Theory and Practice of Relational-to-RDF Temporal Data Exchange and Query Answering

JING AO, North Carolina State University
ZEHUI CHENG, University of California Santa Cruz
RADA CHIRKOVA, North Carolina State University
PHOKION G. KOLAITIS, IBM Research - Almaden

We consider the problem of answering temporal queries on RDF stores, in presence of atemporal 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 practice-oriented 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 a lightweight temporal extension of SPARQL, and ensures successful evaluation of the queries on the materialized temporally-enriched RDF data. To study the quality of the information generated by the algorithms, we develop a general framework that formalizes the relational-to-RDF temporal data-exchange problem. The framework includes a chase formalism and a formal solution for the problem of answering temporal queries in the context of relational-to-RDF temporal data exchange. In this article, we present the algorithms and the formal framework that proves correctness of the information output by the algorithms, and also report on the algorithm implementation and experimental results for two application domains.

CCS Concepts: • **Information systems → Data exchange**; **Resource Description Framework (RDF)**; **Query languages**; **Spatial-temporal systems**; • **Theory of computation → Database query processing and optimization (theory);**

Additional Key Words and Phrases: Data-intensive sciences and databases, theory of temporal databases, information quality in temporal data exchange, RDF / RDFS / SPARQL

ACM Reference format:
Jing Ao, Zehui Cheng, Rada Chirkova, and Phokion G. Kolaitis. 2023. Theory and Practice of Relational-to-RDF Temporal Data Exchange and Query Answering. *ACM J. Data Inform. Quality* 15, 2, Article 15 (June 2023), 27 pages.
https://doi.org/10.1145/3591359

All authors contributed equally to this research.
Authors' addresses: J. Ao and R. Chirkova, North Carolina State University, Raleigh, North Carolina, 27695; emails: {jao, rychirko}@ncsu.edu; Z. Cheng, University of California Santa Cruz, Santa Cruz, California, 95064; email: zecheng@ucsc.edu; P. G. Kolaitis, IBM Research - Almaden, San Jose, California, 95120; email: kolaitis@ucsc.edu.

1936-1955/2023/06-ART15 $15.00
https://doi.org/10.1145/3591359

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 into a unified *target* format *(exchanged* [5, 18]), using ***source-to-target (s-t)*** *rules* developed by domain experts for the purpose. Ensuring high quality of the information generated by the data-exchange process is paramount in many real-life applications. In many cases, 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. In this article, we consider the problem of ensuring high quality of the information generated by data exchange in the context of 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 [34] enriched with additional *RDFS* specifications [35], – and are queried using SPARQL [33, 36], while the source data are relational. The s-t rules that can be used in data exchange in this scenario should follow standard specifications, see, e.g., [5, 18], and can be composed using, e.g., W3C recommendations [40, 41] issued for relational-to-RDF data translation.

In applications conforming to the relational-to-RDF/S data-exchange scenario, e.g., in studies of **antimicrobial resistance *(AMR)*** [15], 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**).[1]) At the same time, solutions that are available for data exchange in the relational-to-RDF/S scenario, see, e.g., [30], 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 atemporal domain ontologies, such as ARO. As an additional burden on the users, to be able to formulate correctly their temporal queries, analysts or domain experts would need to be aware of how the temporal information is modeled and represented in the resulting systems.

Contributions: In this article, in the context of relational-to-RDF/S data exchange, we consider the scenario in which analysts and domain scientists are interested in obtaining correct answers to *temporal* queries formulated in terms of the given *atemporal* 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 target ontology that is atemporal from the perspective of the users' needs. (That is, the chosen ontology does not provide temporal constructs for the structures that the users are interested in querying). This situation appears to be common in practice, with realistic temporal-querying scenarios arising in the context of RDF ontologies that can be found online,

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

including the ARO ontology, the popular[2] DBLP Ontology,[3] the Food Ontology,[4] and the Media Ontology.[5]

In this scenario, we propose a formally justified approach that

— Enables users to formulate SPARQL-based temporal queries, and
— Returns to them correct 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 and data exchange.

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 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. (The SPARQL fragment used in this article covers conjunctions of triple patterns and variable projections [36]; this scope corresponds to that of conjunctive queries in the relational realm. Extending the scope of the SPARQL fragment used with the approach to, e.g., include OPTIONAL patterns is a direction of future work.) To prove correctness of the algorithms, we undertake a formal study of the relational-to-RDF temporal data-exchange problem. The resulting framework, which we present in this article, includes a chase formalism and a formal solution to the problem of correctly answering temporal queries in the context of relational-to-RDF temporal data exchange. In this article, we present the practice-oriented algorithms and the formal framework that ensures their correctness, and also report on the algorithm implementation and experimental results for two application domains.[6]

Related Work: RDFS [35] is a language used in practice for describing ontologies, see [1] and the Protege Ontology Library[7] 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 [24] defined temporal RDF and studied properties of inference in temporal graphs. Another approach, which focuses on querying, is presented in [37]. For the RDFS layer, research has been done on inference of temporal properties in temporal ontologies, see, e.g., [42]. At the same time, the temporal aspect is usually not included in the practical development of domain ontologies; the approach that we present in this article 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 [5, 13, 18]. Recently, considerable formal work has been done on temporal **ontology-mediated query access** (**OMQA**), see [6]. OMQA differs in its objectives from data exchange, which is our focus in this article, as the latter considers mainly materialization of exchanged data, while the former concentrates on certain

[2] https://blog.dblp.org/2022/03/02/dblp-in-rdf/.
[3] https://dblp.org/rdf/schema.ttl.
[4] https://raw.githubusercontent.com/FoodOntology/foodon/master/foodon.owl.
[5] https://www.w3.org/ns/ma-ont.ttl.
[6] The results of Sections 4, 5.4, and 6 of this article are based on [4]; the results of Sections 2, 3, and of the remainder of 5 have not been previously published.
[7] https://protegewiki.stanford.edu/wiki/Protege_Ontology_Library.

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 [8, 9], followed up by a proposal of a tool [7] for mapping elements of the source schemas into the target RDFS ontology. [7–9] 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, [13, 21] are the only works that formally address 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 [13, 21] are rather involved.

Article outline: After presenting the preliminaries in Section 2, we introduce in Section 3 the formal foundations of relational-to-RDF temporal data exchange. Section 4 presents an algorithm that applies these foundations in practical scenarios in which the available RDF ontology is atemporal. Section 5 reports on our formal results and practical algorithm for querying instances of temporal RDF graph schemas. Finally, Section 6 reports on our implementation and experimental results.

2 PRELIMINARIES

In this section we review the background definitions.

Temporal Values. Let $\mathbb{N}$ be the set of all natural numbers. In the *abstract* model of time, natural numbers represent *time points*. 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*. The domain of time intervals is defined as a set $T_I = \{[s, e) : s < e \text{ and } s, e \in \mathbb{N}\}$. Let $[s, e)$ and $[s', e')$ be two time intervals; the possible relationships between these time intervals can be expressed using Allen's relations [2], which are defined as follows:

Allen's relation	Definition	Example
`overlaps(o)`	$[s,e)$ o $[s',e') = \{[s,e),[s',e') : s < s' < e < e'\}$	$[2010, 2013)$ o $[2011, 2014)$
`during(d)`	$[s,e)$ d $[s',e') = \{[s,e),[s',e') : s' < s < e < e'\}$	$[2010, 2013)$ d $[2009, 2014)$
`before(≺)`	$[s,e) \prec [s',e') = \{[s,e),[s',e') : s < e < s' < e'\}$	$[2010, 2013) \prec [2014, 2015)$
`meets(m)`	$[s,e)$ m $[s',e') = \{[s,e),[s',e') : s < e = s' < e'\}$	$[2010, 2013)$ m $[2013, 2014)$
`starts(s)`	$[s,e)$ s $[s',e') = \{[s,e),[s',e') : s' = s < e < e'\}$	$[2010, 2013)$ s $[2010, 2014)$
`finishes(f)`	$[s,e)$ f $[s',e') = \{[s,e),[s',e') : s' < s < e = e'\}$	$[2010, 2013)$ f $[2009, 2013)$
`equals(=)`	$[s,e) = [s',e') = \{[s,e),[s',e') : s = s' \text{ and } e = e'\}$	$[2010, 2013) = [2010, 2013)$

Temporal Relational Databases. A *relational schema* is a finite collection $\mathbf{R}$ of relation symbols of the form $R(A_1, \ldots, A_k)$, where $A_1, \ldots, A_k$ 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.

A *temporal relation symbol* is a relation symbol R in which one or more attributes are designated as temporal attributes, i.e., they can only take temporal values. In this article, we assume that every temporal relation symbol has exactly one temporal attribute, which, without loss of generality, is the last attribute on 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, while an *abstract* $\mathbf{R}$-*instance* is an $\mathbf{R}$-instance in which the values of the temporal attributes are time points. (The values of all the other attributes belong to the countable domain Const of objects called *constants*.) In this article, we will discuss the temporal data-exchange problem in the concrete model of time. Therefore, in

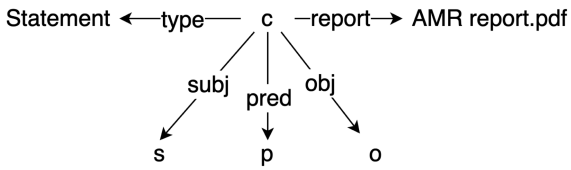

Fig. 1. Example of reification being applied to triple (s, p, o) in an RDF graph.

the remainder of this article, the term *relational schema* refers to the *temporal relational schema*, and *relational instance* refers to the *concrete instance* over a temporal relational schema. The *active domain of a* relational instance is the set of all the values in $\textsc{Const} \cup T_I$ that occur in it.

Resources and RDF Graphs. In the RDF model, all things, i.e., web documents and real-world objects, are modeled as *resources* using **universal resource identifiers *(URIs)***. Assume an infinite set U of RDF URI references, 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 $(U \cup B) \times U \times (U \cup B \cup L)$, 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. 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. Property "rdfs:domain" in the statement $(p, \text{rdfs:domain}, c)$ 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'. There are two types of properties, built in and domain specific. 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] [23].

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 [10, 22].

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 1 shows an RDF graph stating that the triple (statement) (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 $s \xrightarrow{p} o$.)

3 RELATIONAL-TO-RDF TEMPORAL DATA EXCHANGE: THE FOUNDATIONS

In this section 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}: $\textsc{Const} \rightarrow U$ 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

Table 1. The Temporal Classes in the Temporal RDF Vocabulary

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

iff at least one of its arguments is a blank node; (ii) F_{ruri} is injective; and (iii) the set B^{+}_{ruri} of blank nodes in the range of F_{ruri} is such that the set $B - B^{+}_{ruri}$ has an infinite computable subset. (Condition (iii) is not burdensome, as we can, e.g., allocate to B^{+}_{ruri} just the odd-numbered values in an enumeration of the values in B.) We will use in our chase definitions in Section 3.3 any such infinite computable subset of $B - B^{+}_{ruri}$, and will refer to the selected subset as B^{-}_{ruri}.

In the remainder of the article we will use the following reserved alphabets:

— $x, y, \ldots$ for non-temporal variables (set V) and t for temporal variables (set T) 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 infinite and disjoint from each other and from Const.

3.1 Basic Concepts, Temporal RDF Graph Schemas, and Instances

In this subsection we review the notion of temporal annotations of RDF graphs due to [24], 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 conditions under which they enable temporal annotations of data in a structural way [26].

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, that is, classes for statements about temporal statements. A *temporal property* is a property that is connected to at least one temporal class; we consider temporal properties *temporal*, *interval*, *validFor* (with values in *timeInterval*), and the properties that describe Allen's relations between time intervals, such as `meets` or `during`. The *temporal RDF vocabulary* is the result of adding these temporal classes and properties to the RDF vocabulary, see Tables 1 and 2. We call the set W = {`sc,sp,type,dom,range,subj,obj,pred`} the *RDFS vocabulary*, and call the set $W^{*} = W \cup$ {`temporal, interval, validFor, overlaps, during, before, meets, starts, finishes, equals`} the *temporal RDFS 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 $\mathcal{T}_t^{RDF} = \{(x, temporal, y), (y, interval, z), (z, validFor, w)\}$ 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 $\mu = \{x/k, y/l, z/m, w/i\}$, 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 [24], 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

Table 2. The Temporal Properties in the Temporal RDF Vocabulary

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
equals	provides equality relation between intervals	Interval	Interval
overlap	provides overlap relation between intervals	Interval	Interval
during	provides during relation between intervals	Interval	Interval
meets	provides meets relation between intervals	Interval	Interval
starts	provides starts relation between intervals	Interval	Interval
finishes	provides finishes relation between intervals	Interval	Interval
before	provides before relation between intervals	Interval	Interval

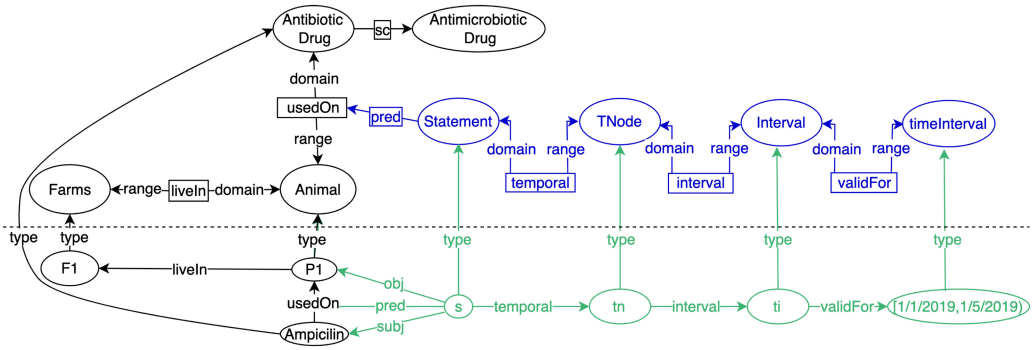

Fig. 2. Example of a temporal RDF graph schema (above the dashed line) and its RDF instance (above and below the line); the notation we use originates from [3]. Here, an RDF triple *Ampicillin-[is]-usedOn-P1* is adorned with a temporal annotation shown in green. The annotation includes an instance of 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.

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 structure for* (s, p, o) *and* i *is well typed* if G also contains the triples $(\mu(y), type, TNode)$, $(\mu(z), type, Interval)$, and $(i, type, timeInterval)$.

Example 1. In Figure 2, the triples $(s, temporal, tn)$, $(tn, interval, ti)$, and $(ti, validFor, [1/1/2019, 1/5/2019])$ shown in green are temporal triples, as they result from 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, together with the triples $(s, type, Statement)$, $(s, subj, Ampicillin)$, $(s, pred, usedOn)$, and $(s, obj, P1)$, comprise a temporal-annotation structure for the RDF triple $(Ampicillin, usedOn, P1)$ and $[1/1/2019, 1/5/2019]$. The structure is well typed, as the triples $(tn, type, TNode)$, $(ti, type, Interval)$, and $([1/1/2019, 1/5/2019], type, timeInterval)$ are all included in the graph.

We call the set of triples $\mathcal{T}_t^{RDFS} = \{(temporal, domain, Statement), (temporal, range, TNode), (interval, domain, TNode), (interval, range, Interval), (validFor, domain, Interval), (validFor, range, timeInterval)\}$ 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 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* $\mathcal{O}^T$ 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 element of the temporal RDFS markup set $\mathcal{T}_t^{RDFS}$, 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 $\mathcal{O}^T = (\mathbf{C}, \mathbf{P}, G^T)$ 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.

Example 2. Consider a temporal RDF graph schema $\mathcal{O}^T = (\mathbf{C}, \mathbf{P}, G^T)$ 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 2.

RDF Instances of Temporal RDF Graph Schemas. Let $\mathcal{O}^T = (\mathbf{C}, \mathbf{P}, G^T)$ 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, type, s) and (b, type, o).

Then we call the RDF graph $J = G^T \cup H$ an *RDF instance* of $\mathcal{O}$. (Note that J includes the graph G^T of $\mathcal{O}$.) We say that an RDF instance J of a well-formed RDF graph schema $\mathcal{O}^T = (\mathbf{C}, \mathbf{P}, G^T)$ 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 3. Figure 2 shows a well-formed instance of the temporal RDF graph schema of Example 2.

An RDF graph schema can also be *atemporal.* In this case, the above definition is modified for $\mathcal{O}^A = (\mathbf{C}, \mathbf{P}, G^A)$ 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 the remainder of the article, 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 [24] of temporal annotations being done structurally; as explained in [26], reification is the only option for structural annotations.

3.2 Temporal Constraints

We begin by defining the notion of *term*, as follows: (a) All variables are terms; (b) the names of all properties and of all classes are terms; (c) all constant symbols are terms; and (d) whenever x, y, and z are terms, then the URI constructors $F_{uri}(x)$ and $F_{ruri}(x, y, z)$ are also terms (we call the latter terms *functional terms).* For a temporal RDF graph schema $O^T = \{\mathbf{C}, \mathbf{P}, G^T\}$, we call an expression of the form (x, p, y), $(x, \texttt{type}, c)$, or $(x, \texttt{validFor}, t)$, in which x and y are variables, $c \in \mathbf{C}$, t is a temporal variable, and $p \in \mathbf{P} \cup \mathbf{W}^*$, an *atomic formula over* O^T.

Let $\mathcal{S}^T$ be a temporal relational schema and O^T a temporal RDF graph schema; we refer to $\mathcal{S}^T$ as the *source schema,* and to $O^{\mathbf{T}}$ as the *target schema.* We consider relational-to-RDF temporal data exchange from $\mathcal{S}^T$ to O^T, which can be formalized using constraints in some logical formalism that describes the relationship between the two schemas. Similarly to relational-to-relational temporal data exchange [13, 21], in relational-to-RDF temporal data exchange we consider constraints called *temporal source-to-target tuple generating dependencies (temporal s-t tgds).* A *relational-to-RDF temporal s-t tgd from* $\mathcal{S}^T$ *to* O^T is a **first-order logic** (**FOL**) sentence of the form $\forall \mathbf{x}, \mathbf{t}\, (\varphi(\mathbf{x}, \mathbf{t}) \wedge \pi(\mathbf{t}) \rightarrow \exists \mathbf{y}\, \psi(\mathbf{x}, \mathbf{y}, \mathbf{t}))$. Here, $\mathbf{t}$ are the only temporal variables (which may be absent); $\varphi(\mathbf{x}, \mathbf{t})$ is a conjunction of atomic formulas over the schema $\mathcal{S}^T$; the formula $\pi(\mathbf{t})$ is a Boolean combination of *Allen atomic formulas* of the form $t_1\ \delta\ t_2$ involving temporal variables in $\mathbf{t}$ and with $\delta \in \{\mathsf{o}, \mathsf{d}, \prec, \mathsf{m}, \mathsf{s}, \mathsf{f}, =\}$ (see Section 2); and $\psi(\mathbf{x}, \mathbf{y}, \mathbf{t})$ is a conjunction of atomic formulas over O^T. A temporal s-t tgd is *full* if it has no existential variables. In the sequel, the term *temporal s-t tgds* refers to the relational-to-RDF s-t tgds with either no temporal variables or with at least one temporal variable.

Example 4. Let $\mathcal{S}^T$ be a source schema with the temporal relation symbol *DrugUsage(farm, animal, drug, time),* with *time* taking values in T_I, and let O^T be the temporal RDF graph schema of Example 2. Consider the following temporal s-t tgd from $\mathcal{S}^T$ to O^T:

$$\begin{aligned}\forall x_1, x_2, x_3, t\ \Big(&DrugUsage(x_1, x_2, x_3, t) \rightarrow (F_{uri}(x_1), \text{type}, \text{Farm}) \wedge (F_{uri}(x_2), \text{type}, \text{Animal})\\ &\wedge (F_{uri}(x_2), \text{livesIn}, F_{uri}(x_1)) \wedge (F_{uri}(x_3), \text{type}, \text{AntibioticDrug}) \wedge (F_{uri}(x_3), \text{usedOn}, F_{uri}(x_2))\\ &\wedge (d_1, \text{subj}, F_{uri}(x_3)) \wedge (d_1, \text{pred}, \text{usedOn}) \wedge (d_1, \text{obj}, F_{uri}(x_2)) \wedge (d_1, \text{temporal}, d_2)\\ &\wedge (d_2, \text{interval}, d_3) \wedge (d_3, \text{validFor}, t)\Big) \wedge (d_1, \text{type}, \text{Statement}) \wedge (d_2, \text{type}, \text{TNode})\\ &\wedge (d_3, \text{type}, \text{Interval}) \wedge (t, \text{type}, \text{timeInterval})).\end{aligned} \tag{1}$$

Here, we use d_1, d_2, and d_3 to represent terms that use the URI constructors F_{uri} and F_{ruri}:

$d_1 = F_{ruri}(F_{uri}(x_2), \text{usedOn}, F_{uri}(x_3))$;

$d_2 = F_{ruri}(F_{ruri}(F_{uri}(x_2), \text{usedOn}, F_{uri}(x_3)), \text{temporal}, t) = F_{ruri}(d_1, \text{temporal}, t)$;

$d_3 = F_{ruri}\Big(F_{ruri}(F_{ruri}(F_{uri}(x_2), \text{usedOn}, F_{uri}(x_3)), \text{temporal}, t), \text{interval}, t) = F_{ruri}(d_2, \text{interval}, t\Big)$.

(Recall that URI constructors F_{uri} and F_{ruri} map terms into URI identifiers.)

The intuition for this s-t tgd is that it translates a tuple in the relation *DrugUsage* in the source schema $\mathcal{S}^T$ into the structural format that conforms to the target schema O^T, complete with the temporal-annotation structure. For the case where the source tuple is *("F1," "P1," "Ampicillin," [1/1/2019,1/5/2019]),* the resulting structure is shown in the bottom half of Figure 2.

We say that a temporal s-t tgd σ with target schema $O^T = (\mathbf{C}, \mathbf{P}, G^T)$ is *well formed* if there exists a substitution μ that replaces temporal variables in the **right-hand side** ***(RHS)*** of σ with values in T_I and consistently replaces the remaining terms in σ with elements of $U \cup B \cup L$, such that μ turns the RHS of σ into a temporal RDF graph H such that $G^T \cup H$ is a well-formed RDF instance

of $\mathcal{O}^T$. For instance, the s-t tgd of Example 4 is well formed. In the remainder of this section and in Section 5 we will assume that all the given temporal s-t tgds with temporal variables are well formed.

Temporal Relational-to-RDF Schema Mappings. A *temporal relational-to-RDF schema mapping* $\mathcal{M}^T$ is a tuple $(\mathcal{S}^T, \mathcal{O}^T, \Sigma^T)$, where $\mathcal{S}^T$ is the source relational schema, $\mathcal{O}^T$ is the target temporal RDF graph schema, and Σ^T is a set of temporal s-t tgds from $\mathcal{S}^T$ to $\mathcal{O}^T$, such that at least one s-t tgd in Σ^T has temporal variables on its right-hand side.

Values in Source Instances and Target RDF Instances. In relational-to-RDF temporal data exchange, the set of values in each source instance is a subset of $\textsc{Const} \cup T_I$, while RDF instances require URIs, literals, or time intervals. For this reason, some data in the target RDF instances are generated by temporal relational-to-RDF schema mappings with the help of URI constructors, that is, functions that map particular source constants to URIs. Other source constants are transferred by schema mappings into the target RDF instances as literals, and the time-interval values in the sources are transferred into the targets without changes. The *universe* of a target RDF instance J, $universe(J)$, is the set of elements of $\{U \cup B \cup L \cup T_I\}$ that occur in the triples of J.

Relational Formula Homomorphisms, RDF Formula, and Instance Homomorphisms. Let I be an instance over a relational schema $\mathcal{S}^T$, and let σ be a temporal s-t tgd with source schema $\mathcal{S}^T$. A *(relational formula) homomorphism for σ and I* is a map h from the variables in the **left-hand side (*LHS*)** of σ to values in I, such that for every atom $R(\mathbf{x})$ in the LHS of σ, I includes the tuple $R(h(\mathbf{x}))$.

Consider an existential conjunctive formula $F = \exists \mathbf{y}\ \phi(\mathbf{x}, \mathbf{y}, \mathbf{t})$ over a temporal RDF graph schema $\mathcal{O}^T$. Let h be a map that maps the variables $\mathbf{x}$ in F to $\textsc{Const}$, the variables $\mathbf{t}$ in F to T_I, and the remaining terms in F to $U \cup B \cup L$, such that (a) $h(c) = c$ for all $c \in \mathbf{C}$, and (b) for functional terms of the form $F_{uri}(w)$ or $F_{ruri}(w_1, p, w_2)$, $h(F_{uri}(w)) = F_{uri}(h(w))$ and $h(F_{ruri}(w_1, p, w_2)) = F_{ruri}(h(w_1), p, h(w_2))$. If, for an RDF instance J of $\mathcal{O}^T$, J includes the triple $(h(w_1), p, h(w_2))$ for each atom (w_1, p, w_2) in $\phi(\mathbf{x}, \mathbf{y}, \mathbf{t})$, then we call h an *(RDF formula) homomorphism for F and J*.

Let J and J' be two RDF instances over a temporal RDF graph schema $\mathcal{O}^T$. An *(RDF instance) homomorphism from J to J'* is a map h from $universe(J)$ to $universe(J')$, such that (a) $h(u) = u$ for all $u \in U \cup L \cup T_I \subseteq universe(J)$; (b) for each blank node $b \in B \subseteq universe(J)$, we have $h(b) \in U \cup L \cup B \subseteq universe(J')$; and (c) for each triple $(s, p, o) \in J$, we have that $(h(s), p, h(o)) \in J'$.

Solutions. Let $\mathcal{M}^T = (\mathcal{S}^T, \mathcal{O}^T, \Sigma^T)$ be a temporal relational-to-RDF schema mapping with target schema $\mathcal{O}^T = (\mathbf{C}, \mathbf{P}, G^T)$, and let I be a source instance over $\mathcal{S}^T$. An RDF instance J is a *solution for I w.r.t. $\mathcal{M}^T$* if the following holds: For each σ: $\forall \mathbf{x}, \mathbf{t}\ (\varphi(\mathbf{x}, \mathbf{t}) \wedge \pi(\mathbf{t}) \rightarrow \exists \mathbf{y}\ \psi(\mathbf{x}, \mathbf{y}, \mathbf{t}))$ in Σ^T, with $\mathbf{x} = [x_1, \ldots, x_n]$ (we abbreviate this as $|\mathbf{x}| = n$), $|\mathbf{y}| = l$, and $|\mathbf{t}| = m$, and for the tuple of constants $\mathbf{a}$ with $|\mathbf{a}| = n$ and tuple of time intervals $\mathbf{i}$ with $|\mathbf{i}| = m$ from the active domain of I such that $h(\mathbf{x}) = \mathbf{a}$, $h(\mathbf{t}) = \mathbf{i}$, and $I \models \varphi(\mathbf{a}, \mathbf{i}) \wedge \pi(\mathbf{i})$, there exists a tuple $\mathbf{b}$ with $|\mathbf{b}| = n + l$ of elements of $U \cup B \cup L \subseteq universe(J)$, such that (1) $J \models \psi(\mathbf{a}, \mathbf{b}, \mathbf{i})$, and (2) J is a well-formed instance of $\mathcal{O}^T$.

An RDF instance J is a *universal solution for I w.r.t. $\mathcal{M}^T$* if (i) J is a solution for I w.r.t. $\mathcal{M}^T$, and (ii) there exists a homomorphism from J to J' for every solution J' for I w.r.t. $\mathcal{M}^T$.

We will be using the term *combined instance* to indicate a tuple $K = (I, J)$, in which I is a relational instance and J is an RDF instance of the given temporal RDF graph schema; we refer to I in K as $K.I$, and to J in K as $K.J$. For a combined instance $K = (I, J)$ and an s-t tgd σ, we call a relational formula homomorphism from the LHS of σ to $K.I$ a *(formula) homomorphism for σ and K*.

Let $K = (I, J)$ and $K' = (I', J')$ be two combined instances. We say that *there is a homomorphism from K to K'* if there is a (relational) homomorphism from I to I' and a homomorphism from J to J'.

3.3 Chase for Temporal Relational-to-RDF Schema Mappings

Let $\mathcal{M}^T = (\mathcal{S}^T, \mathcal{O}^T, \Sigma^T)$ be a temporal relational-to-RDF schema mapping, and I a source instance over $\mathcal{S}^T$. We propose a chase algorithm to generate a solution for I w.r.t. $\mathcal{M}^T$. In the process of chasing the instance $K = (I, J)$ w.r.t. $\mathcal{M}^T$ (with J set initially to the graph G^T in $\mathcal{O}^T$), for every constant in I mapped into an instance of a domain-specific class of $\mathcal{O}^T$, the URI constructor F_{uri} generates the corresponding URI value in J.

Definition 1 (Chase Step). For a given $\mathcal{M}^T = (\mathcal{S}^T, \mathcal{O}^T, \Sigma^T)$, 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. We denote by B_{curr} the set of all the blank nodes in J_{curr}. Let σ: $\forall \mathbf{x}, \mathbf{t}\ (\varphi(\mathbf{x}, \mathbf{t}) \wedge \pi(\mathbf{t}) \rightarrow \exists \mathbf{y}\ \psi(\mathbf{x}, \mathbf{y}, \mathbf{t}))$ be a temporal s-t tgd in Σ^T, in which $|\mathbf{x}| = n$, $|\mathbf{y}| = l$, and $|\mathbf{t}| = m$, with $n, l, m \geq 0$.[8] 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 $K \not\models \psi(\mathbf{c}, \mathbf{z}, \mathbf{i})$ for each vector $\mathbf{z}$ of values in $U \cup B$. In this case, we extend h to map each y_j in $\mathbf{y}$ to a fresh blank node a_j w.r.t. J_{curr} from the set B^-_{ruri}. (That is, $\mathbf{a} = [a_1, \ldots, a_l]$ is a vector of distinct values in $B^-_{ruri} - B_{curr}$.) We denote this extension of h by h'. Adding to J_{curr} the set of triples $\psi(\mathbf{c}, \mathbf{a}, \mathbf{i})$ gives rise to a new combined instance K'; by construction, $K' \models \varphi(\mathbf{c}, \mathbf{i}) \wedge \pi(\mathbf{t}) \wedge \psi(\mathbf{c}, \mathbf{a}, \mathbf{i})$. We say that *$K'$ is obtained from K via a chase step with σ and h'*, and write $K \xrightarrow{\sigma, h'} K'$.

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

- We define a *chase sequence for I w.r.t $\mathcal{M}^T$* 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 $\mathcal{M}^T$* 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 $\mathcal{M}^T$.

Theorem 1. *Let $\mathcal{M}^T = (\mathcal{S}^T, \mathcal{O}^T, \Sigma^T)$, with $\mathcal{O}^T = (\mathbf{C}, \mathbf{P}, G^T)$, be a temporal relational-to-RDF 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. $\mathcal{M}^T$.*

Proof. Let $K^{'}$ be an arbitrary solution for I w.r.t. $\mathcal{M}^T$; thus $K^{'}$ satisfies Σ^T. Observe that the identity mapping from (I, G^T) to $K^{'}$ is a homomorphism, and $K = (I, J)$ satisfies Σ. By repeatedly applying Lemma 1 at each chase step starting with $K_0 = (I, G^T)$, we obtain a homomorphism h from K to $K^{'}$. We conclude that J in K is a universal solution I w.r.t. $\mathcal{M}^T$. □

Lemma 1. *Let $\mathcal{M}^T = (\mathcal{S}^T, \mathcal{O}^T, \Sigma^T)$, with $\mathcal{O}^T = (\mathbf{C}, \mathbf{P}, G^T)$, be a temporal relational-to-RDF schema mapping, and I a source instance. Let $K_0 = (I, G^T)$. In a chase sequence $K_0, K_1, \ldots$ for I w.r.t. $\mathcal{M}^T$, 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: $K_{j-1} \rightarrow K$. Then there exists a homomorphism h': $K_j \rightarrow K$.*

Proof. The temporal s-t tgd σ is of the form $\forall \mathbf{x}, \mathbf{t}\ (\varphi(\mathbf{x}, \mathbf{t}) \wedge \pi(\mathbf{t}) \rightarrow \exists \mathbf{y}\ \psi(\mathbf{x}, \mathbf{y}, \mathbf{t}))$, with the vector $\mathbf{y} = [y_1, \ldots, y_l]$ having $l \geq 0$ elements. By definition of the chase step, there exists a relational formula homomorphism $h'_j : \varphi(\mathbf{x}, \mathbf{t}) \wedge \pi(\mathbf{t}) \rightarrow K_{j-1}$. Since composing homomorphisms yields homomorphisms, $h'_j \circ h : \varphi(\mathbf{x}, \mathbf{t}) \wedge \pi(\mathbf{t}) \rightarrow K$ is a homomorphism. In addition, since $K \models \sigma$, there exists a homomorphism $h'' : \varphi(\mathbf{x}, \mathbf{t}) \wedge \pi(\mathbf{t}) \wedge \psi(\mathbf{x}, \mathbf{y}, \mathbf{t}) \rightarrow K$, such that h'' is an extension of $h'_j \circ h$, i.e.,

[8] In those cases where $m = 0$, we take the conjunct $\pi(\mathbf{t})$ in the LHS of σ to be $true$.

$h(h'_j(\mathbf{x})) = h''(\mathbf{x})$ and $h(h'_j(\mathbf{t})) = h''(\mathbf{t})$. We denote the vector $h''(\mathbf{y}) = [c''_1, \ldots, c''_l]$ by $\mathbf{c}''$; observe that for each element c''_m of $\mathbf{c}''$, $c''_m \in U \cup B$.

Consider now the homomorphism $h_j : \varphi(\mathbf{x}, \mathbf{t}) \wedge \pi(\mathbf{t}) \wedge \psi(\mathbf{x}, \mathbf{y}, \mathbf{t}) \rightarrow K_j$. 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 denote the vector $h_j(\mathbf{y}) = [c_1^{(j)}, \ldots, c_l^{(j)}]$ by $\mathbf{c}^{(\mathrm{j})}$. By construction, for each element $c_m^{(j)}$ of $\mathbf{c}^{(\mathrm{j})}$ we have that $c_m^{(j)}$ is a fresh blank node that does not occur either in the set of blank nodes in K_{j-1} or in the range of the function F_{ruri}.

We define the homomorphism h' as follows: (1) h' maps each element of the vector $\mathbf{c}^{(\mathrm{j})}$ to the corresponding element of the vector $\mathbf{c}''$; and (2) 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 $\psi(\mathbf{x}, \mathbf{y}, \mathbf{t})$, 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 $p_1, \ldots, p_r$. 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 $p_1, \ldots, p_r$. By definition of the chase step, the triples in the set $h_j(\psi(\mathbf{x}, \mathbf{y}, \mathbf{t}))$ 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. □

4 APPLYING RELATIONAL-TO-RDF TEMPORAL DATA EXCHANGE IN PRACTICE

Given a temporal relational-to-RDF schema mapping $\mathcal{M}^T = (\mathcal{S}^T, \mathcal{O}^T, \Sigma^T)$ and a source instance I over $\mathcal{S}^T$, temporal information can be exchanged from I into a target RDF instance J over $\mathcal{O}^T$ via the chase algorithm of Section 3.3. At the same time, domain ontologies in many real-life applications are atemporal, i.e., they do not have temporal components. (In the remainder of this article, we will use the term *ontology* to refer to the RDF graph G in some temporal or atemporal RDF graph schema $\mathcal{O} = (\mathbf{C}, \mathbf{P}, G)$, with $\mathbf{C}$ and $\mathbf{P}$ understood from the context.) Thus, domain experts would only be able to provide s-t tgds whose RHS would not contain temporal variables, even though their LHS could contain temporal variables. As a result, some real-life applications would give rise to *atemporal relational-to-RDF schema mappings* $\mathcal{M}^A = (\mathcal{S}^T, \mathcal{O}^A, \Sigma^A)$, in which $\mathcal{S}^T$ is a *temporal* relational schema and $\mathcal{O}^A$ is an *atemporal* RDF graph schema. Further, each s-t tgd in Σ^A is a FOL sentence of the form σ: $\forall \mathbf{x}, \mathbf{t}\ (\varphi(\mathbf{x}, \mathbf{t}) \wedge \pi(\mathbf{t}) \rightarrow \exists \mathbf{y}\ \psi(\mathbf{x}, \mathbf{y}))$, such that the elements of $\mathbf{t}$ (if present) are the only temporal variables in σ, $\varphi(\mathbf{x}, \mathbf{t})$ is a conjunction of atomic formulas over $\mathcal{S}^T$, $\pi(\mathbf{t})$ is a Boolean combination of Allen atomic formulas,[9] and $\psi(\mathbf{x}, \mathbf{y})$ is a conjunction of atomic formulas over $\mathcal{O}^A$. While our chase algorithm of Section 3.3 would still work correctly with atemporal relational-to-RDF schema mappings, its solutions would be atemporal RDF instances. Thus, the target RDF instances would miss the temporal information from the source.

To address this issue of loss of temporal information from the source, we introduce Algorithm 1, which is used as a preprocessing step that enables temporal data exchange in the situation where the given relational-to-RDF schema mapping is atemporal. Given an $\mathcal{M}^A = (\mathcal{S}^T, \mathcal{O}^A, \Sigma^A)$, Algorithm 1 works by converting the atemporal RDF graph schema $\mathcal{O}^A$ (respectively Σ^A) in $\mathcal{M}^A$ into its temporal version $\mathcal{O}^T$ (respectively Σ^T). This is achieved by temporally enriching both the atemporal ontology G^A and the corresponding s-t tgds in Σ^A. Algorithm 1 is based on straightforward declarative 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 those s-t tgds in Σ^A whose LHS has temporal variables. The patterns use the temporal RDF template of Section 3.1 to reify triples on the RHS of each relevant s-t tgd in

[9] In those cases where $\mathbf{t}$ is empty, we take the conjunct $\pi(\mathbf{t})$ in the LHS of σ to be $true$.

ALGORITHM 1: Temporal enrichment of relational-to-RDF schema mappings and instances.

Data: Atemporal relational-to-RDF schema mapping $\mathcal{M}^A = (\mathcal{S}^T, \mathcal{O}^A, \Sigma^A)$, with $\mathcal{O}^A = (\mathbf{C}, \mathbf{P}, G^A)$, and instance I over $\mathcal{S}^T$.
Result: Temporally enriched version $\mathcal{M}^T = (\mathcal{S}^T, \mathcal{O}^T, \Sigma^T)$ of $\mathcal{M}^A$ and RDF instance J over $\mathcal{O}^T$ that is a solution for I w.r.t. $\mathcal{M}^T$.
begin
  $\Sigma^T \leftarrow \Sigma^A$; $\mathcal{O}^T \leftarrow \mathcal{O}^A$; $G^T \leftarrow G^A$; `// initialization`
  for *each atom* (s, p, o) *on the right-hand side of each* $\sigma \in \Sigma^T$ **do**
    if (s, p, o) *is in the temporal-enrichment scope of* σ **then**
      $\Sigma^T \leftarrow$ result of temporally enriching (s, p, o) in σ; `// first stage`
      $\mathcal{O}^T \leftarrow$ result of temporally enriching the p-related part of the graph G^T in $\mathcal{O}^T$; `// second stage`
  $J \leftarrow$ universal solution for I w.r.t. $\mathcal{M}^T = (\mathcal{S}^T, \mathcal{O}^T, \Sigma^T)$; `// third stage`
  return $\mathcal{M}^T$ and J;

such a way that the resulting temporal s-t tgd is well formed. (Recall that we employ reification [26] to implement temporal annotations in target instances so as to allow storing the RDF results of materializing temporal source data by using graph DBMSs that do not have any special features.) In the second stage, the atemporal input ontology G^A is augmented with temporal markups (see Section 3.1) of the properties for which temporal-enrichment atom patterns have been added to s-t tgds in Σ^A. Once both the input RDF graph schema $\mathcal{O}^A$ and the input set of s-t tgds Σ^A have been thus temporally enriched, the resulting s-t tgds Σ^T are used in the third stage of Algorithm 1 to exchange the input temporal source instance I into the temporally-aware format consistent with the (now) temporal target RDF graph schema $\mathcal{O}^T$. The following result holds by construction of Algorithm 1.

Proposition 1. *Given an atemporal relational-to-RDF schema mapping $\mathcal{M}^A = (\mathcal{S}^T, \mathcal{O}^A, \Sigma^A)$ and a temporal relational instance I over $\mathcal{S}^T$, the elements $\mathcal{O}^T$ and Σ^T of the temporally enriched version $\mathcal{M}^T$ of $\mathcal{M}^A$ and the RDF instance J over $\mathcal{O}^T$ returned by Algorithm 1 are all well formed.*

As an illustration, consider the scenario of Example 4, with its temporal source relation *DrugUsage(farm, animal, drug, time),* which records the temporal history of AMR drug usage for animals in farms. Recall that in Example 4, the relation has a single tuple *("F1," "P1," "Ampicillin," [1/1/2019,1/5/2019]).* Suppose that AMR scientists are interested in obtaining answers to temporal queries posed using the domain-specific ontology terminology shown in black in the top half of Figure 2. As the domain-specific part of the ontology is atemporal, the best way to exchange data from the *DrugUsage* source to a target consistent with that part of the ontology would be to use the s-t tgd

$$\forall x_1, x_2, x_3, t \Big(DrugUsage(x_1, x_2, x_3, t) \rightarrow (F_{uri}(x_1), \text{type}, \text{Farm}) \wedge (F_{uri}(x_2), \text{type}, \text{Animal}) \\ \wedge (F_{uri}(x_2), \text{livesIn}, F_{uri}(x_1)) \wedge (F_{uri}(x_3), \text{type}, \text{AntibioticDrug}) \wedge (F_{uri}(x_3), \text{usedOn}, F_{uri}(x_2))\Big). \tag{2}$$

Applying this s-t tgd to the source relation would result in the data shown in black in the bottom half of Figure 2. Clearly, AMR scientists cannot get from these RDF 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 data transferred from the source using the s-t tgd of Equation (2).

This problem can be solved by applying Algorithm 1 to the above ontology, s-t tgd, and data source. The algorithm will yield the *temporal* s-t tgd of Example 4 shown in Equation (1), as well as the *temporal* ontology and RDF instance shown in Figure 2. We assume that domain experts can mark up individual atoms on the RHS of the s-t tgds that are provided as inputs to Algorithm 1, to indicate the specific target RDF predicates that would receive the temporal annotations. e.g., in the case of this example, we assume that domain experts have so marked the last atom in the RHS of the s-t tgd of Equation (2). This is the meaning of the "temporal-enrichment scope" criterion in Algorithm 1. The alternative, which is also feasible in Algorithm 1, is to allow the algorithm to temporally annotate *all* the domain-specific target RDF predicates used in the input s-t tgds.

5 QUERYING INSTANCES OF TEMPORAL RDF GRAPH SCHEMAS

In this section, we address the problem of answering temporal queries on the temporal RDF graph schemas and RDF instances that are obtained by relational-to-RDF temporal data exchange. As the starting point for the formal template for the queries considered in this article, we take the query language for RDF called SPARQL [16, 33, 36]. We will focus on temporal queries expressed by the conjunctive fragment of SPARQL that uses only the AND operator and SELECT clause [19]. (In Section 5.4 and Section 6 we will extend our consideration to queries with Allen interval relations.) Temporal queries in this fragment of SPARQL can return temporal information by matching up temporal triples in the given temporal RDF graph instances, and are also allowed to match up `subClassOf` and `subPropertyOf` hierarchies in the corresponding ontologies. We show that such queries can be correctly answered either by reasoning on the universal solutions returned by relational-to-RDF temporal data exchange, or by applying rewritings of the queries over the relevant temporal RDF graph schemas to the universal solutions. Our notion of correctness of query answering is expressed precisely by the concept of *certain answers,* which refers to the intersection of all possible answers to the given queries over the solutions. We also introduce a user interface that makes it easier for domain experts to construct, rewrite, and obtain correct answers to temporal queries in real-world applications.

5.1 Reasoning on Universal Solutions: Querying with Normal Forms of RDF Instances

5.1.1 *Basic Concepts.*

Deductive System for RDF. One can add more information to a given RDF graph by using rules:

$$\frac{(A, \text{type}, \mathit{prop})}{(A, \text{sp}, A)}(3) \qquad \frac{(A, \text{sp}, B)(B, \text{sp}, C)}{(A, \text{sp}, C)}(4) \qquad \frac{(A, \text{sp}, B)(X, A, Y)}{(X, B, Y)}(5)$$

$$\frac{(A, \text{type}, \mathit{class})}{(A, \text{sc}, A)}(6) \qquad \frac{(A, \text{sc}, B)(B, \text{sc}, C)}{(A, \text{sc}, C)}(7) \qquad \frac{(A, \text{sc}, B)(X, \text{type}, A)}{(X, \text{type}, B)}(8)$$

$$\frac{(A, \text{dom}, C)(X, A, Y)}{(X, \text{type}, C)}(9) \qquad \frac{(A, \text{range}, D)(X, A, Y)}{(Y, \text{type}, D)}(10)$$

Here we only list the rules that are relevant to this article. For more details, see [23, 25, 31].

Temporal Entailment. Let G_1 and G_2 be two temporal RDF graphs. We use the definition of [23] of G_1 *entailing* G_2, denoted by $G_1 \models G_2$: $G_1 \models G_2$ if for each interpretation $\mathcal{I}$ such that $\mathcal{I} \models G_1$, we have $\mathcal{I} \models G_2$. Furthermore, G_1 and G_2 are *equivalent,* denoted $G_1 \equiv G_2$, iff $G_1 \models G_2$ and $G_2 \models G_1$. We use the soundness and completeness results for the deductive system of [31] that allow us to claim that $G_1 \models G_2$ if and only if there is a proof of G_2 from G_1 using the above deductive rules. (We omit the relevant definitions and results of [23, 31] due to the space limit.)

Graph Closures, Cores, and Normal Forms. An *RDF-closure* of a temporal RDF graph G, denoted by $cl(G)$, is defined as the closure of G under the deductive rules of [31], see Equation (3)–(10). A temporal RDF graph G is *lean* if there is no map μ such that $\mu(G)$ is a proper subset of G. It follows from the results of [23] that each temporal RDF graph G contains a lean subgraph, called *core of* G and denoted $core(G)$, that is unique up to isomorphism.[10] The RDF-closures and cores are defined similarly for temporal RDF graph schemas and RDF instances. For each graph G that is a temporal RDF graph, ontology in a temporal RDF graph schema, or RDF instance, $core(G)$ is the unique (up to isomorphism) minimal (w.r.t. number of triples) graph that is equivalent to G.

We now consider the notion of the normal form [23] of a temporal RDF graph, needed to define answers to queries on graphs. This notion arises due to the potential presence in graphs of blank nodes, which may give rise to non-unique closure and core representations of equivalent graphs. (See [23] for the details.) The *normal form of a temporal RDF graph* G is defined as the core of its closure: $nf(G) = core(cl(G))$. We use the same definition of normal form for temporal RDF graph schemas and RDF instances. (It is easy to see that if a temporal RDF graph schema or RDF instance is well formed, then so are its closure and its normal form.) As a straightforward extension of the results of [23], we have that $nf(G)$ is unique (up to isomorphism) and syntax independent (i.e., $G \equiv H$ iff $nf(G) \cong nf(H)$) in case G is a temporal RDF graph, ontology in a temporal RDF graph schema, or RDF instance. The claim of Theorem 2, next, follows from the results of [23, 25].

Theorem 2. *Let G_1 and G_2 be two temporal RDF graphs, ontologies in temporal RDF graph schemas, or RDF instances. $G_1 \models G_2$ if and only if there are homomorphisms from G_2 to $cl(G_1)$ and to $nf(G_1)$.*

Query Language. The definition of the graph query language SPARQL admits two types of query outputs: (i) sets of *triples* structured as RDF graphs, and (ii) sets of *tuples,* with all the tuples in each set organized in the same way, see Equation (11) for an illustration. The formal developments of [23, 24] focus on queries with outputs of type (i), modeled in those articles as queries whose heads are conjunctions of triples. In this article, we will focus on answering queries with outputs of type (ii), and will then comment on extensions to the query formalizations of type (i) of [23, 24].

Definition 2. A *conjunctive (CQ) query q over a temporal RDF graph schema $\mathcal{O}^T$* is a formula of the form $\exists \mathbf{y} \exists \mathbf{u}\ \psi(\mathbf{x}, \mathbf{t}, \mathbf{y}, \mathbf{u})$, where $\psi(\mathbf{x}, \mathbf{t}, \mathbf{y}, \mathbf{u})$ is a conjunction of atomic formulas over $\mathcal{O}^T$. We call $\exists \mathbf{y} \exists \mathbf{u}\ \psi(\mathbf{x}, \mathbf{t}, \mathbf{y}, \mathbf{u})$ the *body* of the query q, call $q(\mathbf{x}, \mathbf{t})$ its *head,* and use the notation

$$q(x_1, \ldots, x_k, t_1, \ldots, t_n) \leftarrow \exists y_1, \ldots, y_m, u_1, \ldots, u_r\ \psi(x_1, \ldots, x_k, t_1, \ldots, t_n, y_1, \ldots, y_m, u_1, \ldots, u_r).$$

Here, $k, n, m, r \geq 0$, and the variables in $\mathbf{x}$ (respectively in $\mathbf{t}$, $\mathbf{y}$, $\mathbf{u}$) take values in $U \cup L$ (respectively in T_I, $U \cup B \cup L$, T_I). The variables in $\mathbf{x}$ and $\mathbf{t}$ are the *free variables* of q; we call $\mathbf{xt}$ the *head vector of* q. A CQ query q is *p-ary* if its head vector has p variables.

In the sequel, we will use the common notation that omits quantifiers in query definitions.

We will also consider unions of CQ queries over temporal RDF graph schemas. A *union of CQ queries (UCQ query) over temporal RDF graph schema $\mathcal{O}^T$* is a finite set $Q = \{q_1, q_2, \ldots, q_r\}$ of CQ queries over $\mathcal{O}^T$, such that all the elements of Q have the same head vector. Note that each CQ query q over $\mathcal{O}^T$ can be viewed as a singleton-set UCQ $Q = \{q\}$ over the same $\mathcal{O}^T$.

The following CQ query over the temporal RDF graph schema of Example 2 asks for drugs and animals in farm 'F1' on which the drugs were used, as well as the time intervals of the applications.

$$\begin{aligned} q(x_1, x_2, t) \leftarrow & (x_1, \text{usedOn}, x_2) \wedge (x_2, \text{liveIn}, F1) \wedge (y_1, \text{subj}, x_1) \wedge (y_1, \text{pred}, \text{usedOn}) \\ & \wedge (y_1, \text{obj}, x_2) \wedge (y_1, \text{temporal}, y_2) \wedge (y_2, \text{interval}, y_3) \wedge (y_3, \text{validFor}, t). \end{aligned} \tag{11}$$

[10] Two graphs G and H are *isomorphic,* denoted $G \cong H$, if there exists a bijective homomorphism from G to H.

A CQ query q over a temporal RDF graph schema $\mathcal{O}^T = (\mathbf{C}, \mathbf{P}, G^T)$ is *well formed* if there exists a map μ that consistently replaces the variables $\mathbf{t}$ and $\mathbf{u}$ (respectively $\mathbf{x}$ and $\mathbf{y}$) in q with values in T_I (respectively in $U \cup B \cup L$), such that μ turns the body of q into a temporal RDF graph H such that $G^T \cup typed(H)$ is a well-formed RDF instance of $\mathcal{O}^T$. Here, $typed(H)$ is the result of typing all the triples in H w.r.t. $\mathcal{O}^T$.[11] For instance, the CQ query of Equation (11) is well formed. The definition extends naturally to the case of UCQ queries. In the remainder of this section, we assume that all the CQ and UCQ queries that we consider are well formed.

Let $q(\mathbf{x},\mathbf{t}) \leftarrow F$, with $F = \psi(\mathbf{x}, \mathbf{t}, \mathbf{y}, \mathbf{u})$, be a CQ query over a temporal RDF graph schema $\mathcal{O}^T$, J be an RDF instance of $\mathcal{O}^T$, and W be the set of all variables in q. A *valuation for q and J* is an RDF formula homomorphism h: $W \rightarrow universe(J)$ for F and J, such that $J \models q(h(\mathbf{x}), h(\mathbf{t}))$. The *normal-form- (nf-) answer $q(J)$ to CQ query q over $\mathcal{O}^T$ on RDF instance J of $\mathcal{O}^T$* is the set of tuples

$$q(J) = \{(h(\mathbf{x}), h(\mathbf{t})) : h \text{ is a valuation for } q \text{ and } nf(J)\}.$$

Further, for a UCQ query $Q = \{q_1, \ldots, q_r\}$ over $\mathcal{O}^T$, the *normal-form (nf-) answer $Q(J)$ to q over $\mathcal{O}^T$ on J of* is the set of tuples

$$Q(J) = \{(h(\mathbf{x}), h(\mathbf{t})) : h \text{ is a valuation for a } q_i \in Q \text{ and } nf(J), i = 1, \ldots, r\}.$$

Theorem 3. *The data complexity of obtaining nf-answers to a UCQ query over a temporal RDF graph schema on an RDF instance without blank nodes over the same schema is in* Ptime.

Proof. Let Q be a UCQ query over a temporal RDF graph schema, and J an RDF instance without blank nodes over the same schema. The result of the theorem follows from the observations that (i) the number of potential valuations for Q and $nf(J)$ is bounded by the number of subgraphs of $nf(J)$ of size of the bodies of the elements of Q, and (ii) $nf(J)$ can be computed in Ptime in the size of J due to J not having blank nodes. □

5.1.2 Certain Answers.

Definition 3. Let $\mathcal{M}^T = (\mathcal{S}^T, \mathcal{O}^T, \Sigma^T)$ be a temporal relational-to-RDF schema mapping, with I a source instance over $\mathcal{S}^T$, and let Q be a UCQ query over $\mathcal{O}^T$. The *certain answer of Q w.r.t. I*, denoted by $certain_{\mathcal{M}^T}(Q, I)$, is defined as

$$certain_{\mathcal{M}^T}(Q, I) = \bigcap_{J \text{ is a solution for } I \text{ w.r.t. } \mathcal{M}^T} Q(J).$$

That is, for a p-ary UCQ query Q over temporal RDF schema $\mathcal{O}^T$, $certain_{\mathcal{M}^T}(Q, I)$ consists of the set of all p-tuples r such that $r \in Q(J)$ for each solution J for I w.r.t. $\mathcal{M}^T$.

Lemma 2. *Let $\mathcal{O}^T$ be a temporal RDF graph schema, and J, J' be two RDF graph instances of $\mathcal{O}^T$, such that $universe(J) \cap B = \emptyset$. Then there exists an RDF instance homomorphism from J to J' iff there exists an RDF instance homomorphism from $nf(J)$ to $nf(J')$.*

Proof. For the only-if direction, consider any one RDF instance homomorphism h that is assumed to exist from J to J'. From $universe(J) \cap B = \emptyset$ we obtain that h is the identity mapping by definition of RDF instance homomorphisms. The proof proceeds by induction: We start from $J_0 = J$ and $J'_0 = J'$, and then ascertain in each ith step, $i > 0$, that applying to the *same* group of triples in both J_{i-1} and J'_{i-1} the *same* specific rule from the deductive system for RDF, see Equation (3)–(10), results in h still holding as identity mapping between the outcomes J_i and J'_i of the ith step. The

[11] For a triple $u = (s, p, o)$ in graph H and temporal RDF graph schema $\mathcal{O}^T = (\mathbf{C}, \mathbf{P}, G^T)$ s. t. $(p, domain, a), (p, range, b) \in nf(G^T)$ for $a, b \in \mathbf{C}$, the *typing of u w.r.t. $\mathcal{O}^T$* is the process of adding $(s, type, a)$ and $(o, type, b)$ to H.

above iterative process is clearly finite. Indeed, due to J not having blank nodes, applying the rules of Equation (3)–(10) to the triples of each J_{i-1} does not create blank nodes in J_i *or* J'_i.

As a result of the above finite iterative process, we obtain the RDF-closures $cl(J)$ of J and $cl(J')$ of J'. Now due to $cl(J)$ not having blank nodes, we have that $nf(J) = cl(J)$. Finally, observe that $h(nf(J)) \subseteq nf(J')$, due to $h(nf(J))$ belonging to the part of $cl(J')$ that does not contain blank nodes. We conclude that h is an RDF instance homomorphism from $nf(J)$ to $nf(J')$.

For the if-direction, we recall that $nf(J) = cl(J)$, thus from the existence of a homomorphism from $nf(J)$ to $nf(J')$ we obtain that there exists a homomorphism h from $cl(J)$ to $nf(J')$. Further, due to $nf(J') \subseteq cl(J')$ by definition of normal forms, it follows that h is also a homomorphism from $cl(J)$ to $cl(J')$. From this point we retrace the proof of the only-if direction in the reverse order, to conclude that h is also a homomorphism from J to J'. □

Canonical CQ Query. Given an RDF instance J of a temporal RDF graph schema $\mathcal{O}^T = (\mathbf{C}, \mathbf{P}, G^T)$, the *canonical CQ query of* J is a Boolean CQ query q_J over $\mathcal{O}^T$ whose body J' is constructed by removing from the graph $J\,|_{G^T} = J \setminus G^T$ all the triples whose predicate is `type`. For example, in case where J' consists of two triples (s, p, o), (s, p, i), we obtain $q_J() \leftarrow (s, p, o), (s, p, i)$.

Canonical RDF Instance. For a CQ query q over $\mathcal{O}^T = (\mathbf{C}, \mathbf{P}, G^T)$, the *canonical RDF instance of* q is the RDF instance $H_q = typed(G^T \cup H'_q)$ of $\mathcal{O}^T$, where the *query-instance graph* H'_q results from replacing each variable in the body of q with a distinct blank node not already used in q. It follows that for each CQ query q over a given well-formed $\mathcal{O}^T$, H_q is a well-formed RDF instance of $\mathcal{O}^T$.

Lemma 3. *Let $\mathcal{O}^T$ be a temporal RDF graph schema, and J, J' be two RDF graph instances of $\mathcal{O}^T$, such that $universe(J) \cap B = \emptyset$. Then the followings two claims are equivalent:*

— *$nf(J')$ satisfies the canonical CQ query*[12] *$q_{nf(J)}$ of $nf(J)$; and*
— *there exists an RDF instance homomorphism from $nf(J)$ to $nf(J')$.*

Proof. *Part 1:* Assuming that $nf(J')$ satisfies the query $q_{nf(J)}$, there must exist a homomorphism, h, from the canonical RDF instance $H_{q_{nf(J)}}$ of $q_{nf(J)}$ to $nf(J')$. Further, by definition of canonical RDF instance there must exist a homomorphism, g, from $nf(J)$ to $H_{q_{nf(J)}}$. (Recall that J, and thus also $nf(J)$, have no blank nodes.) Thus, $h \circ g$ is a homomorphism from $nf(J)$ to $nf(J')$.

Part 2: By definition, there exists a valuation g' for the canonical CQ query $q_{nf(J)}$ of $nf(J)$ and the graph $nf(J)$. We use the homomorphism h' that must exist from $nf(J)$ to $nf(J')$ to obtain the homomorphism $h' \circ g'$ from $q_{nf(J)}$ to $nf(J')$. We conclude that $nf(J')$ satisfies the query $q_{nf(J)}$. □

Theorem 4. *Let $\mathcal{M}^T = (\mathcal{S}^T, \mathcal{O}^T, \Sigma^T)$ be a temporal relational-to-RDF schema mapping, such that all the s-t tgds in Σ^T are full.*

(1) *Let Q be a UCQ query over $\mathcal{O}^T$. Then for every source instance I over $\mathcal{S}^T$ and universal solution J for I w.r.t. $\mathcal{M}^T$ it holds that $Q(J) = certain_{\mathcal{M}^T}(Q, I)$.*
(2) *Given a source instance I over $\mathcal{S}^T$, let J be a solution for I w.r.t. $\mathcal{M}^T$ such that (i) $universe(J) \cap B = \emptyset$, and (ii) for every UCQ query Q over $\mathcal{O}^T$ we have that $Q(J) = certain_{\mathcal{M}^T}(Q, I)$. Then J is a universal solution for I w.r.t. $\mathcal{M}^T$.*

Proof. We provide a proof for the case where Q is a singleton set $Q = \{q\}$; the extension to the general case of UCQ queries over temporal RDF graph schemas is straightforward.

(1) We prove first that $certain_{\mathcal{M}^T}(Q, I) \subseteq Q(J)$. Let $Q = \{q\}$ be a p-ary UCQ query, and consider a p-tuple r of values in $U \cup L \cup T_I$. By definition of certain answer, $r \in certain_{\mathcal{M}^T}(Q, I)$ implies $r \in Q(J)$, because J is a solution for I w.r.t. $\mathcal{M}^T$.

[12] An instance J *satisfies* a query q if $q(J) \neq \emptyset$.

Next, to prove $Q(J) \subseteq certain_{\mathcal{M}^T}(Q, I)$, we fix an arbitrary solution J' for I w.r.t. $\mathcal{M}^T$. Let h be a valuation for q and $nf(J)$, and g a homomorphism from J to J'. By Lemma 2, there exists a homomorphism g' from $nf(J)$ to $nf(J')$. We thus obtain a valuation $g' \circ h$ from q to $nf(J')$. Hence, for each p-tuple r of values in $U \cup L \cup T_I$ such that $r \in q(J)$, we have that $r \in q(J')$. (As all the elements of Σ^T are full s-t tgds, J cannot have blank nodes.) It follows that $Q(J) \subseteq certain_{\mathcal{M}^T}(Q, I)$.

(2) For the canonical CQ query $q_{nf(J)}$ of $nf(J)$, we are given that $q_{nf(J)}(J) = Q_{nf(J)}(J) = certain_{\mathcal{M}^T}(Q_{nf(J)}, I)$. In addition, by definition of $q_{nf(J)}$, we have that $q_{nf(J)}(J) = true$. Thus, for an arbitrary fixed solution J' for I w.r.t. $\mathcal{M}^T$, $q_{nf(J)}(J') = true$. Thus there must exist a valuation, h, for $q_{nf(J)}$ and $nf(J')$. As the canonical RDF instance of $q_{nf(J)}$ is $nf(J)$, h is also a homomorphism from $nf(J)$ to $nf(J')$; hence, by Lemma 2, there is a homomorphism from J to J'. As J' is an arbitrary solution for I w.r.t. $\mathcal{M}^T$, we conclude that J is a universal solution for I w.r.t. $\mathcal{M}^T$. □

5.2 Querying RDF Instances using Rewritings

In Section 5.1 we showed that UCQ queries over temporal RDF graph schemas can be correctly answered by reasoning on the universal solutions returned by relational-to-RDF temporal data exchange. In this subsection we discuss an alternative way to achieve the same query-answering results, which applies rewritings of the queries over the relevant temporal RDF graph schemas to the universal solutions. The results that we present in this section follow from the atemporal ones in [11]; see [26] for the background and further details connecting the results of [11] to SPARQL.

Let $Q = \{q_1, \ldots, q_r\}$ be a UCQ query over a temporal RDF schema $\mathcal{O}^T = \{\mathbf{C}, \mathbf{P}, G^T\}$. We can obtain from Q a UCQ query $R^{(Q)}$, by applying to the definition of each $q_i \in Q$ the inverses of the rules in the deductive system of Section 5.1, see Equation (3)–(10). The *inverse* of a rule is obtained by swapping the rule's antecedent and consequent; for instance, the inverse of the rule of Equation (8) is

$$\frac{(X, \text{type}, B)}{(A, \text{sc}, B)(X, \text{type}, A)}. \tag{12}$$

Consider, for instance, a UCQ query $Q = \{q\}$ over the graph schema of Example 2, which asks for drugs and animals on which the drugs were used, as well as the times of the applications.

$$\begin{aligned} q(x_1, x_2, t) \leftarrow\ & (x_1, \text{usedOn}, x_2) \wedge (y_1, \text{subj}, x_1) \wedge (y_1, \text{pred}, \text{usedOn}) \wedge (y_1, \text{obj}, x_2) \\ & \wedge (y_1, \text{temporal}, y_2) \wedge (y_2, \text{interval}, y_3) \wedge (y_3, \text{validFor}, t) \wedge (x_1, \text{type}, \text{AntimicrobialDrug}). \end{aligned} \tag{13}$$

Note the last conjunct of q, which types x_1 as "AntimicrobialDrug." The query Q returns the empty answer if posed directly on the RDF instance of Example 3, as the subject of the predicate `usedOn` is typed there as "AntibioticDrug." The correct answer $\{(Ampicillin, P1, [1/1/2019, 1/5/2019])\}$ to Q can be obtained by using the normal form of the RDF instance of Example 3, as discussed in Section 5.1. Alternatively, one could apply to q the rule of Equation (12), which would result in the query

$$\begin{aligned} q'(x_1, x_2, t) \leftarrow\ & (x_1, \text{usedOn}, x_2) \wedge (y_1, \text{subj}, x_1) \wedge (y_1, \text{pred}, \text{usedOn}) \wedge (y_1, \text{obj}, x_2) \\ & \wedge (y_1, \text{temporal}, y_2) \wedge (y_2, \text{interval}, y_3) \wedge (y_3, \text{validFor}, t) \\ & \wedge (z, \text{sc}, \text{AntimicrobialDrug}) \wedge (x_1, \text{type}, z). \end{aligned} \tag{14}$$

We can obtain the correct answer to Q by applying q' *directly* to the RDF instance of Example 3.

The query-rewriting process sketched above is well understood for UCQ queries over atemporal RDF graph schemas [11, 26]. We extend the approach to apply to temporal RDF graph schemas. The query-rewriting rules that we use, which are the inverses of those of Section 5.1, do not involve temporal triples. Thus, it is easy to see that, similarly to the atemporal case, the rewriting process terminates, and the resulting rewriting, which is a UCQ query over the input graph schema, is

unique, well defined, and is also well formed in case the input query is well formed. Accordingly, we define the rewriting-answer to a UCQ query Q with head vector $\mathbf{xt}$ over temporal RDF graph schema $\mathcal{O}^T$ on RDF instance J of $\mathcal{O}^T$ as follows: Let $R^{(Q)} = \{r_1^{(Q)}, \ldots, r_u^{(Q)}\}$ be the UCQ rewriting of Q over $\mathcal{O}^T$. Then the *rewriting- (rw-) answer $Q^{rw}(J)$ to Q on J* is the set of tuples

$$Q^{rw}(J) = \{(h(\mathbf{x}), h(\mathbf{t})) : h \text{ is a valuation for an } r_i^{(Q)} \in R^{(Q)} \text{ and } J, i = 1, \ldots, u\}.$$

As the rewriting rules do not involve temporal triples, the following is immediate from [11]:

Theorem 5. *For any UCQ query Q over $\mathcal{O}^T$ and RDF instance J of $\mathcal{O}^T$, we have $Q^{rw}(J) = Q(J)$.*

We now redefine the notion of certain answer using rw-answers to UCQ queries on RDF instances on temporal RDF graph schemas. The only change that needs to be made in Definition 3 for this purpose is to replace $Q(J)$ with $Q^{rw}(J)$ throughout. That is, for a p-ary UCQ query Q over temporal RDF schema $\mathcal{O}^T$, $certain^{rw}_{\mathcal{M}^T}(Q, I)$ consists of the set of all p-tuples u such that $u \in Q^{rw}(J)$ for each solution J for I w.r.t. $\mathcal{M}^T$. The following result is then immediate from Theorems 4 and 5.

Theorem 6. *Let $\mathcal{M}^T = (\mathcal{S}^T, \mathcal{O}^T, \Sigma^T)$ be a temporal relational-to-RDF schema mapping, such that all the s-t tgds in Σ^T are full.*

(1) *Let Q be a UCQ query over $\mathcal{O}^T$. Then for every source instance I over $\mathcal{S}^T$ and universal solution J for I w.r.t. $\mathcal{M}^T$ it holds that $Q(J) = certain^{rw}_{\mathcal{M}^T}(Q, I) = certain_{\mathcal{M}^T}(Q, I)$.*
(2) *Given a source instance I over $\mathcal{S}^T$, let J be a solution for I w.r.t. $\mathcal{M}^T$ such that (i) $universe(J) \cap B = \emptyset$, and (ii) for every UCQ query Q over $\mathcal{O}^T$ we have that $Q(J) = certain^{rw}_{\mathcal{M}^T}(Q, I)$. Then J is a universal solution for I w.r.t. $\mathcal{M}^T$.*

5.3 Temporal and Graph-generating Aspects of Queries over Temporal RDF Graphs

In this section we study, in the context of relational-to-RDF temporal data exchange, the issue of preservation of temporal information from the sources by query answers on the target instances. We then address the question of how the language of UCQ queries studied in Sections 5.1–5.2 relates to the query languages explored in [23, 24].

Temporal Annotations of Triples in Conjunctions, S-T Tgds, and Queries. Given an existential conjunctive formula $F = \exists \mathbf{y} \exists \mathbf{u}\ \psi(\mathbf{x}, \mathbf{t}, \mathbf{y}, \mathbf{u})$ over a temporal RDF graph schema $\mathcal{O}^T$, we say that F *provides a temporal annotation with temporal variable t_k to triple with predicate p, variables x_i and x_j, and functions f_1 and f_2*, if $(f_1(x_i), p, f_2(x_j))$ is a triple in F and there is a map, μ, that consistently replaces the variables $\mathbf{t}$ and $\mathbf{u}$ (respectively $\mathbf{x}$ and $\mathbf{y}$) in F with values in T_I (respectively in $U \cup B \cup L$), such that μ turns $\psi(\mathbf{x}, \mathbf{t}, \mathbf{y}, \mathbf{u})$ into a temporal RDF graph H in which the triple $(\mu(f_1(x_i)), p, \mu(f_2(x_j)))$ is temporally annotated with $\mu(t_k)$. Here, t_k is in $\mathbf{t}$, each of x_i and x_j is in $\mathbf{x}$, each of $\mathbf{y}$ and $\mathbf{u}$ can be the empty vector, and each of f_1 and f_2 is either F_{uri} or the identity function denoted by F_{id}.

Given a temporal s-t tgd σ from temporal relational schema $\mathcal{S}^T$ to temporal RDF graph schema $\mathcal{O}^T$ of the form $\forall \mathbf{x}, \mathbf{t}\ (\varphi(\mathbf{x}, \mathbf{t}) \wedge \pi(\mathbf{t}) \rightarrow \exists \mathbf{y}\ \psi(\mathbf{x}, \mathbf{y}, \mathbf{t}))$, we say that σ *imports from $\mathcal{S}^T$ a temporal annotation with t_k* (in $\mathbf{t}$) *into a triple with predicate p, variables x_i and x_j* (both in $\mathbf{x}$), *and functions f_1 and f_2*, if the RHS of σ provides a temporal annotation with t_k to triple with p, x_i, x_j, f_1, and f_2.

Given a CQ query q with head vector $\mathbf{xt}$ and body $F = \exists \mathbf{y} \exists \mathbf{u}\ \psi(\mathbf{x}, \mathbf{t}, \mathbf{y}, \mathbf{u})$ over graph schema $\mathcal{O}^T$, we say that q *returns a temporal t_k-annotation of its subgoal (x_i, p, x_j)* if F provides a temporal annotation with t_k to triple with p, x_i, x_j, and $f_1 = f_2 = F_{id}$. Here, x_j and x_j are in $\mathbf{x}$, and t_k is in $\mathbf{t}$.

Extended Projections on Relations. Given a p-ary relation R with domain $\textsc{Const} \cup U \cup L \cup T_I$, the *extended projection* $\Pi_{f_1(i_1), \ldots, f_k(i_k)}(R)$ for a $k \leq p$ is is obtained by (a) computing the

relational-algebra projection P of R on columns with indexes $i_1, \ldots, i_k$, and then (b) replacing each value, c, in the mth column of P with $f_m(c)$, $\forall\, m \in [1, k]$. Here, each of $f_1, \ldots, f_k$ is either F_{uri} or F_{id}.

The following result is immediate from the properties of chase with temporal relational-to-RDF schema mappings, see Sections 5.1–5.2. (An extension to UCQ queries is straightforward.)

THEOREM 7. *Let $\mathcal{M}^T = (\mathcal{S}^T, \mathcal{O}^T, \Sigma^T)$ be a temporal relational-to-RDF schema mapping, where all s-t tgds are full. Let σ be an s-t tgd in Σ^T of the form $\forall \mathbf{x}, \mathbf{t}(\varphi(\mathbf{x}, \mathbf{t}) \wedge \pi(\mathbf{t}) \rightarrow \psi(\mathbf{x}, \mathbf{t}))$, with $|\mathbf{x}| = n$, such that σ imports from $\mathcal{S}^T$ a temporal annotation with t_m (in $\mathbf{t}$) into a triple with predicate p, variables x_r and x_u (in $\mathbf{x}$), and functions f_1 and f_2. Let q be a CQ query with head vector $\mathbf{yu}$, with $|\mathbf{y}| = l$, over $\mathcal{O}^T$, that returns a temporal u_k-annotation of its subgoal (y_i, p, y_j). We define the relational CQ query $q'(\mathbf{x}, \mathbf{t}) \leftarrow \varphi(\mathbf{x}, \mathbf{t}) \wedge \pi(\mathbf{t})$ based on the LHS of σ. Then for each instance I of $\mathcal{S}^T$ we have that*

$$\Pi_{f_1(r), f_2(u), F_{id}(n+m)}(q'(I)) \subseteq \Pi_{F_{id}(i), F_{id}(j), F_{id}(l+k)}(certain_{\mathcal{M}^T}(q, I)).$$

That is, if an s-t tgd in a given temporal relational-to-RDF schema mapping imports temporal annotations from sources into its target triples, then queries that return the corresponding temporal annotations of their subgoals output all the temporal-annotation information from the given source when evaluated on any target instance. The s-t tgd of Equation (1), with triple $(F_{uri}(x_3), \text{usedOn}, F_{uri}(x_2))$, and the CQ query of Equation (11), with its subgoal $(x_1, \text{usedOn}, x_2)$, provide an illustration.

Query Answers as RDF Graphs. We now turn to the issue of how the language of UCQ queries studied in this article relates to the query languages explored in [23, 24]. The formalizations of [23, 24] admit conjunctions of triples, potentially with blank nodes, in query heads, instead of tuples as in this current article. We take the view, supported by the specification of SPARQL, that the outputs of queries whose heads are conjunctions of triples (we refer to them as queries of type (i)) can be obtained via a postprocessing step of forming RDF graphs out of values in output tuples of CQ queries formalized in Definition 2 (we call them queries of type (ii)), with the evaluation of the query body being the same for both types. Indeed, to form RDF graphs in answers to queries of type (i) it is sufficient to use the values returned in tuples, r, in answers to appropriate queries of type (ii), and to then form Skolem values based on the values in r for the blank nodes in the resulting RDF graphs, see [23]. Thus, the results of this article can be extended in a straightforward way to (U)CQ queries over temporal RDF graph schemas that return RDF graphs as answers.

5.4 Querying in Practice

We now introduce a user interface that makes it easier for domain experts to construct, rewrite, and obtain correct answers to temporal queries in real-world applications.

Consider a query that domain experts would like to be answered on an RDF instance J of a temporal RDF graph schema $\mathcal{O}^T$. Let J be the result of relational-to-RDF temporal data exchange from a temporal relational source, via a temporal relational-to-RDF schema mapping $\mathcal{M}^T$ with target schema $\mathcal{O}^T$, see Section 3.3. Here, $\mathcal{M}^T$ may have been provided directly by domain experts or, alternatively, obtained by the process of temporal enrichment of an atemporal relational-to-RDF schema mapping with atemporal RDF graph schema $\mathcal{O}^A$, as discussed in Section 4. We consider the scenario in which domain experts are not comfortable formulating queries directly over $\mathcal{O}^T$, due to not being familiar with the representation of temporal data and metadata that was introduced in Section 3. (This assumption is especially likely to hold in the case where the original atemporal domain ontology was temporally enriched via Algorithm 1 of Section 4.) Thus, while the query format of Section 5 admits a direct translation into SPARQL, we would be faced with domain experts not being expected to produce temporal queries over $\mathcal{O}^T$ that would align with that query format.

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

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

```
SELECT ?d ?f ?t
WHERE {
  ?d amr:usedOn ?a.                ?d rdf:type amr:AntibioticDrug.
  ?a amr:livesIn ?f.
  ?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.
  FILTER( initialDate(?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-referencing* 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. Unlike *(a)–(b)*, version *(c)* is directly executable by standard SPARQL processors.

In this scenario, our goal is to make it as easy as possible for domain experts to introduce temporal notation into SPARQL queries that they can already formulate over the atemporal RDF graph schema $\mathcal{O}^A$ that they are familiar with. (In the remainder of the article we consider UCQ queries with Allen interval relations, which is an extension of the query language considered in Section 5.) Consider a temporal query Q^T_{am}: "Return farms that used antimicrobial drugs in the year 2019," see Figure 3(c). Q^T_{am} can be processed directly by a standard SPARQL processor on any instance of the temporal RDF graph schema $\mathcal{O}^T$ of Example 2, with a nonempty answer successfully returned on the instance of Example 3. We introduce a *temporal user interface (temporal UI)* for SPARQL, to enable domain experts to concentrate on the domain-ontology part of formulating temporal queries such as Q^T_{am}. In the UI, standard SPARQL constructs are supplemented with *temporal references* on triple patterns, using the notation from [42], and with constructs for Allen interval relations. See Figure 3(a) for an illustration, in which ?t is a temporal reference. 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-referencing SPARQL queries.*

We now present a domain-independent approach for reformulating temporally-referencing SPARQL queries into (standard) SPARQL queries that respect the structure of the temporal information in the given temporal RDF graph schema while preserving the semantics of the questions. Acting on top of a SPARQL processor, our Algorithm 2 ensures successful evaluation of temporally-referencing SPARQL queries on instances of the given temporal RDF graph schema. The algorithm accepts as inputs an RDF instance J of a temporal RDF graph schema $\mathcal{O}^T$ and a temporally-referencing SPARQL query 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 references and comparisons in our temporal UI.) The algorithm reformulates each given Q into a set $\mathcal{R}$ of standard SPARQL queries conforming to $\mathcal{O}^T$, and then uses the SPARQL processor to obtain the answer to Q, by processing all the queries in $\mathcal{R}$ on the input RDF instance J.

The reformulation part of Algorithm 2 works in two stages, rewriting (1st stage) and expansion (2nd stage). In the 1st stage, Algorithm 2 uses domain-independent pattern-based rules to repeatedly "unfold," in the queries being rewritten, `:subClassOf` and `:subPropertyOf` hierarchies w.r.t. $\mathcal{O}^T$, using the approach of Section 5.2. As a result, the input query Q is turned into a set $\mathcal{R}$ of SPARQL queries that would be directly executable on the input instance J *but for* their time references and comparisons. This process would transform the query of Figure 3(a) into the query of Figure 3(b).

The 2nd, expansion, stage of the query-reformulation process in Algorithm 2 uses domain-independent pattern-based rules to replace the temporal terminology in the queries $\mathcal{R}$ with

ALGORITHM 2: Temporal querying over temporal RDF graph schemas and instances

Data: RDF instance J of temporal RDF graph schema $\mathcal{O}^T$, temporally-referencing SPARQL query Q.
Result: Answer set $\mathcal{A}$ to a SPARQL reformulation of Q on J.
begin
  $\mathcal{R} \leftarrow \{Q\}$; `// initializing the reformulation` $\mathcal{R}$ `of` Q `that will end up being executable on` J
  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` $\mathcal{R}$ `with the approach of Section 5.2`
  for *each temporal reference or Allen interval relation* T *in* $\mathcal{R}$ **do**
    $\mathcal{R} \leftarrow$ expand T in $\mathcal{R}$ into temporal annotation or interval comparison; `// 2nd stage: temporal expansion of` $\mathcal{R}$
  $\mathcal{A} \leftarrow \emptyset$; `// initializing set of answers to` $\mathcal{R}$ `on RDF instance` J
  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 J; `// 3rd stage: evaluation of` $\mathcal{R}$ `on` J
  return $\mathcal{A}$;

standard RDF/S constructs. Specifically, all the temporal references in individual triple patterns in $\mathcal{R}$ are replaced with their structural counterparts of Section 3, and 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 Figure 3(b) into the query of Figure 3(c).) The resulting queries, which are SPARQL queries without any nonstandard temporal terminology, are submitted by the algorithm to the SPARQL processor to obtain the answers to the input query.

6 IMPLEMENTATION AND EXPERIMENTAL RESULTS

6.1 Implementation and Experimental Setup

We have implemented Algorithms 1–2 on top of Java 1.8, using the Llunatic [20] rule interpreter for rewriting and expanding SPARQL queries. The source relations were stored using PostgreSQL 11; target RDF triples were stored and manipulated with RDF4J 3.0.1. The experiments were conducted in the environment of Ubuntu 18.04 Bionic with Core i7 3.20 GHz, 16 GB RAM, and 2 TB HDD.

For the experiments, we used data environments in two application domains. Each environment included a relational source schema, an atemporal target RDFS domain ontology, and a set of GLAV s-t tgds each with at most one temporal variable, which, if present, would occur exactly once on the left-hand side. Each data environment also included relational source data generated with DataFiller [14] 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 6.3. The first data environment captures a real-life AMR scenario obtained from our collaboration [27] 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 other tables would contain information about the farms, their locations, and relevant weather events. We also used the RDFS version of the atemporal ARO ontology for AMR,

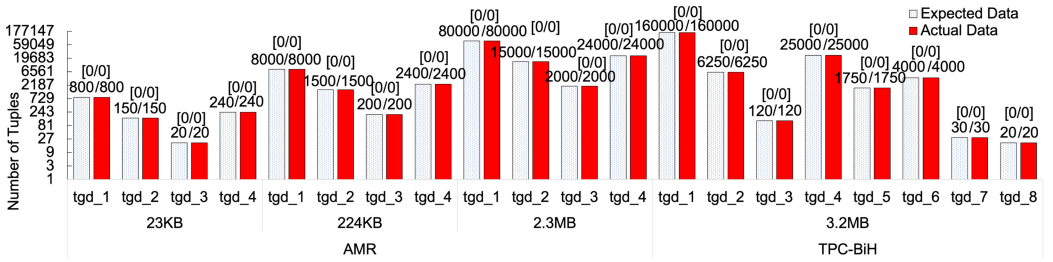

Fig. 4. Evaluating information loss in data exchange with temporal RDF enrichment, vs. expected (relational-to-relational) outcomes. The X-axis shows the s-t tgds and the source sizes for the environments tested; the (logarithmic) Y-axis shows the number of resulting data tuples. The $[A/B]$ notation on top of the bars shows the relative number of unmatched tuples between the two sets, see Section 6.3. Specifically, the meaning of $[0/0]$ is that the two data sets are identical.

as well as s-t tgds developed in our collaboration with AMR researchers. The second data environment was based on the TPC-BiH benchmark [17, 28, 29], which is used for evaluating the performance of temporal databases. The TPC-BiH schema is based on eight TPC-H [38] relations, which collectively reflect realistic information in a business-application scenario. 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 an atemporal (cf. TPC-H) ontology and generated the associated s-t tgds using an approach similar to those of [12, 39].

6.2 Experimental Methodology and Expected Outcomes

The experiments, whose results are reported in Section 6.3, were designed around specific properties of the outcomes of applying Algorithms 1–2 for temporal RDF target enrichment and querying to the AMR and TPC-BiH data environments. We evaluated the following properties of the outcomes:

(a) degree of preservation in the target of the source temporal information, see Figure 4; and
(b) 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.[13]

We evaluated property (b) both for queries that required rewriting w.r.t. the `:subClassOf` and `:subPropertyOf` hierarchies in the given atemporal ontologies and for queries that did not require such rewriting, see Figure 5. Finally, we evaluated the efficiency of our implementation, see Figure 6.

Our methodology for evaluating the above properties was as follows. Observe that Algorithms 1–2 would be provably sound if it applied to the relational-to-*relational* scenario. That is, if the target data were relational, then, under our assumption of each s-t tgd being GLAV with at most one temporal variable, correctness of temporal RDF target enrichment and of temporal querying on the materialized target data would follow from the formal results of [21]. 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 [21], that experiment would have provably correct outputs, which we would then adopt as our *expected outputs* for *Experiment (II)*; and

[13]Recall that, in contrast with Section 5, Algorithm 2 covers queries with Allen interval relations.

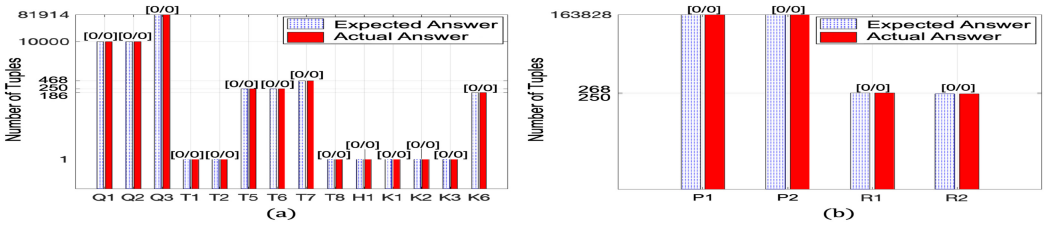

Fig. 5. Evaluating information loss in answers to temporal queries, vs. (relational) benchmark outcomes, for (a) queries not requiring rewriting w.r.t. RDFS hierarchies, and for (b) queries requiring such rewriting. The X-axes show the queries tested. The (logarithmic) Y-axes show query-answer sizes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

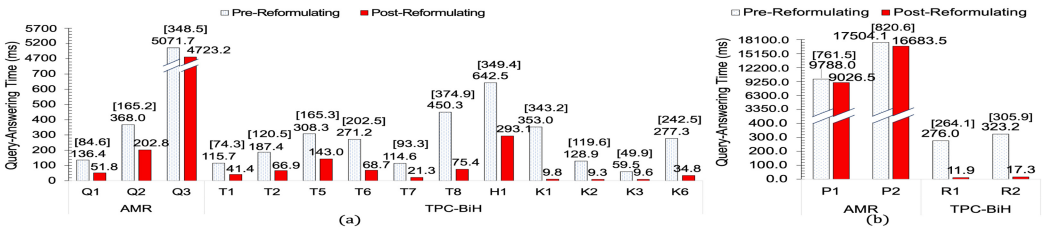

Fig. 6. Measuring the time overhead of reformulating temporally-referencing queries in query processing, for queries not requiring (a) vs requiring (b) rewriting w.r.t. RDFS hierarchies. The X-axes show the queries tested. The (logarithmic) Y-axes show the response times in *ms;* each bar height is the average of 10 runtimes. The values in square brackets show the difference between the processing times with the reformulation overhead included (left bar) and excluded (right bar).

(2) *Experiment (II)* would be conducted in the relational-to-*RDF* scenario using the original Algorithms 1–2. We would then translate the RDF outputs into relations in a straightforward manner, and compare the results with the outputs of *Experiment (I)* for the same inputs.

For the cases where the results of *Experiments (I)* and *(II)* were identical for a given input, we would count the cases as corroborating the experimental validation of correctness of Algorithms 1–2.

6.3 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 6.2, were identical. We conclude that all these results experimentally validate the correctness of Algorithms 1–2.

The data environments used to obtain the results in Figures 4–6 are as follows: The AMR environment with the source comprising 100 / 1,000 / 10,000 tuples, at the respective storage sizes of 23 KB through 2.3 MB, and the TPC-BiH environment with the source comprising 10,000 tuples, at storage size of 3.2 MB. (We are not reporting the results for additional values of the scale factors that were tested for both environments, due to all such results being in line with the results reported in Figures 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. Figure 4 shows the sizes of the target data, both relational *(Experiments (I))* and RDF (*Experiments (II)*) 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* setting.[14] 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 Algorithm 1.

Next, we discuss our results, in the AMR and TPC-BiH environments, for the degree of correctness of the answers to temporal queries on the RDF target *(Experiment (II)),* w.r.t. the "benchmark" relational answers obtained via the respective *Experiment (I),* see Figure 5. All the input queries were temporally-referencing SPARQL queries of the form illustrated in Figure 3(a), which were then subjected to reformulation via Algorithm 2; the outputs were SPARQL queries of the form illustrated in Figure 3(c). We used the certain-answer semantics [5, 18] in processing all the queries. Among the queries used in the experiments, the meaning of, e.g., the AMR query $Q2$ in Figure 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 [28].

None of the queries in Figure 5(a) required rewriting w.r.t. RDFS hierarchies using the 1st stage of Algorithm 2. As a result, devising their semantically equivalent SQL counterparts for the respective *Experiments (I)* was straightforward. The experiments whose results are shown in Figure 5(b) were set up differently, as their temporally-referencing input queries did require rewriting w.r.t. the RDFS `:subClassOf` and `:subPropertyOf` hierarchies. Among the queries used in the experiments, the meaning of, e.g., the query $R2$ in the TPC-BiH environment (Figure 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-referencing SPARQL queries in Figure 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.[15]

To summarize the results shown in Figure 5, 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 Figure 5 is the same as in Figure 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 target *experimentally validate* the correctness of our Algorithm 2.

Finally, Figure 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 Figure 6 show the overall response times for the temporally-referencing 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 Figure 6(a), and for queries requiring such rewriting, see Figure 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

[14]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).*

[15]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.

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 [32], 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 [20], the values of the Algorithm 2 reformulation overhead dropped to tens of *ms* vs. the time-difference values shown in Figure 6. **We conclude** that the *runtime overhead of using Algorithm 2 in the reformulation of temporally-referencing 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.

REFERENCES

[1] Dean Allemang and James Hendler. 2011. *Semantic Web for the Working Ontologist: Effective Modeling in RDFS and OWL* (2nd. ed.). Morgan Kaufmann.

[2] James F. Allen. 1983. Maintaining knowledge about temporal intervals. *Commun. ACM* 26, 11 (1983), 832–843. https://doi.org/10.1145/182.358434

[3] Grigoris Antoniou, Paul T. Groth, Frank van Harmelen, and Rinke Hoekstra. 2012. *A Semantic Web Primer,*. (3rd. ed.). MIT Press.

[4] Jing Ao, Zehui Cheng, Rada Chirkova, and Phokion G. Kolaitis. 2020. Temporal enrichment and querying of ontology-compliant data. In *Proceedings of the ADBIS 2020 Short Papers*. 129–139.

[5] M. Arenas, P. Barceló, L. Libkin, and F. Murlak. 2014. *Foundations of Data Exchange*. Cambridge University Press.

[6] A. Artale, R. Kontchakov, A. Kovtunova, V. Ryzhikov, F. Wolter, and M. Zakharyaschev. 2017. Ontology-mediated query answering over temporal data: A survey. In *Proceedings of the TIME 2017, October 16–18, 2017, Mons, Belgium*. 1:1–1:37.

[7] Iovka Boneva, Jérémie Dusart, Daniel Fernández-Álvarez, and José Emilio Labra Gayo. 2019. Shape designer for ShEx and SHACL constraints. In *Proceedings of the ISWC Satellite Tracks*. 269–272.

[8] Iovka Boneva, Jose Lozano, and Sławek Staworko. 2018. Relational to RDF data exchange in presence of a shape expression schema. In *Proceedings of the 12th Alberto Mendelzon International Workshop on Foundations of Data Management*, Dan Olteanu and Barbara Poblete (Eds.). Vol. 2100. https://ceur-ws.org/Vol-2100/paper6.pdf.

[9] Iovka Boneva, Slawek Staworko, and Jose Lozano. 2020. Consistency and certain answers in relational to RDF data exchange with shape constraints. In *Proceedings of the ADBIS 2020 Short Papers*. 97–107.

[10] Dan Brickley and Ramanathan Guha. 2014. *RDF Schema 1.1*. W3C Recommendation. W3C. Retrieved 25 February 2014 from https://www.w3.org/TR/2014/REC-rdf-schema-20140225/.

[11] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. 2007. Tractable reasoning and efficient query answering in description logics: The *DL-Lite* family. *J. Autom. Reason.* 39, 3 (2007), 385–429.

[12] Yijian Cheng, Pengjie Ding, Tongtong Wang, Wei Lu, and Xiaoyong Du. 2019. Which category is better: benchmarking relational and graph database management systems. *Data Sci. Eng.* 4, 4 (2019), 309–322. https://doi.org/10.1007/s41019-019-00110-3

[13] Zehui Cheng and Phokion G. Kolaitis. 2020. Universal solutions in temporal data exchange. In *Proceedings of the TIME 2020*. Emilio Muñoz-Velasco, Ana Ozaki, and Martin Theobald (Eds.). 8:1–8:17.

[14] F. Coelho. 2014. DataFiller – Generate Random Data from Database Schema. Retrieved 23 March 2014 from https://www.cri.ensmp.fr/people/coelho/datafiller.html.

[15] combating 2017. *Combating Antimicrobial Resistance: A One Health Approach to a Global Threat: Proc. National Academies of Sciences Workshop*. National Academies Press.

[16] Richard Cyganiak. 2005. A relational algebra for SPARQL. *Digital Media Systems Laboratory HP Laboratories Bristol. HPL-2005-170* 35, 9 (2005).

[17] Anton Dignös, Boris Glavic, Xing Niu, Michael Böhlen, and Johann Gamper. 2019. Snapshot semantics for temporal multiset relations. *Proc. VLDB Endow.* 12, 6 (2019), 639–652.

[18] Ronald Fagin, Phokion G. Kolaitis, Renée J. Miller, and Lucian Popa. 2005. Data exchange: Semantics and query answering. *Theoretical Computer Science* 336, 1 (2005), 89–124.

[19] Enrico Franconi, Jos de Bruijn, and Sergio Tessaris. 2005. Logical reconstruction of normative RDF. In *Proceedings of the OWLED*05 Workshop on OWL*. Bernardo Cuenca Grau, Ian Horrocks, Bijan Parsia, and Peter F. Patel-Schneider (Eds.).

[20] Floris Geerts, Giansalvatore Mecca, Paolo Papotti, and Donatello Santoro. 2020. Cleaning data with Llunatic. *VLDBJ* 29, 4 (2020), 867–892. https://doi.org/10.1007/s00778-019-00586-5

[21] Ladan Golshanara and Jan Chomicki. 2020. Temporal data exchange. *Inf. Syst.* 87 (2020), 1–13. https://doi.org/10.1016/j.is.2019.07.004

[22] Ramanathan Guha and Dan Brickley. 2004. *RDF Vocabulary Description Language 1.0: RDF Schema*. W3C Recommendation. W3C. Retrieved 10 February 2004 from https://www.w3.org/TR/2004/REC-rdf-schema-20040210/.
[23] Claudio Gutiérrez, Carlos A. Hurtado, Alberto O. Mendelzon, and Jorge Pérez. 2011. Foundations of semantic web databases. *Journal of Computer and System Sciences* 77, 3 (2011), 520–541. DOI: https://doi.org/10.1016/j.jcss.2010.04.009
[24] Claudio Gutiérrez, Carlos A. Hurtado, and Alejandro A. Vaisman. 2007. Introducing time into RDF. *IEEE Transactions on Knowledge and Data Engineering* 19, 2 (2007), 207–218. DOI: https://doi.org/10.1109/TKDE.2007.34
[25] P. Hayes. 2004. *RDF Semantics*. W3C Recommendation. W3C. Retrieved 10 February 2004 from www.w3.org/TR/2004/REC-rdf-mt-20040210/.
[26] Aidan Hogan. 2020. *The Web of Data*. Springer.
[27] P.-Y. Hou, J. Ao, A. Rindos, S. Keelara, P. Fedorka-Cray, and R. Chirkova. 2019. Collaborative workflow for analyzing large-scale data for antimicrobial resistance: An experience report. In *Proceedings of the IEEE BigData*.
[28] M. Kaufmann, P. M. Fischer, N. May, A. Tonder, and D. Kossmann. 2014. TPC-BiH: A benchmark for bitemporal databases. In *Proceedings of the Performance Characterization and Benchmarking*. R. Nambiar and M. Poess (Eds.).
[29] Wei Lu, Zhanhao Zhao, Xiaoyu Wang, Haixiang Li, Zhenmiao Zhang, Zhiyu Shui, Sheng Ye, Anqun Pan, and Xiaoyong Du. 2019. A Lightweight and Efficient Temporal Database Management System in TDSQL. *Proc. VLDB Endow.* 12, 12 (2019), 2035–2046. http://www.vldb.org/pvldb/vol12/p2035-lu.pdf.
[30] Franck Michel, Johan Montagnat, and Catherine Faron Zucker. 2014. A Survey of RDB to RDF Translation Approaches and Tools. Rapport de Recherche ISRN I3S/RR 2013-04-FR.
[31] Sergio Muñoz, Jorge Pérez, and Claudio Gutiérrez. 2007. Minimal deductive systems for RDF. In *Proceedings of the ESWC*.
[32] Jakob Nielsen. 1993. *Usability Engineering*. Morgan Kaufmann.
[33] M. Tamer Özsu. 2016. A survey of RDF data management systems. *Frontiers of Computer Science* 10, 3 (2016), 418–432.
[34] rdfsemw3c 2004. RDF Semantics: W3C Recommendation 10 February 2004. Hayes, Patrick (Ed.) Retrieved 10 February 2004 from https://www.w3.org/TR/2004/REC-rdf-mt-20040210/.
[35] rdfsw3c 2014. RDF Vocabulary Description Language: RDF Schema. Brickley, D. and Guha, R.V. (Eds.) Retrieved 25 February 2014 from https://www.w3.org/TR/rdf-schema/.
[36] sparql 2008. SPARQL Query Language for RDF. W3C 15/01/2008. Retrieved from https://www.w3.org/TR/rdf-sparql-query/.
[37] Jonas Tappolet and Abraham Bernstein. 2009. Applied temporal RDF: Efficient temporal querying of RDF data with SPARQL. In *Proceedings of the ESWC*. 308–322.
[38] TPC 2018. TPC BENCHMARK H Rev. 2.18.0. Retrieved from www.tpc.org/tpc_documents_current_versions/pdf/tpc-h_v2.18.0.pdf.
[39] tpch. 2023. Working with the TPCH Data. https://docs.cambridgesemantics.com/anzograph/archive/v2.3/userdoc/ghib.htm.
[40] W3C. 2012. A Direct Mapping of Relational Data to RDF. Retrieved 27 September 2012 from https://www.w3.org/TR/rdb-direct-mapping/.
[41] W3C. 2012. R2RML: RDB to RDF Mapping Language. Retrieved 27 September 2012 from http://www.w3.org/TR/r2rml/.
[42] Antoine Zimmermann, Nuno Lopes, Axel Polleres, and Umberto Straccia. 2012. A general framework for representing, reasoning and querying with annotated Semantic Web data. *Journal of Web Semantics* 11 (2012), 72–95. https://doi.org/10.1016/j.websem.2011.08.006

Received 24 May 2022; revised 2 December 2022; accepted 19 January 2023