $a(\sigma)$ for the s-t tgd or the target egd in the abstract model of time resulting from σ via a conversion. As a result, the set $\Sigma_{st}^a = \{a(\sigma) : \sigma \in \Sigma_{st}\}$ and the set $\Sigma_t^a = \{a(\sigma) : \sigma \in \Sigma_t\}$.

Challenge 2: Suitable definitions are required for the languages of an abstract s-t tgd and an abstract target egd, such that the abstract s-t tgds and the abstract target egds produced by a well-designed conversion are in the defined languages. As stated previously, the $<$ relation should be taken into account in the languages.

Challenge 3: Another challenge arises when we try to apply abstract schema mappings into the snapshot chase algorithm. In Section 3.2.2, a database in the abstract model of time is regarded as a set of snapshots; yet, temporal relationships between different snapshots are not defined, i.e., atoms such as $t_1^- \leq t_1 \leq t_1^+$ can not be applied to snapshots. One solution is to design an abstract chase algorithm such that if the abstract chase algorithm does not fail, it generates a universal solution for a given abstract source instance w.r.t. an abstract schema mapping; otherwise, no solution exists.

Temporal Data Exchange with Target Tuple-generating Dependencies Explore temporal data exchange for schema mappings that contain *target tuple-generating dependencies*. Several challenges arise in this case, including the translation of the constraints from the concrete model of time to the abstract model of time, the management of time-stamped nulls, and the design of a suitable chase algorithm.

A temporal target tuple-generating dependency, called *temporal target tgd*, is a first-order sentence of the form: $\forall \mathbf{x}(\varphi(\mathbf{x},\mathbf{t}) \wedge \rho'(\mathbf{t}) \rightarrow \exists \mathbf{y}\psi(\mathbf{x},\mathbf{y},\mathbf{t}))$, where $\varphi(\mathbf{x},\mathbf{t})$ and $\psi(\mathbf{x},\mathbf{y},\mathbf{t}))$ are two conjunctions of target atoms, and $\rho'(\mathbf{t})$ is a Boolean combination Allen atomic formulas involving variables from $\mathbf{t}$.

A *relational-to-relational temporal schema mapping* is a tuple $\mathcal{M} = (\mathbf{S},\mathbf{T},\Sigma_{st},\Sigma_t)$, where Σ_{st} is a finite set of temporal s-t tgds, and Σ_t is a finite set of temporal target tgds or temporal target egds.

When dealing with such temporal schema mappings, we will encounter similar challenges that were mentioned in the first research direction. Additionally, another challenge arises in designing a suitable concrete chase algorithm. The original paper [32] on relational-to-relational data exchange allowed standard target tgds and, furthermore, identified a structural condition, called *weakly acyclicity*, on the standard schema mappings, which is a sufficient condition that yields efficient algorithms for determining whether, given a source instance I, a universal solution for I exists. Target tgds have not been studied in temporal data exchange so far as the handling of time-stamped nulls for such constraints becomes more complicated since additional time-stamped nulls may have to be generated (and perhaps equated with other time-stamped nulls later on).

Temporal Data Exchange with Existentially Quantified Temporal Variables Explore the temporal data exchange problem for schema mappings in which the constraints have existentially quantified temporal variables. Several challenges of different nature arise in this case, some of which are similar to the challenges in answering queries over temporal data with the help of ontologies (see [17] for a comprehensive survey of that area).

Example 18. Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$ be a *relational-to-relational* temporal schema mapping, where the source schema $\mathbf{S}$ has a relation *Hur* with attributes *h-name*, *loc*, *time* for hurricane hitting a location with a farm and a relation *Farm* with attributes *f-name* and *loc*; the target schema $\mathbf{T}$ have a relation *Flood* for the flood and a relation *Damage* with attributes *farms*, *h-name*, *damage*, and *time*; the set Σ contains two temporal s-t tgds and a temporal target egd:

$$\forall n,c,f,t(Hur(n,c,t) \wedge Farm(f,c) \rightarrow \exists d\ (Damage(f,n,d,t))),$$

$$\forall n,c,t(Hur(n,c,t) \rightarrow \exists t'((t \text{ precedes } t') \wedge Flood(c,t'))).$$

$$\forall n,f,d,d',t(Damage(f,n,d,t) \wedge Damage(f,n,d',t) \rightarrow d = d').$$

The first s-t tgd asserts that if there is a hurricane n hitting a farm f in location c during the time interval t, then there exists damage of cost d at the farm f caused by the hurricane n. The second s-t tgd asserts that if there is a hurricane n hitting a location c during the time interval t, then there exists a time interval t' after t, such that there is a flood happening in c during t'. The third constraint is a target egd that asserts that if there is a damage of cost d at the farm f caused by hurricane n during time t and a damage cost d' of the farm f caused by hurricane n, then the cost d must equal the cost d'.

Observe that the first s-t tgd of $\mathcal{M}$ contains an existentially quantified general variable d (ranging over cost values). In contrast, the second s-t tgd contains an existentially quantified temporal variable t. In the target instance, the variable d will be populated by a null annotated by a time-stamped null, while the variable t will be populated by an anonymous time. By using the proof techniques in the paper [16, 56], we have been able to show that there is a source instance I for which *no* universal solution for I w.r.t. $\mathcal{M}$ exists. This finding indicates new possible challenges may arise when considering existentially quantified temporal variables. Moreover, the target egd listed above may force two time-stamped nulls to be equated. Equating two time-stamped nulls with different time stamps is a delicate matter that requires a special determination which of the two nulls should be replaced by the other when populating the target instance. Golshanara and Chomicki [36] avoided this issue by restricting the class of schema mappings considered, at the expense of having a rather limited framework for temporal data exchange.

Bibliography

[1] Bogdan Alexe, Balder ten Cate, Phokion G. Kolaitis, and Wang Chiew Tan. Designing and refining schema mappings via data examples. In Timos K. Sellis, Renée J. Miller, Anastasios Kementsietsidis, and Yannis Velegrakis, editors, *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2011, Athens, Greece, June 12-16, 2011*, pages 133–144. ACM, 2011. doi: 10.1145/1989323.1989338. URL `https://doi.org/10.1145/1989323.1989338`.

[2] Bogdan Alexe, Balder ten Cate, Phokion G. Kolaitis, and Wang Chiew Tan. Characterizing schema mappings via data examples. *ACM Transactions on Database Systems*, 36(4): 23:1–23:48, 2011. doi: 10.1145/2043652.2043656. URL `https://doi.org/10.1145/2043652.2043656`.

[3] Dean Allemang and James Hendler. *Semantic Web for the Working Ontologist: Effective Modeling in RDFS and OWL.* Morgan Kaufmann, 2nd edition, 2011.

[4] James F. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843, 1983. doi: 10.1145/182.358434. URL `http://doi.acm.org/10.1145/182.358434`.

[5] Shun'ichi Amano, Leonid Libkin, and Filip Murlak. XML schema mappings. In Jan Paredaens and Jianwen Su, editors, *Proceedings of the Twenty-Eigth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2009, June 19 - July 1, 2009, Providence, Rhode Island, USA*, pages 33–42. ACM, 2009. doi: 10.1145/1559795.1559801. URL `https://doi.org/10.1145/1559795.1559801`.

[6] Dana Angluin. Learning regular sets from queries and counterexamples. *Information and Computation*, 75(2):87–106, 1987. doi: 10.1016/0890-5401(87)90052-6. URL `https://doi.org/10.1016/0890-5401(87)90052-6`.

[7] Dana Angluin. Queries and concept learning. *Machine Learning*, 2(4):319–342, 1987. doi: 10.1007/BF00116828. URL `https://doi.org/10.1007/BF00116828`.

[8] Grigoris Antoniou, Paul T. Groth, Frank van Harmelen, and Rinke Hoekstra. *A Semantic Web Primer, 3rd Edition*. MIT Press, 2012. ISBN 978-0-262-01828-9.

[9] Jing Ao, Zehui Cheng, Rada Chirkova, and Phokion G. Kolaitis. Temporal enrichment and querying of ontology-compliant data. In Jérôme Darmont, Boris Novikov, and Robert Wrembel, editors, *New Trends in Databases and Information Systems - ADBIS 2020 Short Papers, Lyon, France, August 25-27, 2020, Proceedings*, volume 1259 of *Communications in Computer and Information Science*, pages 129–139. Springer, 2020. doi: 10.1007/978-3-030-54623-6_12. URL `https://doi.org/10.1007/978-3-030-54623-6_12`.

[10] Jing Ao, Zehui Cheng, Rada Chirkova, and Phokion G. Kolaitis. Theory and practice

of relational-to-rdf temporal data exchange and query answering. *Journal of Data and Information Quality*, 15(2):1–27, 2023. doi: https://doi.org/10.1145/3591359.

[11] Marcelo Arenas and Leonid Libkin. XML data exchange: Consistency and query answering. *Journal of the ACM*, 55(2):7:1–7:72, 2008. doi: 10.1145/1346330.1346332. URL `https://doi.org/10.1145/1346330.1346332`.

[12] Marcelo Arenas, Jorge Pérez, Juan L. Reutter, and Cristian Riveros. Composition and inversion of schema mappings. *SIGMOD Record*, 38(3):17–28, 2009. doi: 10.1145/1815933.1815938. URL `https://doi.org/10.1145/1815933.1815938`.

[13] Marcelo Arenas, Pablo Barceló, Leonid Libkin, and Filip Murlak. *Relational and XML Data Exchange*. Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2010. doi: 10.2200/S00297ED1V01Y201008DTM008. URL `https://doi.org/10.2200/S00297ED1V01Y201008DTM008`.

[14] Marcelo Arenas, Pablo Barceló, Leonid Libkin, and Filip Murlak. *Foundations of Data Exchange*. Cambridge University Press, 2014. ISBN 9781107016163. URL `http://www.cambridge.org/9781107016163`.

[15] Patricia C. Arocena, Boris Glavic, Radu Ciucanu, and Renée J. Miller. The ibench integration metadata generator. *PVLDB*, 9(3):108–119, 2015.

[16] Alessandro Artale, Roman Kontchakov, Frank Wolter, and Michael Zakharyaschev. Temporal description logic for ontology-based data access. In Francesca Rossi, editor, *IJCAI 2013, Proceedings of the 23rd International Joint Conference on Artificial Intel-*

ligence, Beijing, China, August 3-9, 2013, pages 711–717. IJCAI/AAAI, 2013. URL http://www.aaai.org/ocs/index.php/IJCAI/IJCAI13/paper/view/6824.

[17] Alessandro Artale, Roman Kontchakov, Alisa Kovtunova, Vladislav Ryzhikov, Frank Wolter, and Michael Zakharyaschev. Ontology-mediated query answering over temporal data: A survey (invited talk). In Sven Schewe, Thomas Schneider, and Jef Wijsen, editors, *24th International Symposium on Temporal Representation and Reasoning, TIME 2017, October 16-18, 2017, Mons, Belgium*, volume 90 of *LIPIcs*, pages 1:1–1:37. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017. doi: 10.4230/LIPIcs.TIME.2017.1. URL https://doi.org/10.4230/LIPIcs.TIME.2017.1.

[18] Pablo Barceló, Jorge Pérez, and Juan L. Reutter. Schema mappings and data exchange for graph databases. In Wang-Chiew Tan, Giovanna Guerrini, Barbara Catania, and Anastasios Gounaris, editors, *Joint 2013 EDBT/ICDT Conferences, ICDT '13 Proceedings, Genoa, Italy, March 18-22, 2013*, pages 189–200. ACM, 2013. ISBN 978-1-4503-1598-2. doi: 10.1145/2448496.2448520. URL http://doi.acm.org/10.1145/2448496.2448520.

[19] Catriel Beeri and Moshe Y. Vardi. Formal systems for tuple and equality generating dependencies. *SIAM Journal on Computing*, 13(1):76–98, 1984. doi: 10.1137/0213006. URL https://doi.org/10.1137/0213006.

[20] Philip A. Bernstein. Applying model management to classical meta data problems. In *First Biennial Conference on Innovative Data Systems Research, CIDR 2003, Asilomar, CA, USA, January 5-8, 2003, Online Proceedings*. www.cidrdb.org, 2003. URL http://www-db.cs.wisc.edu/cidr/cidr2003/program/p19.pdf.

[21] Chris Bizer, Richard Cyganiak, Luís Eufrasio, Teixeira Neto, Hannes Mühleisen, Aftab Iqbal, Jacobus Geluk, Giovanni Mels, David Venable, Christian Becker, Olaf Hartig, Andreas Langegger, Herwig Leimer, Inigo Surguy, Oliver Maresch, and Jörg Garbers. D2rq, 2012. URL `http://d2rq.org/`.

[22] Iovka Boneva, Jose Lozano, and Slawomir Staworko. Relational to RDF data exchange in presence of a shape expression schema. In Dan Olteanu and Barbara Poblete, editors, *Proceedings of the 12th Alberto Mendelzon International Workshop on Foundations of Data Management, Cali, Colombia, May 21-25, 2018*, volume 2100 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2018. URL `http://ceur-ws.org/Vol-2100/paper6.pdf`.

[23] Iovka Boneva, Jérémie Dusart, Daniel Fernández-Álvarez, and José Emilio Labra Gayo. Shape designer for shex and SHACL constraints. In Mari Carmen Suárez-Figueroa, Gong Cheng, Anna Lisa Gentile, Christophe Guéret, C. Maria Keet, and Abraham Bernstein, editors, *Proceedings of the ISWC 2019 Satellite Tracks (Posters & Demonstrations, Industry, and Outrageous Ideas) co-located with 18th International Semantic Web Conference (ISWC 2019), Auckland, New Zealand, October 26-30, 2019*, volume 2456 of *CEUR Workshop Proceedings*, pages 269–272. CEUR-WS.org, 2019. URL `https://ceur-ws.org/Vol-2456/paper70.pdf`.

[24] Iovka Boneva, Slawek Staworko, and Jose Lozano. Consistency and certain answers in relational to RDF data exchange with shape constraints. In Jérôme Darmont, Boris Novikov, and Robert Wrembel, editors, *New Trends in Databases and Information Systems*

- *ADBIS 2020 Short Papers, Lyon, France, August 25-27, 2020, Proceedings*, volume 1259 of *Communications in Computer and Information Science*, pages 97–107. Springer, 2020.

[25] Angela Bonifati, Ugo Comignani, Emmanuel Coquery, and Romuald Thion. Interactive mapping specification with exemplar tuples. *ACM Transactions on Database Systems*, 44(3): 10:1–10:44, 2019. doi: 10.1145/3321485. URL `https://doi.org/10.1145/3321485`.

[26] Dan Brickley and Ramanathan Guha. RDF schema 1.1. W3C recommendation, W3C, February 2014. https://www.w3.org/TR/2014/REC-rdf-schema-20140225/.

[27] Andrea Calì, Georg Gottlob, and Thomas Lukasiewicz. Datalog$^{\pm}$: a unified approach to ontologies and integrity constraints. In Ronald Fagin, editor, *Database Theory - ICDT 2009, 12th International Conference, St. Petersburg, Russia, March 23-25, 2009, Proceedings*, volume 361 of *ACM International Conference Proceeding Series*, pages 14–30. ACM, 2009. doi: 10.1145/1514894.1514897. URL `https://doi.org/10.1145/1514894.1514897`.

[28] Zehui Cheng and Phokion G. Kolaitis. Universal solutions in temporal data exchange. In Emilio Muñoz-Velasco, Ana Ozaki, and Martin Theobald, editors, *27th International Symposium on Temporal Representation and Reasoning, TIME 2020, September 23-25, 2020, Bozen-Bolzano, Italy*, volume 178 of *LIPIcs*, pages 8:1–8:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. doi: 10.4230/LIPIcs.TIME.2020.8. URL `https://doi.org/10.4230/LIPIcs.TIME.2020.8`.

[29] Zehui Cheng and Phokion G. Kolaitis. Universal solutions for temporal data exchange.

Information and Computation, 281:104793, 2021. doi: 10.1016/j.ic.2021.104793. URL `https://doi.org/10.1016/j.ic.2021.104793`.

[30] Jan Chomicki and David Toman. Temporal databases. In Michael Fisher, Dov M. Gabbay, and Lluís Vila, editors, *Handbook of Temporal Reasoning in Artificial Intelligence*, volume 1 of *Foundations of Artificial Intelligence*, pages 429–467. Elsevier, 2005. ISBN 978-0-444-51493-6. doi: 10.1016/S1574-6526(05)80016-1. URL `https://doi.org/10.1016/S1574-6526(05)80016-1`.

[31] Ronald Fagin. Inverting schema mappings. *ACM Transactions on Database Systems*, 32(4): 25, 2007. doi: 10.1145/1292609.1292615. URL `https://doi.org/10.1145/1292609.1292615`.

[32] Ronald Fagin, Phokion G. Kolaitis, Renée J. Miller, and Lucian Popa. Data exchange: semantics and query answering. *Theoretical Computer Science*, 336(1):89–124, 2005. doi: 10.1016/j.tcs.2004.10.033. URL `https://doi.org/10.1016/j.tcs.2004.10.033`.

[33] Ronald Fagin, Phokion G. Kolaitis, and Lucian Popa. Data exchange: getting to the core. *ACM Transactions on Database Systems*, 30(1):174–210, 2005. doi: 10.1145/1061318.1061323. URL `https://doi.org/10.1145/1061318.1061323`.

[34] Ronald Fagin, Phokion G. Kolaitis, Lucian Popa, and Wang Chiew Tan. Composing schema mappings: Second-order dependencies to the rescue. *ACM Transactions on Database Systems*, 30(4):994–1055, 2005. doi: 10.1145/1114244.1114249. URL `https://doi.org/10.1145/1114244.1114249`.

[35] Nadime Francis and Leonid Libkin. Schema mappings for data graphs. In Emanuel Sallinger, Jan Van den Bussche, and Floris Geerts, editors, *Proceedings of the 36th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 2017, Chicago, IL, USA, May 14-19, 2017*, pages 389–401. ACM, 2017. doi: 10.1145/3034786.3056113. URL `https://doi.org/10.1145/3034786.3056113`.

[36] Ladan Golshanara and Jan Chomicki. Temporal data exchange. *Information Systems*, 87, 2020. doi: 10.1016/j.is.2019.07.004. URL `https://doi.org/10.1016/j.is.2019.07.004`.

[37] Georg Gottlob and Pierre Senellart. Schema mapping discovery from data instances. *Journal of the ACM*, 57(2):6:1–6:37, 2010. doi: 10.1145/1667053.1667055. URL `https://doi.org/10.1145/1667053.1667055`.

[38] Gösta Grahne and Adrian Onet. Anatomy of the chase. *Fundamenta Informaticae*, 157 (3):221–270, 2018. doi: 10.3233/FI-2018-1627. URL `https://doi.org/10.3233/FI-2018-1627`.

[39] Ramanathan Guha and Dan Brickley. RDF vocabulary description language 1.0: RDF schema. W3C recommendation, W3C, February 2004. https://www.w3.org/TR/2004/REC-rdf-schema-20040210/.

[40] Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2022. URL `https://www.gurobi.com`.

[41] Claudio Gutiérrez, Carlos A. Hurtado, and Alejandro A. Vaisman. Introducing time into

RDF. *IEEE Transactions on Knowledge and Data Engineering*, 19(2):207–218, 2007. doi: 10.1109/TKDE.2007.34. URL `https://doi.org/10.1109/TKDE.2007.34`.

[42] Claudio Gutiérrez, Carlos A. Hurtado, Alberto O. Mendelzon, and Jorge Pérez. Foundations of semantic web databases. *Journal of Computer and System Sciences*, 77(3):520–541, 2011. doi: 10.1016/j.jcss.2010.04.009. URL `https://doi.org/10.1016/j.jcss.2010.04.009`.

[43] Aidan Hogan. *The Web of Data*. Springer, 2020. ISBN 978-3-030-51579-9. doi: 10.1007/978-3-030-51580-5. URL `https://doi.org/10.1007/978-3-030-51580-5`.

[44] David S. Johnson and Anthony C. Klug. Testing containment of conjunctive queries under functional and inclusion dependencies. In Jeffrey D. Ullman and Alfred V. Aho, editors, *Proceedings of the ACM Symposium on Principles of Database Systems, March 29-31, 1982, Los Angeles, California, USA*, pages 164–169. ACM, 1982. doi: 10.1145/588111.588138. URL `https://doi.org/10.1145/588111.588138`.

[45] Krishna G. Kulkarni and Jan-Eike Michels. Temporal features in SQL: 2011. *SIGMOD Record*, 41(3):34–43, 2012. doi: 10.1145/2380776.2380786. URL `https://doi.org/10.1145/2380776.2380786`.

[46] Maurizio Lenzerini. Data integration: A theoretical perspective. In Lucian Popa, Serge Abiteboul, and Phokion G. Kolaitis, editors, *Proceedings of the Twenty-first ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 3-5, Madison,*

Wisconsin, USA, pages 233–246. ACM, 2002. doi: 10.1145/543613.543644. URL `https://doi.org/10.1145/543613.543644`.

[47] David Maier, Alberto O. Mendelzon, and Yehoshua Sagiv. Testing implications of data dependencies. *ACM Transactions on Database Systems*, 4(4):455–469, 1979. doi: 10.1145/320107.320115. URL `https://doi.org/10.1145/320107.320115`.

[48] Alberto O. Mendelzon. Database states and their tableaux. *ACM Transactions on Database Systems*, 9(2):264–282, 1984. doi: 10.1145/329.318579. URL `https://doi.org/10.1145/329.318579`.

[49] Franck Michel, Johan Montagnat, and Catherine Faron Zucker. A survey of RDB to RDF translation approaches and tools, 2014. Rapport de Recherche ISRN I3S/RR 2013-04-FR.

[50] Alan Nash, Philip A. Bernstein, and Sergey Melnik. Composition of mappings given by embedded dependencies. *ACM Transactions on Database Systems*, 32(1):4, 2007. doi: 10.1145/1206049.1206053. URL `https://doi.org/10.1145/1206049.1206053`.

[51] National Academies of Sciences Engineering Medicine et al. *Combating Antimicrobial Resistance: A One Health Approach to a Global Threat: Proceedings of a Workshop*. National Academies Press, 2017.

[52] Adrian Onet. The chase procedure and its applications in data exchange. In Phokion G. Kolaitis, Maurizio Lenzerini, and Nicole Schweikardt, editors, *Data Exchange, Integration, and Streams*, volume 5 of *Dagstuhl Follow-Ups*, pages 1–37. Schloss Dagstuhl - Leibniz-

Zentrum für Informatik, 2013. doi: 10.4230/DFU.Vol5.10452.1. URL `https://doi.org/10.4230/DFU.Vol5.10452.1`.

[53] Eric Prud'hommeaux, Alexandre Bertails, Juan Sequeda, and Marcelo Arenas. A direct mapping of relational data to RDF. W3C recommendation, W3C, September 2012. https://www.w3.org/TR/2012/REC-rdb-direct-mapping-20120927/.

[54] rdfsemw3c. RDF Semantics: W3C recommendation 10 February 2004, 2004. Hayes, Patrick (Ed.), `https://www.w3.org/TR/2004/REC-rdf-mt-20040210/`.

[55] rdfsw3c. RDF Vocabulary Description Language: RDFS, 2004. Brickley, D. and Guha, R.V. (Eds), `https://www.researchgate.net/publication/319395331_RDF_Vocabulary_Description_Language_10_RDF_Schema`.

[56] Andrea Schaerf. On the complexity of the instance checking problem in concept languages with existential quantification. *Journal of Intelligent Information Systems*, 2(3):265–278, 1993. doi: 10.1007/BF00962071. URL `https://doi.org/10.1007/BF00962071`.

[57] Alexander Schrijver. *Theory of linear and integer programming*. Wiley-Interscience series in discrete mathematics and optimization. Wiley, 1999. ISBN 978-0-471-98232-6.

[58] Abdullah Uz Tansel, James Clifford, Shashi K. Gadia, Sushil Jajodia, Arie Segev, and Richard T. Snodgrass, editors. *Temporal Databases: Theory, Design, and Implementation*. Benjamin/Cummings, 1993. ISBN 0-8053-2413-5.

[59] Jonas Tappolet and Abraham Bernstein. Applied temporal RDF: efficient temporal querying of RDF data with SPARQL. In Lora Aroyo, Paolo Traverso, Fabio Ciravegna, Philipp

Cimiano, Tom Heath, Eero Hyvönen, Riichiro Mizoguchi, Eyal Oren, Marta Sabou, and Elena Paslaru Bontas Simperl, editors, *The Semantic Web: Research and Applications, 6th European Semantic Web Conference, ESWC 2009, Heraklion, Crete, Greece, May 31-June 4, 2009, Proceedings*, volume 5554 of *Lecture Notes in Computer Science*, pages 308–322. Springer, 2009. doi: 10.1007/978-3-642-02121-3_25. URL `https://doi.org/10.1007/978-3-642-02121-3_25`.

[60] Balder ten Cate and Phokion G. Kolaitis. Structural characterizations of schema-mapping languages. *Communications of the ACM*, 53(1):101–110, 2010. doi: 10.1145/1629175.1629201. URL `https://doi.org/10.1145/1629175.1629201`.

[61] Balder ten Cate, Phokion G. Kolaitis, and Wang Chiew Tan. Database constraints and homomorphism dualities. In David Cohen, editor, *Principles and Practice of Constraint Programming - CP 2010 - 16th International Conference, CP 2010, St. Andrews, Scotland, UK, September 6-10, 2010. Proceedings*, volume 6308 of *Lecture Notes in Computer Science*, pages 475–490. Springer, 2010. doi: 10.1007/978-3-642-15396-9_38. URL `https://doi.org/10.1007/978-3-642-15396-9_38`.

[62] Balder ten Cate, Víctor Dalmau, and Phokion G. Kolaitis. Learning schema mappings. *ACM Transactions on Database Systems*, 38(4):28:1–28:31, 2013. doi: 10.1145/2539032.2539035. URL `https://doi.org/10.1145/2539032.2539035`.

[63] Balder ten Cate, Phokion G. Kolaitis, Kun Qian, and Wang-Chiew Tan. Approximation algorithms for schema-mapping discovery from data examples. *ACM Transactions on*

Database Systems, 42(2):12:1–12:41, 2017. doi: 10.1145/3044712. URL `https://doi.org/10.1145/3044712`.

[64] Balder ten Cate, Phokion G. Kolaitis, Kun Qian, and Wang-Chiew Tan. Active learning of GAV schema mappings. In Jan Van den Bussche and Marcelo Arenas, editors, *Proceedings of the 37th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, Houston, TX, USA, June 10-15, 2018*, pages 355–368. ACM, 2018. doi: 10.1145/3196959.3196974. URL `https://doi.org/10.1145/3196959.3196974`.

[65] David Toman. Point vs. interval-based query languages for temporal databases. In Richard Hull, editor, *Proceedings of the Fifteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 3-5, 1996, Montreal, Canada*, pages 58–67. ACM Press, 1996. ISBN 0-89791-781-2. doi: 10.1145/237661.237676. URL `http://doi.acm.org/10.1145/237661.237676`.

[66] Jeffrey D. Ullman. Corrigendum: The theory of joins in relational databases. *ACM Transactions on Database Systems*, 8(2):287, 1983.

[67] Frits W. Vaandrager. Model learning. *Communications of the ACM*, 60(2):86–95, 2017. doi: 10.1145/2967606. URL `https://doi.org/10.1145/2967606`.

[68] Moshe Y. Vardi. Inferring multivalued dependencies from functional and join dependencies. *Acta Informatica*, 19:305–324, 1983. doi: 10.1007/BF00290729. URL `https://doi.org/10.1007/BF00290729`.

[69] Antoine Zimmermann, Nuno Lopes, Axel Polleres, and Umberto Straccia. A general

framework for representing, reasoning and querying with annotated Semantic Web data. *Journal of Web Semantics*, 11:72–95, 2012.