OWL2 based Data Cleansing Using Conditional Exclusion Dependencies

Olivier Curé[1], Chan Le Duc[2], Myriam Lamolle[2]

[1] Université Paris-Est, LIGM, Marne-la-Vallée, France
ocure@univ-mlv.fr
[2] LIASD Université Paris 8 - IUT de Montreuil
chan.leduc,myriam.lamolle@iut.univ-paris8.fr

1 Introduction

Ontology-based Data Access (OBDA) [6] aims to provide access to heterogeneous data sources through a mediating ontology. Most research conducted in this area tackles ontology expressivity, computational efficiency of reasoning services and inferences associated to query answering. In this paper, we argue that data quality and data cleansing are domains where OBDA could contribute in an efficient manner. That is we aim to prevent the execution of update operations which are corrupting database instances and to propose their (semi-)automatic cleansing. This can be performed by handling integrity constraints (ICs), especially those that can not be easily represented and processed in a strict relational context.

In [3], we have already proposed a first solution based on extensions of standard dependencies, i.e. Conditional Functional and INclusion Dependencies (henceforth denoted CFDs and CINDs) in the context of OWL2. Based on experiments conducted on real world databases [2], we found out that a form of Conditional Exclusion Dependency (CED) may be relevant in capturing more real-life data inconsistencies. To the best of our knowledge, this work is a first approach to address data quality and cleansing problems using CEDs in either a strict relational or OBDA context. Intuitively, CEDs correspond to standard exclusion dependencies extended with a pattern tableau (inspired from tableau queries presented in [1]) containing variables and constant values which are part of the active domain of the database. The discovery of CEDs is a hard problem since negative information are generally not stored in relational databases. In this work, we highlight that novel OWL2 constructs, e.g. negative property assertions, can assist our system in this discovery process. Moreover, the use of OWL2 reasoning services over ontologies containing concept and property axioms enables to represent and process CEDs in a compact and efficient way using the formalism of SPARQL queries.

In the rest of this paper, we consider that elements of a domain ontology are mapped to relations of a relational database. Due to space limitation, we do not present these mapping assertions on our running example.

2 Conditional Exclusion Dependencies

An exclusion dependency (ED) [1] corresponds to forbidding the appearance of a given tuple in a relation S when a tuple appears in a relation R and is represented as the following axiom: $\forall x, y, z, x', z' R(x, y, z) \rightarrow \neg S(x', x, z')$.

A conditional extension of an ED forbids the appearance of tuples in S when tuples satisfying a set of patterns appear in R. Formally, a CED ϕ, defined over a pair of relations R and S, is a pair $(R(X; X_p) \subseteq \neg S(Y; Y_p), T_p)$ where X, X_p and Y, Y_p are attribute sets of respectively R and S. $R(X) \subseteq \neg S(Y)$ is a standard exclusion dependency and T_p is a tableau pattern of ϕ with attribute sets X_p and Y_p such that for each pattern t_p and each attribute B in X_p and Y_p, $t_p[B]$ is either a constant in the domain of B or a wild card, denoted $'\_'$.

An instance (I_1, I_2) of (R, S) satisfies a CED ϕ, denoted $(I_1, I_2) \models \phi$, iff for each tuple t_p in T_p and for each t_1 in I_1, if $t_1[X_p] = t_p[X_p]$ then there does not exist a tuple t_2 in S such that $t_1[X] = t_2[Y]$ and $t_2[Y_p] = t_p[Y_p]$.

Example 1: We consider an extract of a medical database with the following relations: `drug(idDrug, nameDrug, form)`, `contraDrug(idDrug, idContra)` and `atcDrug(idDrug, atcCode)` which respectively contain information concerning drug products (with an identifier, name and the form of the product, e.g. allopathy, homeopathy), contraindications of drug products (with a drug identifier and a contraindication identifier) and molecules of drugs identified by ATC[3] codes. In this context, the following CEDs hold:
$\phi_1 : (drug(idDrug; form) \subseteq \neg contraDrug(idDrug; idContra), T_{p1})$ with T_{p1} as $\{('\_';' homeopathy' \| '\_';' \_'), ('\_';' phytotherapy' \| '\_';' \_')\}$
$\phi_2; (atcDrug(idDrug; atcCode) \subseteq \neg contraDrug(idDrug; idContra), T_{p2})$ where T_{p2} is: $\{('\_';' R5DA9' \| '\_';'\text{Anti-coughing}')\}$
These CEDs respectively state that homeopathy and phytotherapy drugs do not have contraindications and that the molecule identified with 'R5DA9' must not be contraindicated to 'anti-coughing'. Note that these forms of CEDs, i.e. constants in the left hand side only for ϕ_1 and constants on both sides in ϕ_2, correspond to the most widely encountered CEDs in studied use cases.

3 Discovery approaches

OWL2 ontologies correspond to the $\mathcal{SROIQ}$ description logic [4] which allows for new role constructors such as composition, disjointness and negation. This enables to represent RBox axioms of the form $R \sqsubseteq \neg S$ where R and S are both DL roles. Note that this axiom corresponds to an exclusion dependency where the relations are necessarily binary and supports the discovery of EDs, i.e. CEDs with an empty pattern tableau. Such axioms are frequently encountered in role hierarchies. For instance, consider a property `hasContraIndication` with two subroles, `hasDiseaseContraIndication` and `hasDrugContraIndication`. Then it would be useful to state that these two subroles are disjoint. Note that class

[3] Anatomical Therapeutic Chemical: http://www.whocc.no/atcddd/

disjointness, already available in the first version of OWL, can also be used to identify CEDs. In both cases, OBDA 's mapping assertions need to be considered in order to maintain the data quality of underlying relational databases.

Another form of CED related axioms found in OWL2 ontologies is supported by General Concept Inclusion (GCI) of the form: $\exists R.C \sqsubseteq \neg\exists S.D$ where concept C corresponds to a nominal and D is either a nominal or the top concept ($\top$). In the context of our running example, consider that correspondences between the `atcDrug` and `contraDrug` are defined with resp. `hasATCCode` and `hasContraDrug` and the `hasForm` property is mapped to the `form` attribute of the `drug` relation, then the following axioms correspond to resp. ϕ_1 and ϕ_2:

$$\exists form.\{homeo\} \sqsubseteq \neg\exists contraDrug.\top$$
$$\exists hasATCCode.\{R5DA9\} \sqsubseteq \neg\exists contraDrug.\{\text{Anti-coughing}\}.$$

Moreover, OWL2 ABoxes enable the definition of negative property assertions which together with property subsumption axioms enable to define CEDs. In an OBDA context, all the extensional data are stored in a (relational) database and serve to generate an ABox satisfying an ontology, e.g. by using a systems like QUONTO [6] or SOR [5]. Hence, end-users generally do not store assertions directly in the ABox. Nevertheless, such an approach could be useful to discover the pattern tableaux of our CEDs. These assertions could be defined as a complementary ABox and would mainly serve to store CEDs and enable some inferences. That is they would not be stored in the relational database generating the ABox.

4 Representation and Processing of CEDs

We propose to represent the CEDs discovered using the formalism of SPARQL queries. This fits into the approach of [7] where the author argued that ICs are epistemic in nature and concern "what the knowledge base knows". These queries aim to detect violations of CEDs and are generated by considering a CED has a graph over elements of the domain ontology. In this graph, the negated property is asserted to be true. Thus a translation of this graph into an SPARQL query enables to detect objects being violated. These objects are identified by the query's distinguished variables which are selected using axioms of the domain ontology, e.g. domain and range of properties. They are working over the knowledge base underlying the application domain and which is mapped to the relational domain. Hence, using an approach similar to the QUONTO system, it is possible to translate these queries into SQL queries executed over relational databases. The SPARQL queries enabling to detect violations of ϕ_1 and ϕ_2 are respectively:

```
SELECT ?x WHERE { ?x rdf:type :ATC. ?y rdf:type :Drug.
                  ?y :hasATCCode ?x. ?y :form 'homeopathy'.}
SELECT ?x WHERE { ?x rdf:type :Drug. ?y rdf:type :ATC.
                  ?x :hasATCCode ?y. ?y :nameAtc 'R5DA9'.
                  ?z rdf:type :Contra. ?x :hasContraDrug ?z.
                  ?z :nameContra 'Anti-coughing'.}
```

Moreover, in order to represent them in a compact way, we exploit and analyze the hierarchies of concepts present as constants in object properties used in CEDs. We provide an example of the use of such inferences with the ATC classification which divides drug molecules into different groups according to the organ or system on which they act and/or their therapeutic and chemical characteristics. The classification is organized in 5 levels with each level encoded using a letter or digits. For instance, the R5CA code subsumes 11 molecules, R5CA1 to R5CA11, which act as expectorants.

Example 2: Consider CED ϕ_2 with a pattern tableau T_{p2} containing all 11 descendants of the R5CA code as X_p and with the 'expectorant' constant in Y_p. Then it will be much more compact to store one tuple with the 'R5CA' code than 11 tuples containing its subsumed codes. The pattern would look like: $('\_';'R5CA'\|'\_';'\text{Expectorant}')$. Note that in the medical domain, such generalizations frequently occur since molecules of a given family generally possess common properties.

The main idea of this approach consists of generating a SPARQL query for each sub concept of the concept stored in a CED. The next step corresponds to the detection of a CED violation. Such detection is activated whenever a tuple of the data sources is updated, i.e. after the execution of a CRUD operation. In the context of a relational database, this can be handled by the definitions of SQL triggers. In fact, we automatically generate an AFTER/ROW LEVEL SQL trigger for each relation mapped to a property involved in a CED. These triggers call some generic programmed methods (in Java) defined in the framework of our data quality system. The purpose of these methods is to execute SPARQL queries and hence to discover and identify the data source tuples causing some inconsistencies. Note that we must associate a trigger to both the left and right hand side relations of a CED to ensure the consistency of data sources.

Finally, we consider that the full potential of a data quality and cleansing implementation based on conditional dependencies lies in the study of possible interactions between discovered sets of CFDs, CINDs and CEDs.

References

1. S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison-Wesley, 1995.
2. O. Curé. Improving the data quality of drug databases using conditional dependencies and ontologies. *ACM Journal of Data and Information Quality (JDIQ)*.
3. O. Curé. Improving the data quality of relational databases using obda and owl2ql. In *OWLED*, 2009.
4. I. Horrocks, O. Kutz, and U. Sattler. The even more irresistible $\mathcal{SROIQ}$. In *KR*, pages 57–67, 2006.
5. J. Lu, L. Ma, L. Z. 0007, J.-S. Brunner, C. Wang, Y. Pan, and Y. Yu. Sor: A practical system for ontology storage, reasoning and search. In *VLDB*, pages 1402–1405, 2007.
6. A. Poggi, D. Lembo, D. Calvanese, G. D. Giacomo, M. Lenzerini, and R. Rosati. Linking data to ontologies. *J. Data Semantics*, 10:133–173, 2008.
7. R. Reiter. On integrity constraints. In *Proc. TARK*, 1988.