Temporal Query Answering in a Fuzzy World*

Veronika Thost
Institute for Theoretical Computer Science
Technische Universität Dresden
Germany
veronika.thost@tu-dresden.de

Erik Zenker
Institute for Theoretical Computer Science
Technische Universität Dresden
Germany
erik.zenker@tu-dresden.de

ABSTRACT

Ontologies play a central role in semantic applications: by providing semantics to the given data, they support the integration and automated processing of knowledge. Systems for ontology-based data access do however not take into account both the fuzzy and the temporal nature of the knowledge, which is often inherent in real-world data. In this paper, we propose an approach for temporal query answering over fuzzy data w.r.t. ontologies.

Categories and Subject Descriptors

H.3 [**Information Storage and Retrieval**]: Miscellaneous; I.2.4 [**Knowledge Representation Formalisms and Methods**]: Representation languages,Temporal logic

General Terms

Algorithms, Languages

Keywords

Ontologies, Query Answering, Description Logics

1. INTRODUCTION

Ontologies play a central role in semantic applications: by providing semantics to the given data, they support the integration and automated processing of knowledge. Well-known medical domain ontologies like SnomedCT[1] and GALEN[2] may, for example, capture the fact that there are several synonyms for the varicella zoster virus (VZV) (e.g., HHV3) and that someone having chickenpox is infected with VZV. This supports the integration of patient data coming from several sources (e.g., if they address VZV differently) and physicians querying patient records (e.g., if they ask if a patient had been infected with VZV and the record states that he has had chickenpox at some time). Hence, ontology-based query answering (OBQA) may assist in finding appropriate participants for a clinical study: by formulating the *eligibility criteria* (i.e., the requirements to be met by the participants) as queries over the (probably linked) patient data, appropriate candidates can be found. The following are two such criteria, for example:[3]

- Either previously infected with VZV or previously vaccinated with VZV vaccine
- Karnofsky score of greater than or equal to 50

Note that the first criterion addresses data of the past, and that the Karnofsky score[4] captures the well-being of a patient and is thus a rather vague measure with smooth transitions (e.g., 40 denotes 'requires special assistance ...', and 50 stands for 'requires considerable assistance ...'). A corresponding database storing the patient data (i.e., their current Karnofsky score and past findings) in two tables could look as follows.

PID	Name	Karnofsky
1	Ann	80
2	Bob	70
3	Chris	48
4	Dan	46
5	Eva	70

PID	Finding	Date
1	HHV3-Infection	13.08.2007
2	Chickenpox	22.01.2010
3	VZV-Infection	01.11.2011
4	VZV-Infection	13.06.2004
5	VZV-Infection	05.02.2012

Regarding the above data, OBQA could enhance standard database query answering (e.g., in SQL) in that not only Eva, but also Ann and Bob would be considered as appropriate candidates. However, only few state of the art systems for OBQA support temporal queries—to the best of our knowledge, it is only [8]. Furthermore, we sometimes might want to relax the query and also consider Chris and Dan as being eligible, whose score is slightly below 50.

In this paper, we therefore do not only apply ontologies to access data, but explicitly integrate temporal and fuzzy aspects into query answering. Specifically, we use *fuzzy classes* to model vague information as the Karnofsky score. That is, instead of considering *individuals* to be either an instance of a class or not, we specify a *degree* of instantiation, from the interval [0,1] (e.g., the scores of Chris and Dan could, respectively, be considered to be instances of a class EligibleKarnofskyScore with degrees 0.8 and 0.6, and degree 1 could be used for scores $\geq$ 50). Then, a given query can be extended with a variable degree to obtain different sets of answers (e.g., querying the example data for patients whose score instantiates the class EligibleKarnofskyScore with degrees

*Partially supported by the DFG in CRC 912 (HAEC).

[1] http://www.ihtsdo.org/snomed-ct

[2] http://www.co-ode.org/ontologies/galen

[3] https://clinicaltrials.gov/ct2/show/NCT01953900

[4] http://en.wikipedia.org/wiki/Performance_status

$EquivalentClasses(\mathsf{VZV}, \mathsf{HHV3})$,	$ObjectPropertyDomain(\mathsf{hasFinding}, \mathsf{Patient})$,
$SubClassOf(\mathsf{VZV}, \mathsf{Virus})$,	$ObjectPropertyRange(\mathsf{hasFinding}, \mathsf{Finding})$,
$SubClassOf(\mathsf{Chickenpox}, \mathsf{VZVInfection})$,	$ObjectPropertyDomain(\mathsf{hasKarnofskyScore}, \mathsf{Patient})$,
$SubClassOf(\mathsf{VZVInfection}, \mathsf{Finding})$,	$ObjectPropertyRange(\mathsf{hasKarnofskyScore}, \mathsf{KarnofskyScore})$,
$SubClassOf(\mathsf{EligibleKarnofskyScore}, \mathsf{KarnofskyScore})$,	...

Figure 1: OWL 2 QL-axioms that capture terminological knowledge of the running example.

≥ 0.8 and ≥ 0.5 would result in answers $\{Chris, Eva\}$ and $\{Chris, Dan, Eva\}$, respectively). Next to the possibility to specify degrees, we use a recently proposed temporal query language that applies the operators of the well-known propositional temporal logic LTL (e.g., to express the 'previously' in the first criterion). We hence integrate two extensions of classical ontology-based query answering, motivated by the often temporal and/or fuzzy nature of real-world data. We also propose an algorithm to answer these queries and describe a prototypical implementation.

2. KNOWLEDGE REPRESENTATION AND THE QUERIES

We start specifying the ontologies, the data model, and the queries we focus on. In particular, we regard ontologies written in the ontology language *DL-Lite*$_\mathcal{R}$, a subset of the OWL 2 QL profile [1]. This language allows for capturing the conceptual features of relational databases and has been tailored for efficient query answering. The terminological information mentioned in the introduction can be, for example, represented by the axioms given in Figure 1.

To make the ontology applicable, the given data has to be linked to the classes and properties of the ontology. This is commonly done via mappings [13], which point out the data representing *individuals* (e.g., such a mapping may describe that every $PID\ pid$ in the example database represents an individual $:pdb/patient/pid$, and that every Karnofsky score associated with a specific $PID\ pid$ and date $date$ represents a unique individual $:pdb/karnofsky/pid/date$, too) and specify the instantiation of classes and properties (e.g., that an individual $:pdb/karnofsky/1/13.08.2007$ is an instance of class EligibleKarnofskyScore if the corresponding score is ≥ 50). Since we want to consider some data to be fuzzy, and thus to be explicitly annotated with a degree, we extend this usual mapping. In particular, we apply a function $fuzzify_S$ which, for each class (property) S defines how the degree—to which the (tuple of) individual(s) instantiates the class (property)—is obtained from the database content (e.g., $fuzzify_{\mathsf{EligibleKarnofskyScore}}$ may map a given Karnofsky score v to degree 1 if $v \geq 50$, to 0 if $v \leq 40$, and to $(v-40)/10$, otherwise). Note that, alternatively, the class EligibleKarnofskyScore could be populated differently, in different scenarios, by using the common approach and adapting the mapping linking the data to the ontology. However, then, the degree would not be explicitly represented and thus could not be addressed in the queries, which are described later in this section.

We specifically integrate the temporal nature of the data by populating the property hasTimeStamp for all data of temporal context (e.g., assuming the current date $'01.06.2015'$, we would define the mapping such that the tuple $(:pdb/karnofsky/1/01.06.2015, '01.06.2015')$ instantiates the property to degree 1).

As query language, we basically apply the language TSPARQL we proposed in [21]; it is described in Figure 2 and basically extends SPARQL by the Boolean and temporal operators of LTL. The eligibility criteria from the example could be formulated in TSPARQL as the below query, asking for all patients that meet the criteria:[5][6]

$q := q_1$ AND $q_2 \mid q_1$ OR $q_2 \mid$
PREVIOUS $q_1 \mid$ NEXT $q_1 \mid$
ALWAYS_IN_PAST $q_1 \mid$ ALWAYS $q_1 \mid$
SOMETIMES_IN_PAST $q_1 \mid$ SOMETIMES $q_1 \mid$
q_1 SINCE $q_2 \mid q_1$ UNTIL q_2

Figure 2: The syntax of our TSPARQL queries, which are built from conjunctive SPARQL queries q_1 and q_2 using the Boolean and temporal LTL operators.

```
((SOMETIMES_IN_PAST
  SELECT ?p WHERE {
  ?p a :Patient.
  ?p :hasFinding ?f.
  ?f a :VZVInfection})
 OR
(SOMETIMES_IN_PAST
  SELECT ?p WHERE {
  ?p a :Patient.
  ?p :isVaccinatedWith ?v.
  ?v a :VZVVaccine}))
AND
SELECT ?p WHERE {
 ?p a :Patient.
 ?p :hasKarnofskyScore ?s.
 ?s a :EligibleKarnofskyScore}
```

Additionally, we assume a degree to be given with every TSPARQL query to be answered—to finally restrict the set of answers. In the next section, we propose an algorithm to answer such queries.

3. THE ALGORITHM

Our algorithm is a so-called *rewriting approach*. In general, such an approach rewrites a given query, written in the abstract vocabulary of an ontology, into a standard database query (e.g., in SQL) that encodes the relevant ontological knowledge but addresses a general database; the latter can then be used to store the data and efficiently answer the (rewritten) queries.[7]

Algorithm 1 gives an overview of the rewriting algorithm sql. Its input consists of a temporal query ϕ^t in TSPARQL and an on-

[5]Note that the 'previously', which occurs in the eligibility criterion of the example, does not only refer to the time point directly preceding the current moment, but to the past in general. For that reason, we use the LTL operator SOMETIMES_IN_PAST instead of the PREVIOUS operator.

[6]Please note that negation is not considered in TSPARQL.

[7]An introduction of the approach is given in [6], for example.

tology $\mathcal{O}$, and the output is a basic SQL query. The algorithm processes the temporal subqueries in ϕ^t recursively until no temporal operators are present any more; that is, until the considered subquery is a plain conjunctive query (CQ) in SPARQL. The then considered CQ ψ is first *extended* with the terminological knowledge contained in the ontology (e.g., if ψ asks for findings that are instances of the class VZVInfection, it is extended such that it also asks for all instances of class Chickenpox) and afterwards *fuzzified* to add the membership degrees (e.g., if ψ asks for instances of EligibleKarnofskyScore, then the above described function $fuzzify_{\text{EligibleKarnofskyScore}}$ is included into the SQL to generate the corresponding membership degree from the data). Subsequently, the extended and fuzzified CQs are recombined by the given temporal operators.

Algorithm 1 TSPARQL to SQL rewriting

function SQL($\phi^t, \mathcal{T}$)
  $\phi' \leftarrow \emptyset$
  ▷ Retrieve list of temporal operators
  $t' \leftarrow operators(\phi^t)$
  for all subqueries ψ in TCQ ϕ^t **do**
    if ψ contains temporal operator **then**
      ▷ Recursively resolve temporal operator
      $\phi' \leftarrow append(\phi',\ sql(\psi, \mathcal{T}))$
    else
      ▷ Extend ψ by information of $\mathcal{T}$
      $\psi_\mathcal{T} \leftarrow extend(\psi, \mathcal{T})$
      ▷ Annotate $\psi_\mathcal{T}$ with fuzzy information
      $\psi^f_\mathcal{T} \leftarrow fuzzify(\psi_\mathcal{T})$
      ▷ Append the fuzzyfied CQ to ϕ'
      $\phi' \leftarrow append(\phi',\ \psi^f_\mathcal{T}))$
    end if
  end for
  ▷ Connects a list of queries by temporal operators
  $\phi^{t,f}_\mathcal{T} \leftarrow temporalize(\phi', t')$
  return $\phi^{t,f}_\mathcal{T}$
end function

For the running example, an abstract representation of the query could look as below.

$$\phi^t = ((\text{SOMETIMES\_IN\_PAST } \psi_1) \text{ OR } (\text{SOMETIMES\_IN\_PAST } \psi_2)) \text{ AND } \psi_3$$

Recall that the function sql separates the temporal operators and thus splits the given query. As an example, we regard the rewriting $sql(\text{SOMETIMES\_IN\_PAST } \psi_1)$ (i.e., ψ_1 denotes the SPARQL query asking for a patient that has a finding which is an instance of class VZVInfection). Since ψ_1 is a plain CQ, it is extended and fuzzified as described above. The SQL statement in Listing 1 shows the result of $sql(\text{SOMETIMES\_IN\_PAST } \psi_1, \mathcal{O})$, assuming $\mathcal{O}$ to be our corresponding ontology. This SQL is then combined with the SQL generated for the other subqueries. The final rewriting $sql(\phi^t, \mathcal{O})$ can then be evaluated over a common relational database, and the obtained answers represent the answers to the query ϕ^t w.r.t. $\mathcal{O}$, with the consideration of fuzzy data. Some more detail on the implementation is given next.

```
SELECT
  /** patient */
  VIEW_A.p,
  /** Membership degree */
  Degree.d,
  /** Time point in the future */
  TVIEW_A.timestamp
FROM
  extend(ψ1) AS VIEW_A,
  fuzzify(ψ1) AS Degree,
  (SELECT
     DISTINCT timestamp
   FROM
     timetable) AS TVIEW_A
WHERE
  /** VIEW_A.timestamp is the past time
  point where ψ1 holds **/
  TVIEW_A.timestamp > VIEW_A.timestamp
```

Listing 1: The SQL rewriting of the TSPARQL query (SOMETIMES_IN_PAST ψ_1)

4. IMPLEMENTATION

To provide a fast and efficient solution, we implemented our approach based on existing systems, including algorithms we developed previously. Specifically, we integrated the algorithm implemented in QuAnTOn [21], a system for answering temporal queries over temporal knowledge bases, with the FLite approach [10], which allows for OBQA over fuzzy data. For rewriting the plain SPARQL queries into SQL, we apply the highly optimized Ontop system [15]. Our implementation is written in Java 1.7 and uses a MySQL[8] database for data storage. An overview is given in Figure 3.

The system input consists of (i) (possibly fuzzy) data referencing different time points; (ii) a pair (q^t, d) containing a TSPARQL query q^t and a degree d; the vector $\vec{x}$ represents the variable tuple which is to be instantiated by the answers; and (iii) an ontology $\mathcal{O}$. Our system then rewrites the query as described in the previous section and evaluates the rewritten query, $q^{f,t}_\mathcal{O}$, over the MySQL database. This evaluation yields a set of answers with corresponding degrees. Finally, those of the answers whose degree is $\geq d$ are returned as output.

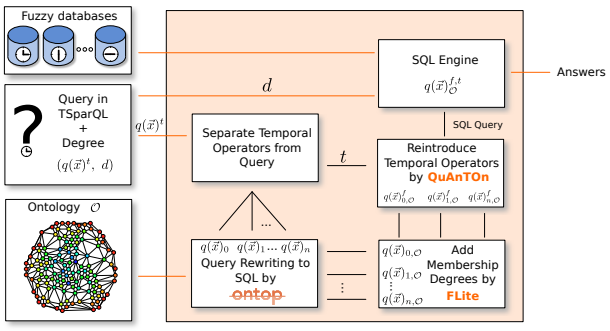

Figure 3: Overview of our implementation for temporal query answering w.r.t. an ontology and fuzzy data.

[8] http://www.mysql.com

5. RELATED WORK

There is a lot of active research on both temporal and fuzzy OBQA. However, the fields are yet separated and, even in theory, the integration of the two aspects has not been considered so far.

Although there is a growing interest of research on the temporal aspects of ontology-based data access, recently, freely available implementations that answer temporal queries are still rare [8, 21]. [8] describe a system tailored to answering rather expressive queries over data streams. [21] prototypically evaluate three different implementations of temporal OBQA, but do not provide fully fledged systems, yet.

Apart from the FLite reasoner, several approaches for answering fuzzy queries in practice have been proposed in the past [18, 19, 11, 22, 5, 17, 4, 12, 20]. However, several of them have been implemented only prototypically [18, 19, 11] or could not be obtained/ installed [12, 20]. In contrast to the fuzzy *DL-Lite*$_{\mathcal{R}}$ reasoner FLite, [22, 5, 17, 4] support more expressive ontology languages; on the other hand, they allow only for class queries as query language (i.e., instead of the more expressive conjunctive queries).

6. SUMMARY & OUTLOOK

We presented an approach for answering temporal queries w.r.t. ontologies that allows to explicitly represent vagueness in the data and the queries. We assume these queries to be useful in many applications, given the fact that temporal and/or fuzzy aspects are usually inherent in real-world data. Based on existing reasoning systems, we implemented our algorithm prototypically.

Future plans include a thorough evaluation targeting specific application scenarios. The latter could also motivate the extension of our approach. In particular, it would be interesting to see if *DL-Lite*$_{\mathcal{R}}$ is expressive enough in practice, or if we have to regard more expressive ontology languages.

7. REFERENCES

[1] Owl 2 Web Ontology Language Profiles (Second Edition). http://www.w3.org/TR/owl2-profiles/.

[2] Andrea Acciarri, Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, Mattia Palmieri, and Riccardo Rosati. QUONTO: Querying Ontologies. In *AAAI*, pages 1670–1671, 2005.

[3] Alessandro Artale, Diego Calvanese, Roman Kontchakov, and Michael Zakharyaschev. The *DL-Lite* Family and Relations. *Journal of artificial intelligence research*, 36(1):1–69, 2009.

[4] Fernando Bobillo, Miguel Delgado, and Juan Gómez-Romero. Reasoning in Fuzzy OWL 2 with DeLorean. In *Uncertainty Reasoning for the Semantic Web II*, pages 119–138. 2013.

[5] Fernando Bobillo and Umberto Straccia. fuzzyDL: An Expressive Fuzzy Description logic Reasoner. In *FUZZ-IEEE*, pages 923–930, 2008.

[6] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, Antonella Poggi, Mariano Rodriguez-Muro, and Riccardo Rosati. *Ontologies and Databases: The DL-Lite Approach*. 2009.

[7] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. Tractable Reasoning and Efficient Query Answering in Description Logics: The *DL-Lite* Family. *Journal of Automated reasoning*, 39, 2007.

[8] Oscar Corcho, Jean-Paul Calbimonte, Hoyoung Jeung, and Karl Aberer. Enabling query technologies for the semantic sensor web. *Int. J. Semant. Web Inf. Syst.*, 8(1):43–63, January 2012.

[9] Stefan Decker, Michael Erdmann, Dieter Fensel, and Rudi Studer. *Ontobroker: Ontology based access to distributed and semi-structured information*. Springer, 1999.

[10] Theofilos Malis, Anni-Yasmin Turhan, and Erik Zenker. A pragmatic approach to answering cqs over fuzzy *DL-Lite*-ontologies - introducing flite. In *Proceedings of the 28th International Workshop on Description Logics (DL-2015)*, June 2015.

[11] Jeff Z Pan, Giorgos B Stamou, Giorgos Stoilos, and Edward Thomas. Expressive Querying over Fuzzy DL-Lite Ontologies. In *Description Logics*, 2007.

[12] Jeff Z Pan, Edward Thomas, and Derek Sleeman. Ontosearch2: Searching and querying web ontologies. *Proc. of WWW/Internet*, 2006:211–218, 2006.

[13] Antonella Poggi, Domenico Lembo, Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Riccardo Rosati. Linking data to ontologies. In *Journal on data semantics X*, pages 133–173. Springer, 2008.

[14] Antonella Poggi, Mariano Rodriguez, and Marco Ruzzi. Ontology-based database access with DIG-Mastro and the OBDA Plugin for Protégé. In *Proc. of OWLED*, 2008.

[15] Mariano Rodriguez-Muro, Roman Kontchakov, and Michael Zakharyaschev. Ontology-Based Data Access: Ontop of Databases. In *International Semantic Web Conference (1)*, pages 558–573, 2013.

[16] Markus Stocker and Michael Smith. Owlgres: A Scalable OWL Reasoner. In *OWLED*, volume 432, 2008.

[17] Giorgos Stoilos, Nikos Simou, Giorgos Stamou, and Stefanos Kollias. Uncertainty and the Semantic Web. *Intelligent Systems*, 21(5):84–87, 2006.

[18] Umberto Straccia. Answering Vague Queries in Fuzzy DL-Lite. In *Proceedings of the 11th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems,(IPMU-06)*, pages 2238–2245, 2006.

[19] Umberto Straccia. Towards Top-k Query Answering in Description Logics: The Case of DL-Lite. In *Logics in Artificial Intelligence*, pages 439–451. Springer, 2006.

[20] Umberto Straccia. SoftFacts: A Top-k Retrieval Engine for Ontology Mediated Access to Relational Databases. In *Systems Man and Cybernetics (SMC), 2010 IEEE International Conference on*, pages 4115–4122. IEEE, 2010.

[21] Veronika Thost, Jan Holste, and Özgür Özçep. On implementing temporal query answering in *DL-Lite*. LTCS-Report 15-12, Chair for Automata Theory, TU Dresden, Germany, 2015. See http://lat.inf.tu-dresden.de/research/reports.html.

[22] Dorothea Tsatsou, Stamatia Dasiopoulou, Ioannis Kompatsiaris, and Vasileios Mezaris. LiFR: A Lightweight Fuzzy DL Reasoner. In *The Semantic Web: ESWC 2014 Satellite Events*, pages 263–267. 2014.

[23] Tassos Venetis, Giorgos Stoilos, and Giorgos Stamou. Query Extensions and Incremental Query Rewriting for OWL 2 QL Ontologies. *Journal on Data Semantics*, pages 1–23, 2014.