Temporal Enrichment and Querying of Ontology-Compliant Data

Jing Ao[1], Zehui Cheng[2], Rada Chirkova[1], and Phokion G. Kolaitis[2]

[1] NC State University, Raleigh NC 27695, USA {jao,rychirko}@ncsu.edu
[2] UC Santa Cruz, Santa Cruz CA 95064, USA
{zecheng,kolaitis}@ucsc.edu

Abstract. We consider the problem of answering temporal queries on RDF stores, in presence of time-agnostic RDFS domain ontologies, of relational data sources that include temporal information, and of rules that map the domain information in the source schemas into the target ontology. Our proposed solution consists of two rule-based domain-independent algorithms. The first algorithm materializes target RDF data via a version of data exchange that enriches both the data and the ontology with temporal information from the relational sources. The second algorithm accepts as inputs temporal queries expressed in terms of the domain ontology, using SPARQL supplemented with a lightweight easy-to-use formalism for time annotations and comparisons. The algorithm translates the queries into the standard SPARQL form that respects the structure of the temporal RDF information while preserving the semantics of the questions, thus ensuring successful evaluation of the queries on the materialized temporally-enriched RDF data. In this paper we present the algorithms, report on their implementation and experimental results for two application domains, and discuss future work.

Keywords: Data-intensive sciences and databases; Temporal databases; Data exchange; RDF / RDFS / SPARQL

1 Introduction

In application domains that span industry, government, science, and global health, data are often collected independently by different teams over time. As the needs of the various data-collecting entities evolve, it is often the case that data from multiple *sources* must be put together under a unified *target* format *(exchanged* [4,14]), using *source-to-target (s-t) rules* developed by domain experts for the purpose. In many real-life applications, there is an added requirement that the target data format be aligned with the standard domain vocabulary. Such vocabularies are developed by experts under the name of domain *ontologies,* with the information including concepts and relationships, as well as domain rules governing their interaction. Our exposition will focus on a common real-life scenario, in which ontologies and ontology-compliant data are expressed using the *RDF/S* capabilities – those of the Resource Description Framework *(RDF)*

data model [26] enriched with additional *RDFS* specifications [27], – and are queried using SPARQL [24,28], while the source data are relational.

In applications conforming to this relational-to-RDF/S data-exchange scenario, e.g., in studies of antimicrobial resistance *(AMR)* [11], the source data may contain important temporal information, while the applicable target domain ontologies lack temporal components. (In AMR this is the case with the Antibiotic Resistance Ontology (ARO).[3]) At the same time, solutions that are available for data exchange in the relational-to-RDF/S scenario, see, e.g., [22], do not directly apply here, as they do not incorporate temporal semantics of the data in easy-to-use ways. As a result, temporal information from the sources can be lost in the exchange process, making it hard or even impossible for domain scientists to efficiently obtain correct answers to temporal queries posed on the contents of the source data in terms of the target ontologies. This problem can be addressed on a case-by-case basis, by using in the data exchange manually designed temporally enriched individual domain ontologies, as well as "temporally aware" s-t rules that would be specifically developed for use with such ontologies. Such custom solutions, however, would delegate to data analysts or domain scientists the nontrivial task of temporally enhancing the originally time-agnostic domain ontologies, such as ARO. As an additional burden on the users, to be able to formulate correctly their temporal queries, domain analysts or experts would need to be aware of how the temporal information is modeled and represented in the resulting systems.

Contributions: In this paper, in the context of relational-to-RDF/S data exchange, we consider the scenario in which domain analysts and scientists are interested in obtaining answers to *temporal* queries formulated in terms of the given *time-agnostic* target domain ontology, with the expectation that the temporal information in the query answers would come from the data sources. We assume that the users posing the queries are expert, rather than casual, users, in the sense that they are familiar with formulating SPARQL queries using the given RDFS ontology. We also assume that the expert users provide the s-t rules that map the domain information in the source schemas into the time-agnostic target ontology. In this scenario, we propose an approach that

- Enables users to formulate SPARQL-based temporal queries, and
- Returns to them answers to the queries, using the domain information enabled in the target by the s-t rules, with the temporal dimension of that information coming from the sources via temporal enrichment.

Our declarative domain-independent approach for achieving these goals focuses on separating temporal semantics from the domain semantics, and comprises two algorithms. The first algorithm materializes target RDF data via a version of data exchange that builds on the given s-t rules to enrich both the data and the ontology with temporal information from the sources. The second algorithm accepts as inputs temporal queries expressed in terms of the target ontology, using SPARQL supplemented with a lightweight easy-to-use formalism for time annotations and comparisons. The algorithm translates queries into

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

the standard SPARQL form that respects the structure of the temporal RDF information while preserving the semantics of the questions, thus ensuring successful evaluation of the queries on the materialized temporally-enriched RDF data. In this paper we present the algorithms, report on their implementation and experimental results for two application domains, and discuss future work.

Related Work: RDFS [27] is a language used in practice for describing ontologies, see [1] and the Protege Ontology Library[4] for RDFS-based ontology examples. The need for temporal annotations and reasoning arises in many application domains; toward addressing the need, Gutierrez et al. in [18] defined temporal RDF and studied properties of inference in temporal graphs. Another approach, which focuses on querying, is presented in [29]. For the RDFS layer, research has been done on inference of temporal properties in temporal ontologies, see, e.g., [32]. At the same time, the temporal aspect is usually not included in the practical development of domain ontologies; our proposed approach in this paper is designed to bridge this gap. In particular, we preserve the temporal source semantics in the target ontology-compliant domain data, by enabling Allen interval relations [2], such as `during` or `before`, in queries on the data.

Data exchange has been studied extensively in the relational setting, see [4,14]. Recently, considerable formal work has been done on temporal ontology-mediated query access (OMQA), see [5]. OMQA differs in its objectives from data exchange, on which we focus in this paper, 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. There has been preliminary work on data exchange from relations to RDF [7], followed up by a proposal of a tool [6] for mapping elements of the source schemas into the target RDFS ontology. [6,7] do not address temporal aspects of data exchange. To the best of our knowledge, temporal data exchange between relational schemas and ontologies has not been studied formally. Even in the relational-to-relational setting, [16] is the only publication that formally addresses temporal data exchange, for the case in which each source-to-target dependency uses at most one temporal variable. As time has its own semantics, adding it to data exchange is not a matter of simply adding temporal attributes to data-exchange rules. Thus, the formal constructs in [16] are rather involved. We use the results of [16] in the experimental validation of our proposed approach.

Paper outline: Sections 2–3 detail the two domain-independent algorithms that comprise the proposed approach, and illustrate their steps with a running example. Section 4 reports on our implementation and experimental results. Finally, Section 5 provides conclusions and directions of future work.

2 Temporal Enrichment of Ontologies and Data

The first problem that we consider is enrichment of time-agnostic RDFS ontologies and of the resulting materialized RDF data with temporal information from the relational sources. Our domain-independent rule-based Algorithm 1, which

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

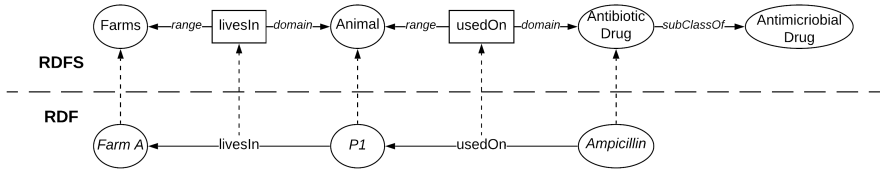

Fig. 1. The RDF (lower) level of this Figure shows two "subject-predicate-object" `(s,p,o)` triples, where *P1, FarmA,* and *Ampicillin* are names (URIs) of resources, and *livesIn* and *usedOn* are predicate names. At the RDFS (upper) level, the classes of the entities involved are related to each other through the domains and ranges of the predicates. Further, class *Antibiotic Drug* is shown to be a `subclassOf` *Antimicrobial Drug.* The two layers are connected via `type` statements, shown as vertical arrows.

addresses the problem, accepts three inputs. The first input comprises *relational data sources* with temporal information. We assume that temporal information in a relation, if present, is expressed via a single *marked* column whose values are time intervals. (Specifically, we assume *concrete* representation of *valid time* [9].) The second input is the target time-agnostic *RDFS domain ontology.* The final input is a set of *source-to-target tuple-generating dependencies (s-t tgds)* expressing the rules by which the source *domain* data can be materialized *(exchanged)* in the format conforming to the target ontology. (We assume that domain experts can design s-t tgds using tools such as that of, e.g., [6].) We assume that each rule is a *GLAV s-t tgd* [4,14] with up to one temporal variable, which (if present in the tgd) occurs once on the left-hand side *(LHS)* [16]. For s-t tgds to make sense in the relational-to-RDF/S scenario, we represent each RDF/S triple on the right-hand side *(RHS)* of the tgds, of the form "subject-predicate-object," or (s, p, o), as a relational atom of the form $p(s, o)$. See Eq. (1) for an illustration.

Algorithm 1: Temporal enrichment of ontologies, s-t tgds, and RDF data

Data: Relational data sources $\mathcal{D}$, RDFS ontology $\mathcal{O}$, and set $\mathcal{M}$ of s-t tgds.
Result: Temporally enriched $\mathcal{O}^T$, $\mathcal{M}^T$, and RDF target data set $\mathcal{F}^T$.
begin
  $\mathcal{M}^T \leftarrow \mathcal{M}$; $\mathcal{O}^T \leftarrow \mathcal{O}$; `// initialization`
  for *each atom* $p(s, o)$ *on the right-hand side of each* $M \in \mathcal{M}$ **do**
    if $p(s, o)$ *is in the temporal-enrichment scope of* M **then**
      $\mathcal{M}^T \leftarrow$ temporally enrich $p(s, o)$ in M; `// first stage`
      $\mathcal{O}^T \leftarrow$ temporally enrich the p-related part of $\mathcal{O}^T$; `// second stage`
  $\mathcal{F}^T \leftarrow$ materialize $\mathcal{D}$ into RDF via data exchange using $\mathcal{M}^T$; `// third stage`
  return $\mathcal{O}^T$, $\mathcal{M}^T$, and $\mathcal{F}^T$;

Algorithm 1 is based on straightforward domain-independent pattern-based rules, and can be viewed as consisting of three conceptually distinct stages. In the first stage, the algorithm adds "temporal-enrichment atom patterns" to the RHS of the input s-t tgds. For the patterns, we use the temporal structures

of [18], which, essentially, reify [17] RDF triples with their relevant temporal adornments, see the RDF level of Fig. 2 for an illustration. (We use the structural patterns of [18] to allow use of graph DBMSs without any special features for storing the RDF results of materializing temporal data from the sources.) In the second stage, the input time-agnostic ontology is augmented with RDFS-level specifications of the temporal-enrichment structures that have been added to the s-t tgds. Once both the input ontology and the input s-t tgds have been thus temporally enriched, in the third stage the resulting s-t tgds can be used to exchange the input (temporally-aware) data sources into the temporally-aware RDF format consistent with the (now) temporally-aware output ontology. (We assume that all of the materialized RDF data conform to the enriched ontology.)

Consider an example in the AMR domain. Suppose a data source has a relation *DrugUsage(Farm, Animal, AMR-Drug,* **Drug-Administration-Time),** which records the temporal history of AMR drug usage for animals in farms. Let the relation have a single tuple *('Farm A','P1','Ampicillin',[1/1/2019,1/5/2019]).* Note that the (marked) *Drug-Administration-Time* column of the relation records temporal-interval information, see the corresponding value in the tuple.

Suppose that AMR scientists are interested in obtaining answers to temporal queries posed using the ontology terminology. A fragment of the ontology information is shown at the RDFS[5] (top) level of Fig. 1. As the ontology is time agnostic, the best way to exchange data from the *DrugUsage* source to a target consistent with the ontology would be to use the s-t tgd

$$DrugUsage(f, a, d, \underline{\mathbf{t}}) \rightarrow livesIn(a, f) \wedge usedOn(d, a). \tag{1}$$

Here, $\underline{\mathbf{t}}$ is a temporal variable for the temporal attribute. Using this s-t tgd on the *DrugUsage* relation would result in the data shown at the RDF level of Fig. 1. Clearly, AMR scientists cannot get from these data a correct (nonempty) answer to the query "return the farms that used antibiotic drugs on their animals in the year 2019," as there is no temporal information in the stored data of Fig. 1.

This problem can be solved by applying Algorithm 1 to the above ontology, data source, and s-t tgd inputs. The algorithm will yield the enriched s-t tgd

$$\begin{aligned} DrugUsage(f, a, d, \underline{\mathbf{t}}) \rightarrow\ & livesIn(a, f) \wedge usedOn(d, a) \wedge tsubj(c_1, d) \\ & \wedge tpred(c_1, \texttt{usedOn}) \wedge tobj(c_1, a) \wedge temporal(c_1, c_2) \\ & \wedge interval(c_2, c_3) \wedge validFor(c_3, \underline{\mathbf{t}}). \end{aligned} \tag{2}$$

The RHS of the s-t tgd of Eq. (2) exhibits the temporal structure of [18] applied to the RDF triple represented by the atom $usedOn(d, a)$ on the RHS of Eq. (1). (We assume that domain experts can mark up RHS atoms of the input s-t tgds, to indicate the specific target RDF predicates that would receive the temporal adornment of [18]. This is the meaning of the "temporal-enrichment scope" criterion in Algorithm 1. The alternative is to allow for *all* the target RDF predicates used in the input s-t tgds to be so temporally adorned.)

[5] The notation used in Fig. 1–2 originates from [3].

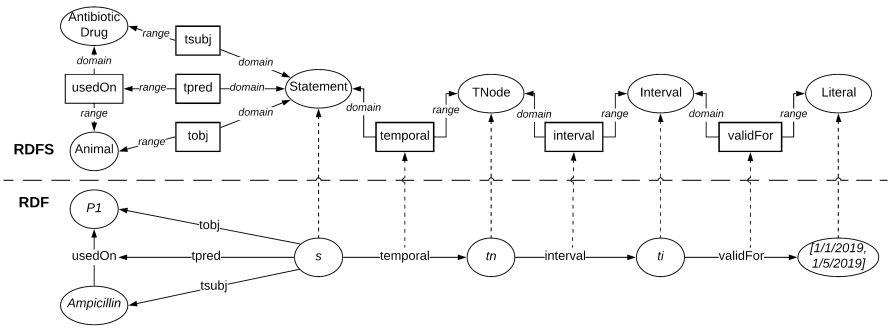

Fig. 2. This Figure shows an adornment of the *Ampicillin-[is]-usedOn-P1* RDF triple of Fig. 1 with a temporal structure of [18]. The RDFS layer shown indicates that the RDFS temporal-adornment metadata associated with that RDF triple of Fig. 1 contain a *Statement* class, linked to a node belonging to the *TNode* (temporal-node) class. The *TNode* is, in turn, characterized by an *interval*-value class. The RDF level shows instantiations of this RDFS structure — a (reified) statement s links the triple to temporal node *tn* characterized (via *ti*) as an interval, with (*validFor*) value [1/1/2019,1/5/2019].

In Eq. (2), c_1 (c_2, c_3, respectively) stands for $f_1(d, a, usedOn)$ (for $f_2(d, a, usedOn)$, $f_3(d, a, usedOn)$, respectively). When Algorithm 1 materializes target RDF data from the relational sources using such temporally enriched s-t tgds, functions such as f_1–f_3 produce unique new URI values for the temporal structures of [18] with the RDF triples being materialized. Please see Fig. 2 for an illustration; here, s, tn, and ti are the values produced by the three functions of Eq. (2) for the RDF triple with predicate *usedOn* shown in Fig. 1. Finally, the top half of Fig. 2 shows the time-enriched ontology information that results from applying Algorithm 1 to the inputs of this example.

3 Querying the Materialized Temporally Enriched Data

Suppose that Algorithm 1 has been applied to the given relational data sources $\mathcal{D}$, time-agnostic target ontology $\mathcal{O}$, and s-t tgds $\mathcal{M}$. As a result, we obtain an RDF/S data set $(\mathcal{F}^T, \mathcal{O}^T)$ that materializes source information, including temporal characterizations of the source data. See Fig. 2 for a partial illustration of $(\mathcal{F}^T, \mathcal{O}^T)$ for the example of Section 2. Now the RDF query language SPARQL [24,28] can be used to formulate, with respect to (w.r.t.) $(\mathcal{F}^T, \mathcal{O}^T)$, temporal queries such as Q^T_{am}: "Return farms that used antimicrobial drugs in the year 2019," see Fig. 3(c). This temporal query can be processed directly on the data set $(\mathcal{F}^T, \mathcal{O}^T)$ by a standard SPARQL processor, with a nonempty answer successfully returned on the data coming from the *DrugUsage* relation.

As illustrated in Fig. 3(c), direct temporal querying of temporal RDF/S data sets is already enabled by our approach of Section 2. At the same time, our additional objective in this paper is to allow domain analysts and experts to concentrate on the domain-ontology part of formulating such temporal queries, while keeping the temporal part of the queries as easy to write as possible. For

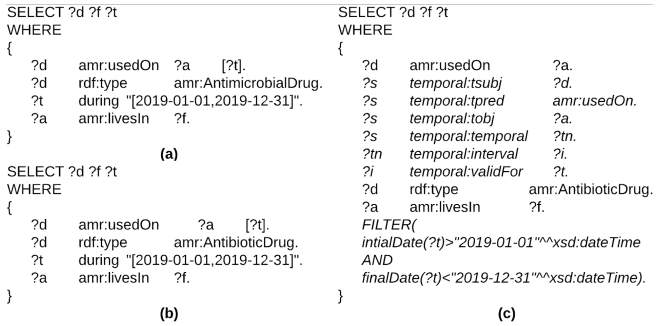

Fig. 3. Query Q^T_{am} asking for the farms that used antimicrobial drugs in the year 2019, as *(a)* the original *temporally-annotated* SPARQL version, *(b)* the result of the rewriting of that version by the 1st stage of Algorithm 2, and *(c)* the result of the expansion of version *(b)* by the 2nd stage of Algorithm 2. (In *(c)*, *initialDate* and *finalDate* are shorthand representations of combinations of standard SPARQL functions for extracting the start/end points from the time-interval values bound to the temporal variable *?t*.) Unlike *(a)–(b)*, version *(c)* is directly executable by standard SPARQL processors.

this purpose, we offer domain experts an opportunity to formulate their temporal queries via a *temporal user interface (temporal UI)* that we provide for SPARQL. In the UI, standard SPARQL constructs are supplemented with *temporal annotations* on RDF/S triple patterns in the queries, using the notation that we borrow from the query format of [32], as well as with constructs for temporal comparisons, such as `during`, `before`, and `after`, which are known as *Allen interval relations* [2]. See Fig. 3(a) for an illustration, in which $?t$ is a temporal annotation. The straightforward details of the temporal UI are omitted due to the space limit. In the remainder of the exposition, we will refer to temporal-UI versions of SPARQL queries as *temporally annotated SPARQL queries.*

Algorithm 2: Temporal querying of temporally enriched RDF/S data sets

Data: RDFS ontology $\mathcal{O}^T$, RDF data set $\mathcal{F}^T$, temporally annotated SPARQL query Q.
Result: Answer set $\mathcal{A}$ to a SPARQL reformulation of Q on $\mathcal{F}^T$.
begin
  $\mathcal{R} \leftarrow \{Q\}$; `// will reformulate` Q `into` $\mathcal{R}$ `that is executable on` $\mathcal{F}^T$
  for *each triple pattern* P *in* $\mathcal{R}$ **do**
    if *there is a hierarchy* H *in* $\mathcal{O}^T$ *that applies to* P **then**
      $\mathcal{R} \leftarrow$ rewrite P in $\mathcal{R}$ in all ways using H; `// 1st stage: rewriting`
  for *each temporal annotation* T *in* $\mathcal{R}$ **do**
    $\mathcal{R} \leftarrow$ expand T in $\mathcal{R}$ into triple patterns; `// 2nd stage: expansion`
  $\mathcal{A} \leftarrow \emptyset$; `// initializing set of answers to` $\mathcal{R}$ `on RDF data set` $\mathcal{F}^T$
  for *each SPARQL query* R *in* $\mathcal{R}$ **do**
    $\mathcal{A} \leftarrow$ use SPARQL processor to add to $\mathcal{A}$ the result of processing R on $\mathcal{F}^T$;
  return $\mathcal{A}$;

We now present a domain-independent approach for reformulating temporally annotated SPARQL queries into (standard) SPARQL queries that respect the structure of the temporal RDF information while preserving the semantics of the questions. Acting on top of a SPARQL processor, our Algorithm 2 ensures successful evaluation of temporally annotated SPARQL queries on the materialized temporally-enriched RDF/S data generated by Algorithm 1 (Section 2).

Algorithm 2 accepts as inputs RDF/S data sets $(\mathcal{F}^T, \mathcal{O}^T)$ and temporally annotated SPARQL queries Q expressed in terms of the domain-ontology part of $\mathcal{O}^T$. (As discussed above, the temporal notation in Q comes from the lightweight easy-to-use domain-independent formalism for time annotations and comparisons in our temporal UI.) The algorithm reformulates each given Q into a set $\mathcal{R}$ of standard SPARQL queries conforming to the input ontology $\mathcal{O}^T$, and then uses the SPARQL processor to obtain the answer to Q, by processing all the queries in $\mathcal{R}$ on the input RDF/S data set $(\mathcal{F}^T, \mathcal{O}^T)$.

The reformulation part of Algorithm 2 works in two stages, rewriting (1st stage) and expansion (2nd stage). In the 1st stage, the algorithm uses domain-independent pattern-based rules to repeatedly "unfold," in the queries being rewritten, `:subClassOf` and `:subPropertyOf` hierarchies w.r.t. the RDFS ontology $\mathcal{O}^T$, using the entailment rules of [12,17,25]. As a result, the input query Q is turned into a set $\mathcal{R}$ of temporally annotated SPARQL queries that would be directly executable on the data set $\mathcal{F}^T$ *but for* their temporal annotations. This process would transform the query of Fig. 3(a) into the query of Fig. 3(b).

The 2nd, expansion, stage of the query-reformulation process in Algorithm 2 uses domain-independent pattern-based rules to replace the temporal annotations in the queries $\mathcal{R}$ with standard RDF/S constructs. Specifically, all the temporal annotations of individual triple patterns in $\mathcal{R}$ are replaced with their structural counterparts of [18] (as in, e.g., Fig. 2), and all the Allen interval relations (e.g., `during`) are replaced with built-in comparisons on the endpoints of the time intervals involved. (This process would transform the query of Fig. 3(b) into the query of Fig. 3(c).) The resulting queries, which are SPARQL queries without any nonstandard annotations, are submitted by the algorithm to the SPARQL processor to obtain the answers to the input query.

4 Implementation and Experimental Results

4.1 Implementation and Experimental Setup

We have implemented Algorithms 1–2 on top of Java 1.8, using the Llunatic [15] rule interpreter for rewriting and expanding temporally annotated queries into standard executable SPARQL queries. The relational source data were stored using PostgreSQL 11, and storing and manipulating target RDF triples was done with RDF4J 3.0.1. All the experiments were conducted in the environment of Ubuntu 18.04 Bionic with Core i7 3.20GHz, 16GB RAM, and 2TB HDD.

For the experiments, we used data environments in two application domains. Each environment included a relational source schema, a time-agnostic target RDFS domain ontology and, for translating the schema into the ontology, a set

of GLAV s-t tgds each with at most one temporal variable, which, if present, would occur exactly once on the LHS. Each data environment also included relational source data generated with DataFiller [10] at multiple scale factors, to calibrate the volume of the resulting data sets, as well as temporal queries defined in terms of the domain ontologies, see Section 4.3. The first data environment captures a real-life AMR scenario obtained from our collaboration [19] with AMR researchers at NC State University. Each source AMR data set used in the experiments consisted of four tables, *Resistance, FarmInfo, CityInfo,* and *WeatherEvents,* with relative table-size ratios of *1:0.5:0.1:0.8*. The *Resistance* table would contain AMR-testing information coming from farms, for samples of specific bacteria w.r.t. different antimicrobial drugs, e.g., *Ampicillin*, and the date ranges for the tests. The three other tables would contain information about the farms, their locations, and relevant weather events. We also used the RDFS version of the time-agnostic ARO ontology for AMR, as well as s-t tgds developed in our collaboration with AMR researchers.

The second data environment used in the experiments was based on the TPC-BiH benchmark [13,20,21], which is used for evaluating the performance of temporal databases. The TPC-BiH schema is based on eight TPC-H [30] relations, which collectively reflect realistic information in a business-application scenario. (For the relative TPC-BiH table-size ratios, see [30].) TPC-BiH introduces temporal semantics for six of the eight TPC-H tables; in our experiments, we used the valid-time interval attributes in the TPC-BiH schema. We converted the TPC-BiH schema into a time-agnostic (cf. TPC-H) ontology and generated the associated s-t tgds using an approach similar to those of [8,31].

4.2 Experimental Methodology and Expected Outcomes

The experiments, whose results are reported in Section 4.3, were designed around specific properties of the outcomes of applying to the AMR and TPC-BiH environments the proposed approach (i.e., Algorithms 1–2) for temporal RDF/S enrichment and querying. We evaluated the following properties of the outcomes:

- degree of preservation in the target of the temporal information from the sources, see Fig. 4; and
- degree of correctness of the answers to temporal queries on the target, w.r.t. the "benchmark" answers, see discussion of *Experiments (I)-(II)* below.

We evaluated the latter property both for queries that required rewriting w.r.t. the `:subClassOf` and `:subPropertyOf` hierarchies in the given time-agnostic ontologies (1st stage of Algorithm 2), and for queries that did not require such rewriting, see Fig. 5. (In addition, we evaluated the efficiency of our implementation, see discussion of Fig. 6 in Section 4.3.)

Our methodology for evaluating the above properties was as follows. Observe that the proposed approach, embodied by Algorithms 1–2, would be provably sound if it applied to the relational-to-*relational* scenario. That is, if the target data were relational (instead of RDF/S), then, under our assumption (see Section 2) of all s-t tgds being GLAV with at most one temporal variable, correctness

of temporal data exchange and of temporal querying on the materialized target data would follow from the formal results of [16]. Thus, each experiment that we are reporting on was done twice for each fixed data-environment input:

1. *Experiment (I)* would be conducted in the relational-to-*relational* scenario using straightforward modifications of Algorithms 1–2. By [16], that experiment would have provably correct outputs, which we would then adopt as our *expected outputs* for the other experiment (i.e., *Experiment (II)*); and
2. *Experiment (II)* would be conducted in the relational-to-*RDF/S* scenario using the original Algorithms 1–2. We would then translate the RDF outputs into relations using straightforward transformations, and compare the results with the expected outputs of *Experiment (I)* for the same input.

For all the cases in which the results of *Experiments (I)* and *(II)* were identical for a given input, we would count all such cases as corroborations of the experimental validation of correctness of the proposed approach, i.e., of Algorithms 1–2.

4.3 Summary of Experimental Results

We now report on our experimental results. As a high-level summary, for each data environment used in the experiments, with each selected scale factor, and for each temporal query that was considered, the results of *Experiments (I)* and *(II),* as described in Section 4.2, were identical. We conclude that all these results experimentally validate the correctness of the proposed approach. (Formally proving correctness of Algorithms 1–2 is a direction of our current work.)

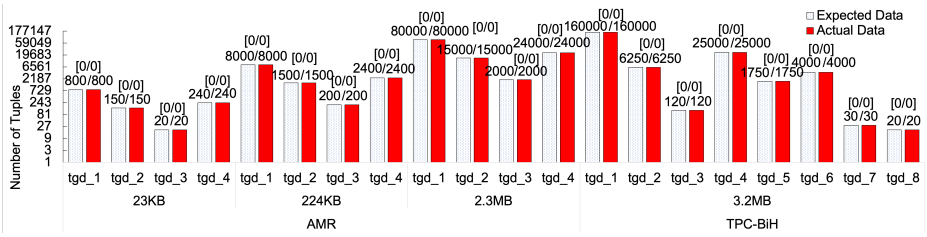

Fig. 4. Evaluating information loss in data exchange with temporal RDF/S enrichment, vs expected (relational-to-relational) outcomes. The X-axis shows the names of the s-t tgds and the source-data sizes for the environments tested; the (logarithmic) Y-axis shows the number of resulting data tuples. The $[A/B]$ notation on top of the bars shows the relative number of unmatched tuples between the two sets, see Section 4.3 for the details. Specifically, the meaning of [0/0] is that the two data sets are identical.

The data environments that were used to obtain the results shown in Fig. 4–6 are as follows: the AMR environment with the source data comprising 100, or 1,000, or 10,000 tuples, at the respective storage sizes of 23KB through 2.3MB, as well as the TPC-BiH environment with the source data comprising 10,000 tuples, at storage size of 3.2MB. (We are not reporting the results for additional values of the scale factors that were tested for the AMR and TPC-BiH environments, due to all such results being in line with the results reported in Fig. 4–6.)

We report first on our results, in the AMR and TPC-BiH data environments, for the degree of preservation in the target of the temporal information from the sources, as enabled by Algorithm 1. Fig. 4 shows the sizes of the target data, both relational *(Experiments (I))* and RDF/S (the corresponding *Experiments (II)),* that were obtained in the experiments. The $[A/B]$ values on top of the target data-size bars show the sizes of the outcomes of the set differences between the two data sets in both directions, with the value A showing the number of additional tuples obtained in the relational-to-*relational* setting, and with B showing the symmetric number for the relational-to-*RDF/S* setting.[6] For all the results, we got A=B=0; that is, in each experiment we obtained identical sets of tuples in the target temporal data. **We conclude** that the results *experimentally validate* the correctness of our temporal-enrichment Algorithm 1.

Next, we discuss our results, in the AMR and TPC-BiH data environments, for the degree of correctness of the answers to temporal queries on the RDF/S target *(Experiment (II)),* w.r.t. the "benchmark" relational answers that would be obtained via the respective *Experiment (I),* see Fig. 5. All the input queries in the experiments were temporally annotated SPARQL queries of the form illustrated in Fig. 3(a), which were then subjected to appropriate reformulations via Algorithm 2, with the outputs being SPARQL queries of the form illustrated in Fig. 3(c). We used the certain-answer semantics [4,14] in processing all the queries. Among the queries that we tested in these experiments, the meaning of, e.g., the AMR query $Q2$ in Fig. 5(a) is "return drugs that have been linked to antibiotic resistance by any bacteria serotypes, and the relevant date ranges." In the TPC-BiH data environment, we tested 11 named temporal queries [20].

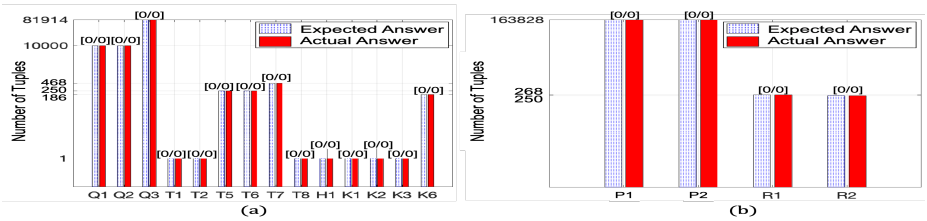

Fig. 5. Evaluating information loss in answers to temporal queries, vs expected (relational-to-relational) benchmark outcomes, for (a) queries not requiring rewriting w.r.t. RDFS hierarchies, and for (b) queries requiring such rewriting. The X-axes show the names of the AMR and TPC-BiH queries tested. The (logarithmic) Y-axes show query-answer sizes in tuples. The $[A/B]$ notation on top of the bars shows the relative number of unmatched tuples between the two sets, see Section 4.3 for the details. Specifically, the meaning of the [0/0] notation is that the two data sets are identical.

None of the queries mentioned in Fig. 5(a) required rewriting w.r.t. RDFS hierarchies using the 1st stage of Algorithm 2. As a result, composing their semantically equivalent SQL counterparts for the respective *Experiments (I)* was

[6] Recall that part of each *Experiment (II)* was to convert the RDF outputs into relations, to enable the comparisons with the respective outputs of *Experiment (I).*

straightforward. The experiments whose results are reported in Fig. 5(b) were set up differently, as their temporally annotated input queries did require rewriting w.r.t. the RDFS `:subClassOf` and `:subPropertyOf` hierarchies. Among the queries tested in these experiments, the meaning of, e.g., the query $R2$ in the TPC-BiH environment (Fig. 5(b)) is "return the nations for all the entities that have associated temporal information," where "entities" is a superclass of both "customers" and "suppliers." For the *Experiment (I)* counterparts of the temporally annotated SPARQL queries mentioned in Fig. 5(b), we manually constructed SQL queries that are semantically equivalent to the result of rewriting the input queries for *Experiment (II)* using the 1st stage of Algorithm 2.[7]

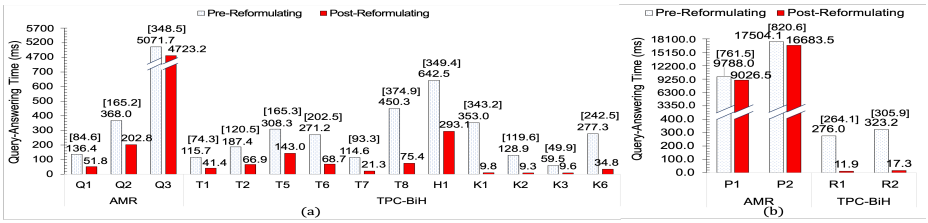

Fig. 6. Measuring the time overhead of reformulating temporally annotated queries into executable SPARQL as part of the overall query processing, for (a) queries not requiring rewriting w.r.t. RDFS hierarchies, and for (b) queries requiring such rewriting. The X-axes show the names of the AMR and TPC-BiH queries tested. The (logarithmic) Y-axes show the response times in *ms*. Each bar height is the average of 10 runtimes. The values in square brackets show the difference, for each query, between the processing time with the reformulation overhead included (left bar) and excluded (right bar).

To summarize the results shown in Fig. 5, we obtained that the sets of query answers were identical between *Experiments (I)* and *(II),* on all the inputs and for all the individual queries tested. (The query answers were relations for both *(I)* and *(II),* because none of the queries used graph-constructing features of SPARQL.) Specifically, the set-difference meaning of the $[A/B]$ notation on top of the bars in Fig. 5 is the same as in Fig. 4, with $A{=}B{=}0$ in all the experiments that we conducted. **We conclude** that our results for the degree of correctness of the answers to temporal queries on the RDF/S target *experimentally validate* the correctness of the proposed query-reformulation Algorithm 2.

Finally, Fig. 6 reports on the results for the runtime overhead of our implementation of the query-reformulation part of Algorithm 2, as part of the overall response times for the queries tested. The heights of the bars shown in Fig. 6 show the overall response times for the temporally annotated queries, with the Algorithm 2 reformulation overhead included in the left bar in each pairing, and excluded in the corresponding right bar. The response times were measured both for queries that did not require rewriting w.r.t. RDFS hierarchies (1st stage of Algorithm 2), see Fig. 6(a), and for queries requiring such rewriting, see Fig.

[7] Recall that, in contrast with the processing of SPARQL queries in RDF stores, automatic rewriting of queries using hierarchies is not possible in relational DBMS.

6(b). Not surprisingly, in all the cases tested, the overhead of Algorithm 2 depended only on the size of the input query, rather than on the size of the stored data processed by the query, or on the size of the query answer. As a result, even for queries whose runtimes were over 16 *sec after* the reformulation part of Algorithm 2, the overhead of applying Algorithm 2 was under 821 *ms*. This value is below the query-response user-tolerance time threshold for interactive systems [23], which means that it is considered by experts to be acceptable for users querying the systems. Moreover, in additional experiments, in which we used hardcoded rules, rather than the rule interpreter [15], the values of the Algorithm 2 reformulation overhead dropped to tens of *ms* vs. the time-difference values shown in Fig. 6. **We conclude** that the *runtime overhead of using Algorithm 2 in the reformulation of temporally annotated SPARQL queries is sufficiently small to be tolerated by users,* even when it uses generic rule interpreters, and can be significantly reduced further if needed in mission-critical systems.

5 Conclusions and Future Work

In this paper we considered the scenario in which domain analysts and scientists are interested in obtaining answers to *temporal* queries formulated in terms of the given *time-agnostic* RDFS domain ontology, in presence of temporal information in relational data sources and of source-to-target (s-t) rules for mapping domain information between the sources and the target ontology. We presented our declarative domain-independent algorithmic approach to addressing the temporal-enrichment and query-answering problems in this scenario. In our report on the approach, we described the algorithms and their implementation, and presented our experimental results for two application domains.

Currently, we are working on formal proofs of correctness of our proposed approach. Other directions of future formal and practical work on these topics include incorporation into the framework of richer ontology formalisms such as OWL [1], as well as of data-exchange dependencies that are more expressive in their temporal aspect than those of [16]. Another promising direction is to develop user interfaces that would make it easier for domain scientists that are not computer experts to query their temporal data in terms of domain ontologies.

References

1. Allemang, D., Hendler, J.: Semantic Web for the Working Ontologist: Effective Modeling in RDFS and OWL. Morgan Kaufmann, 2nd edn. (2011)
2. Allen, J.F.: Maintaining knowledge about temporal intervals. CACM **26** (1983)
3. Antoniou, G., Groth, P.T., van Harmelen, F., Hoekstra, R.: A Semantic Web Primer, 3rd Edition. MIT Press (2012)
4. Arenas, M., Barceló, P., Libkin, L., Murlak, F.: Foundations of Data Exchange. Cambridge University Press (2014)
5. Artale, A., Kontchakov, R., Kovtunova, A., Ryzhikov, V., Wolter, F., Zakharyaschev, M.: Ontology-mediated query answering over temporal data: A survey. In: TIME 2017, October 16-18, 2017, Mons, Belgium. pp. 1:1–1:37 (2017)

6. Boneva, I., Dusart, J., Fernández-Álvarez, D., Gayo, J.E.L.: Shape designer for ShEx and SHACL constraints. In: Proc. ISWC Satellite Tracks. pp. 269–272 (2019)
7. Boneva, I., Lozano, J., Staworko, S.: Relational to RDF data exchange in presence of a Shape Expression Schema. arXiv preprint arXiv:1804.11052 (2018)
8. Cheng, Y., Ding, P., et al.: Which category is better: Benchmarking relational and graph database management systems. D. Sci. Engnr. **4**, 309–322 (2019)
9. Chomicki, J., Toman, D.: Temporal databases. In: Fisher, M., Gabbay, D.M., Vila, L. (eds.) Handbook of Temporal Reasoning in AI, pp. 429–467. Elsevier (2005)
10. Coelho, F.: DataFiller − generate random data from database schema. `https://www.cri.ensmp.fr/people/coelho/datafiller.html` (2014)
11. Combating Antimicrobial Resistance: A One Health Approach to a Global Threat: Proc. National Academies of Sciences Workshop. National Academies Press (2017)
12. Curé, O., Blin, G. (eds.): RDF Database Systems. Morgan Kaufmann (2015)
13. Dignös, A., Glavic, B., Niu, X., Böhlen, M., Gamper, J.: Snapshot semantics for temporal multiset relations. Proc. VLDB Endow. **12**(6), 639–652 (2019)
14. Fagin, R., Kolaitis, P.G., Miller, R.J., Popa, L.: Data exchange: semantics and query answering. Theor. Comput. Sci. **336**(1), 89–124 (2005)
15. Geerts, F., Mecca, G., Papotti, P., Santoro, D.: Cleaning data with Llunatic. VLDBJ (2019), `https://doi.org/10.1007/s00778-019-00586-5`
16. Golshanara, L., Chomicki, J.: Temporal data exchange. Inf. Systems **87** (2020)
17. Gutiérrez, C., Hurtado, C.A., Mendelzon, A.O., Pérez, J.: Foundations of Semantic Web databases. Journal of Computer and System Sciences **77**(3), 520–541 (2011)
18. Gutiérrez, C., Hurtado, C.A., Vaisman, A.A.: Introducing time into RDF. IEEE Trans. Knowl. Data Eng. **19**(2), 207–218 (2007)
19. Hou, P.Y., Ao, J., Rindos, A., Keelara, S., Fedorka-Cray, P., Chirkova, R.: Collaborative workflow for analyzing large-scale data for antimicrobial resistance: An experience report. In: Proc. IEEE BigData (December 2019)
20. Kaufmann, M., Fischer, P.M., May, N., Tonder, A., Kossmann, D.: TPC-BiH: A benchmark for bitemporal databases. In: Nambiar, R., Poess, M. (eds.) Performance Characterization and Benchmarking (2014)
21. Lu, W., Zhao, Z., Wang, X., et al.: A lightweight and efficient temporal database management system in TDSQL. Proc. VLDB Endow. **12**, 2035–2046 (2019)
22. Michel, F., Montagnat, J., Zucker, C.F.: A survey of RDB to RDF translation approaches and tools (2014), Rapport de Recherche ISRN I3S/RR 2013-04-FR
23. Nielsen, J.: Usability Engineering. Morgan Kaufmann (1993)
24. Özsu, M.T.: A survey of RDF data management systems. Frontiers of Computer Science **10**(3), 418–432 (2016)
25. Polleres, A., Hogan, A., Delbru, R., Umbrich, J.: RDFS and OWL reasoning for linked data. In: Proc. Semantic Technologies for Intelligent Data Access (2013)
26. RDF Semantics: W3C recommendation 10 February 2004 (2004), Hayes, Patrick (Ed.), `https://www.w3.org/TR/2004/REC-rdf-mt-20040210/`
27. RDF Vocabulary Description Language: RDF Schema (2014), Brickley, D. and Guha, R.V. (Eds), `https://www.w3.org/TR/rdf-schema/`
28. SPARQL query language for RDF. W3C 15/01/2008, `https://www.w3.org/TR/rdf-sparql-query/`
29. Tappolet, J., Bernstein, A.: Applied temporal RDF: efficient temporal querying of RDF data with SPARQL. In: Proc. ESWC. pp. 308–322 (2009)
30. TPC benchmark H rev. 2.18.0. `http://www.tpc.org/tpc_documents_current_versions/pdf/tpc-h_v2.18.0.pdf` (2018)
31. Working with the TPCH data. `https://docs.cambridgesemantics.com/anzograph/userdoc/ghib.htm#top-supplier/` (2020)

32. Zimmermann, A., Lopes, N., Polleres, A., Straccia, U.: A general framework for representing, reasoning and querying with annotated Semantic Web data. Journal of Web Semantics **11**, 72–95 (2012)