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 the 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 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 the data and the ontology with temporal information from the sources. The second algorithm accepts as inputs temporal queries expressed in terms of the domain ontology, using SPARQL supplemented with time annotations. The algorithm translates the queries into the standard SPARQL form that respects the structure of the temporal RDF information while preserving the question semantics. We present the algorithms, report on their implementation and experimental results for two application domains, and discuss future.

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* [1]), using expert-developed *source-to-target (s-t) rules.* In many applications, the target data formats also have to be aligned with the standard domain vocabularies called *ontologies.* 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 [2] enriched with additional *RDFS* specifications [3], – and are queried using SPARQL [4], 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)*, the source data may

J. Darmont et al. (Eds.): ADBIS 2020, CCIS 1259, pp. 129–139, 2020.
https://doi.org/10.1007/978-3-030-54623-6_12

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*). Existing relational-to-RDF/S data-exchange solutions 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. Custom solutions developed on a case-by-case basis [5] would delegate to data analysts or domain scientists the nontrivial task of temporally enhancing the originally time-agnostic domain ontologies, such as ARO. In addition, to correctly formulate temporal queries, domain analysts 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 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 analysts (users) are familiar with formulating SPARQL queries using the given RDFS ontology, and that they provide the s-t rules that map the domain information in the source schemas into the time-agnostic target ontology, using tools such as that of [11]. In this scenario, we propose a declarative domain-independent 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 approach 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 the target data and ontology with temporal information from the sources. The second algorithm accepts as inputs temporal queries expressed in terms of the ontology, using SPARQL supplemented with a lightweight formalism for time annotations and comparisons. The algorithm translates queries into the standard SPARQL form that respects the structure of the temporal RDF information while preserving the question semantics, thus ensuring successful evaluation of the queries on the materialized temporally-enriched RDF data. In this paper we present the algorithms (Sect. 2–3), report on their implementation and experimental results for two application domains (Sect. 4), and discuss future work (Sect. 5). Please see the full version of the paper [18] for the details.

Related Work. RDFS [3] is a language used in practice for describing ontologies. Existing works have focused on representing and reasoning with temporal RDF data [6], querying such data [7], and inferring temporal properties in temporal RDFS ontologies [8]. 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.

Relational data exchange has been studied extensively [1]. For relational-to-RDF data exchange, see [10,11]. To the best of our knowledge, temporal data exchange between relational schemas and ontologies has not been studied formally. The only formal work on temporal relational data exchange is in [12]. We use the results of [12] in the experimental validation of our approach.

2 Temporal Enrichment of Ontologies and Data

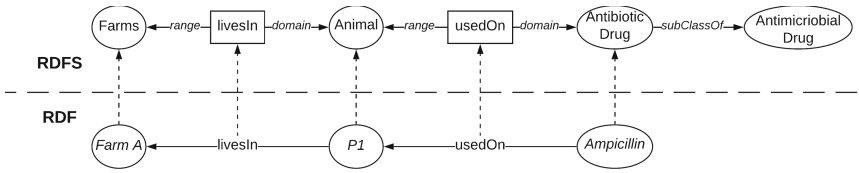

Fig. 1. The RDF (lower) level of this Figure shows two "subject-predicate-object" `(s,p,o)` triples, with names (URIs) of resources (e.g., *Farm A*), and predicate names (e.g., *livesIn*). At the RDFS level, the classes of the entities are related to each other through the domains and ranges of the predicates. Class *Antibiotic Drug* is shown to be a `subclassOf` *Antimicrobial Drug.* The two layers are connected via `type` statements.

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 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* [13].) 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 each rule is a *GLAV s-t tgd* [1] with up to one temporal variable, which (if present) occurs once on the left-hand side *(LHS)* [12]. 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 $p(s, o)$.

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 [6], which, essentially, reify [14] RDF triples with their relevant temporal adornments, see the RDF level of Fig. 2 for an illustration. (We use the structural patterns of [6] to allow use of graph DBMSs without any special features for storing the RDF results of materializing temporal data from the sources.) In

Algorithm 1: Temporally enriching 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 target RDF 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 $\mathcal{F}^T$;

the second stage, the input time-agnostic ontology is augmented with RDFS-level specifications of the temporal-enrichment structures that enriched the s-t tgds. 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*) for recording 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]).* Suppose that analysts would like to obtain answers to temporal queries posed using the ontology terminology shown at the RDFS (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 Eq. (2) exhibits the temporal structure of [6] applied to the RDF triple represented by the atom $usedOn(d, a)$. c_1 (c_2, c_3, resp.) stands for unique

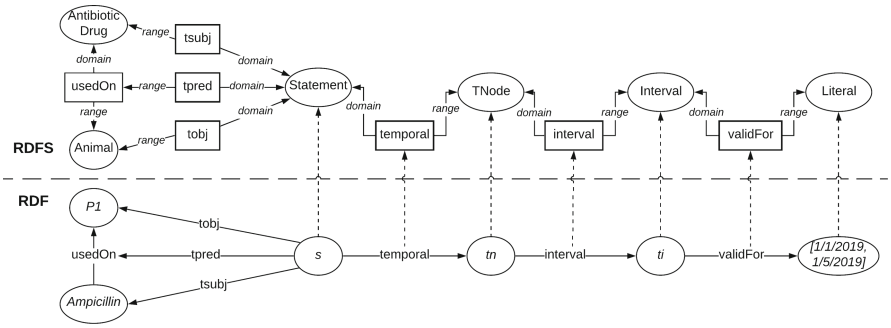

Fig. 2. An adornment of the *Ampicillin-[is]-usedOn-P1* RDF triple of Fig. 1 with a temporal structure of [6]. The RDFS layer shows the metadata of [6], including a *Statement* class and a *TNode* (temporal-node) class. The *TNode* is characterized by an *interval*-value class. The RDF level shows instantiations of these RDFS metadata.

new URIs generated for the temporal structure of [6] with the RDF triples being materialized; e.g., *s*, *tn*, and *ti* are generated for the triple *Ampicillin-[is]-usedOn-P1* in Fig. 2. The top half of Fig. 2 shows the time-enriched ontology information that results from applying Algorithm 1 to the inputs of the example.

3 Querying the Materialized Temporally Enriched Data

```
SELECT ?d ?f ?t
WHERE
{
    ?d      amr:usedOn    ?a      [?t].
    ?d      rdf:type      amr:AntimicrobialDrug.
    ?t      during "[2019-01-01,2019-12-31]".
    ?a      amr:livesIn   ?f.
}
```

(a)

```
SELECT ?d ?f ?t
WHERE
{
    ?d      amr:usedOn    ?a      [?t].
    ?d      rdf:type      amr:AntibioticDrug.
    ?t      during "[2019-01-01,2019-12-31]".
    ?a      amr:livesIn   ?f.
}
```

(b)

```
SELECT ?d ?f ?t
WHERE
{
    ?d      amr:usedOn          ?a.
    ?s      temporal:tsubj      ?d.
    ?s      temporal:tpred      amr:usedOn.
    ?s      temporal:tobj       ?a.
    ?s      temporal:temporal   ?tn.
    ?tn     temporal:interval   ?i.
    ?i      temporal:validFor   ?t.
    ?d      rdf:type            amr:AntibioticDrug.
    ?a      amr:livesIn         ?f.
    FILTER(
    intialDate(?t)>"2019-01-01"^^xsd:dateTime
    AND
    finalDate(?t)<"2019-12-31"^^xsd:dateTime).
}
```

(c)

Fig. 3. Query Q^T_{am} asking for the farms that used antimicrobial drugs in 2019, as *(a)* the original *temporally-annotated* SPARQL version, *(b)* the result of its rewriting 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 for SPARQL functions for extracting the start/end points from the time-interval values bound to *?t*.) Unlike *(a)–(b)*, version *(c)* is directly executable by standard SPARQL processors.

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. Now the RDF query language

SPARQL [4] 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 Sect. 2. At the same time, our additional objective is to allow domain analysts 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 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 [8], as well as with constructs for temporal comparisons, such as `during`, which are known as *Allen's interval relations* [9]. See Fig. 3(a) for an illustration, with temporal annotation ?t. We will be referring to temporal-UI versions of SPARQL queries as *temporally annotated SPARQL queries.*

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

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 (Sect. 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 $\mathcal{O}^T$. The algorithm reformulates each given Q into a set $\mathcal{R}$ of SPARQL queries conforming to the 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 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 entailment rules, see, e.g., [14]. 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 [6] (as in, e.g., Fig. 2), and all the Allen's 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 SPARQL queries are submitted by the algorithm to the SPARQL processor to obtain the answers to the input query.

4 Implementation and Experimental Results

We have implemented Algorithms 1–2 on top of Java 1.8, PostgreSQL 11, and RDF4J 3.0.1, using the Llunatic [15] rule interpreter for reformulating temporally annotated queries into standard executable SPARQL queries. For the experiments, we used data environments in two application domains, AMR and TPC-BiH [17]. 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 [16] at multiple scale factors, as well as temporal queries defined in terms of the domain ontologies.

The experiments were designed around two properties of the outcomes of applying to the AMR and TPC-BiH environments the approach of Algorithms 1–2 for temporal RDF/S enrichment and querying: (1) degree of preservation in the target of the temporal information from the sources, see Fig. 4; and (2) degree of correctness of the answers to temporal queries on the target, w.r.t. the answers obtained in the baseline relational-to-relational approach supported by the formal results of [12], see Fig. 5. We evaluated the latter property both for queries that required rewriting w.r.t. `:subClassOf` and `:subPropertyOf` hierarchies in the given ontologies (1st stage of Algorithm 2), and for queries that did not require such rewriting. (See [18] for the details of our methodology.) We also evaluated the efficiency of our implementation, see Fig. 6.

As a high-level summary of our experimental results, for each data environment used in the experiments, with each selected scale factor, and for each temporal query that was considered, the experimental results were identical between our relational-to-RDFS setting and the baseline relational-to-relational setting. (The formal correctness of the outcomes in the latter setting is supported by the results of [12].) We conclude that all these results experimentally validate the correctness of the proposed approach.

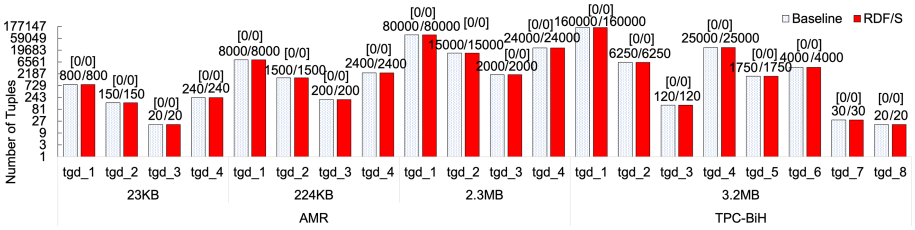

Fig. 4. Evaluating information loss in data exchange with temporal RDF/S enrichment vs. baseline outcomes. The X-axis shows the names of the s-t tgds and the source-data sizes for the environments tested; the (logarithmic) Y-axis shows the number of resulting data tuples. The $[A/B]$ notation on top of the target data-size bars shows the relative number of unmatched tuples between the two sets.

Figure 4 reports 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. For all the results, we got $A = B = 0$; that is, in each experiment we obtained the same sets of tuples in the target temporal data as in the baseline case. We conclude that the results experimentally validate the correctness of our temporal-enrichment Algorithm 1.

Figure 5 reports 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 (Algorithm 2) w.r.t. the relational answers that would be obtained in the baseline approach. All the input queries were temporally annotated SPARQL queries of the form illustrated in Fig. 3(a), which were then reformulated into standard SPARQL queries via Algorithm 2, as illustrated in Fig. 3(c). We used the certain-answer semantics [1] in processing all the queries. Given that $A = B = 0$ in all cases, 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.

Figure 6 reports 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 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. 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

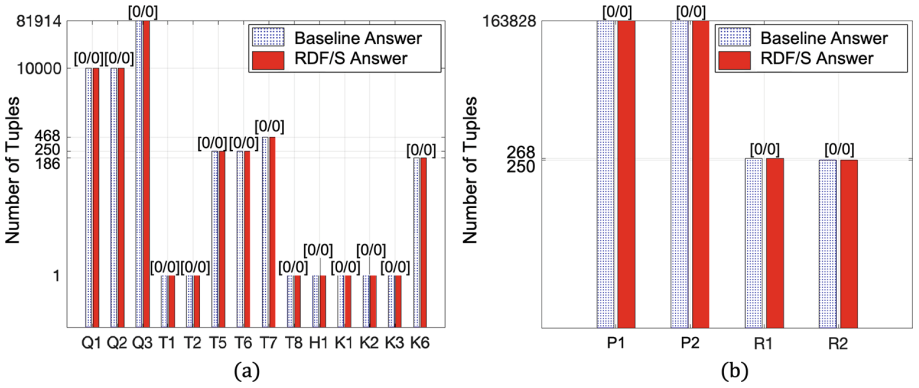

Fig. 5. Evaluating information loss in answers to temporal queries vs expected baseline outcomes. The X-axes show the names of the AMR and TPC-BiH queries tested. The (logarithmic) Y-axes show query-answer sizes in tuples. The $[A/B]$ notation on top of the bars shows the relative number of unmatched tuples between the two sets.

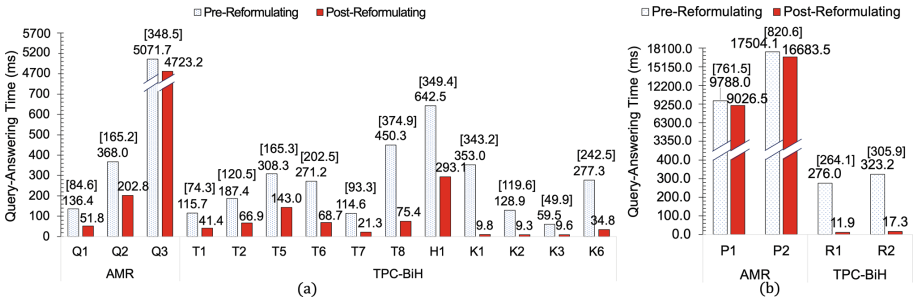

Fig. 6. Measuring the time overhead of reformulating temporally annotated queries into executable SPARQL. The X-axes show the names of the queries tested. The (logarithmic) Y-axes show the 10-runtime averaged overall response times in *ms*. The values in square brackets show the difference, for each query, between the processing time with the reformulation overhead included (left bar) and excluded (right bar).

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 user-tolerance time threshold for interactive systems [19]. We conclude that the runtime overhead of Algorithm 2 in the reformulation of temporally annotated SPARQL queries is sufficiently small to be tolerated by users.

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 the 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.

Providing formal proofs of correctness of our proposed approach is an immediate direction of future work. Other directions of future formal and practical work on the topics discussed in this paper include incorporation into the framework of richer ontology formalisms such as OWL, as well as of data-exchange dependencies that are more expressive in their temporal aspect than those of [12]. Another promising direction of research lies in designing and developing 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. Arenas, M., Barceló, P., Libkin, L., Murlak, F.: Foundations of Data Exchange. Cambridge University Press, Cambridge (2014)
2. Hayes, P. (ed.) RDF Semantics: W3C Recommendation (2004). https://www.w3.org/TR/2004/REC-rdf-mt-20040210/
3. Brickley, D., Guha, R.V. (eds.) RDF Vocabulary Description Language: RDF Schema (2014). https://www.w3.org/TR/rdf-schema/
4. SPARQL query language for RDF. https://www.w3.org/TR/rdf-sparql-query/
5. Michel, F., Montagnat, J., Zucker, C.F.: A survey of RDB to RDF translation approaches and tools, Rapport de Recherche ISRN I3S/RR 2013–04-FR (2014)
6. Gutiérrez, C., Hurtado, C.A., Vaisman, A.A.: Introducing time into RDF. IEEE Trans. Knowl. Data Eng. **19**(2), 207–218 (2007)
7. Tappolet, J., Bernstein, A.: Applied temporal RDF: efficient temporal querying of RDF data with SPARQL. In: Proceedings of the ESWC, pp. 308–322 (2009)
8. Zimmermann, A., Lopes, N., Polleres, A., Straccia, U.: A general framework for representing, reasoning and querying with annotated Semantic Web data. J. Web Semant. **11**, 72–95 (2012)
9. Allen, J.F.: Maintaining knowledge about temporal intervals. CACM **26**, 832–843 (1983)
10. Boneva, I., Lozano, J., Staworko, S.: Relational to RDF data exchange in presence of a Shape Expression Schema. arXiv preprint arXiv:1804.11052 (2018)
11. Boneva, I., Dusart, J., Fernández-Álvarez, D., Gayo, J.E.L.: Shape designer for ShEx and SHACL constraints. In: Proceedings ISWC Satellite Tracks, pp. 269–272 (2019)
12. Golshanara, L., Chomicki, J.: Temporal data exchange. Inf. Syst. **87**, 101414 (2020)
13. Snodgrass, R.T.: Temporal databases. In: Frank, A.U., Campari, I., Formentini, U. (eds.) GIS 1992. LNCS, vol. 639, pp. 22–64. Springer, Heidelberg (1992). https://doi.org/10.1016/S1574-6526(05)80016-1
14. Gutiérrez, C., Hurtado, C.A., Mendelzon, A.O., Pérez, J.: Foundations of Semantic Web databases. J. Comput. Syst. Sci. **77**(3), 520–541 (2011)

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. Coelho, F.: DataFiller - generate random data from database schema (2014). https://www.cri.ensmp.fr/people/coelho/datafiller.html
17. Kaufmann, M., Fischer, P.M., May, N., Tonder, A., Kossmann, D.: TPC-BiH: a benchmark for bitemporal databases. In: Nambiar, R., Poess, M. (eds.) TPCTC 2013. LNCS, vol. 8391, pp. 16–31. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-04936-6_2
18. Ao, J., et al.: Temporal Enrichment and Querying of Ontology-Compliant Data (Technical report TR-2020-3). https://www.csc.ncsu.edu/research/tech/reports.php
19. Nielsen, J.: Usability Engineering. Morgan Kaufmann, Burlington (1993)