

# Discovering Functional Dependencies for Multidimensional Design

Oscar Romero  
Dept. Lleng. i Sistemes Inf.  
Univ. Politècnica de Catalunya  
Barcelona, Spain  
oromero@lsi.upc.edu

Diego Calvanese  
KRDB Research Centre  
Free Univ. of Bozen-Bolzano  
Bolzano, Italy  
calvanese@inf.unibz.it

Alberto Abelló  
Dept. Lleng. i Sistemes Inf.  
Univ. Politècnica de Catalunya  
Barcelona, Spain  
aabello@lsi.upc.edu

Mariano Rodríguez-Muro  
KRDB Research Centre  
Free Univ. of Bozen-Bolzano  
Bolzano, Italy  
rodriguez@inf.unibz.it

## ABSTRACT

Nowadays, it is widely accepted that the data warehouse design task should be largely automated. Furthermore, the data warehouse conceptual schema must be structured according to the multidimensional model and as a consequence, the most common way to automatically look for subjects and dimensions of analysis is by discovering functional dependencies (as dimensions functionally depend on the fact) over the data sources. Most advanced methods for automating the design of the data warehouse carry out this process from relational OLTP systems, assuming that a RDBMS is the most common kind of data source we may find, and taking as starting point a relational schema. In contrast, in our approach we propose to rely instead on a conceptual representation of the domain of interest formalized through a domain ontology expressed in the *DL-Lite* Description Logic. We propose an algorithm to discover functional dependencies from the domain ontology that exploits the inference capabilities of *DL-Lite*, thus fully taking into account the semantics of the domain. We also provide an evaluation of our approach in a real-world scenario.

## Categories and Subject Descriptors

H.4.2 [Information Systems Applications]: Types of Systems—*Decision Support*; H.2.1 [Database Management]: Logical Design

## General Terms

Algorithms, Design, Experimentation, Management, Performance

## Keywords

Functional dependencies, Ontologies, Multidimensional design

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

*DOLAP'09*, November 6, 2009, Hong Kong, China.

Copyright 2009 ACM 978-1-60558-801-8/09/11 ...\$10.00.

## 1. INTRODUCTION

Like in most information systems, the data warehouse design has been typically carried out manually, and the experts' ability and experience are crucial to successfully perform the modeling task and eventually give rise to the desired data warehousing system. The data warehouse conceptual schema must be derived by a *reengineering* process from the data sources and to some extent, this process should be automatic. In this sense, some efforts have proposed the automation of the data warehouse design to free this task from being (completely) performed by an expert, and facilitate the whole process. Mostly, these approaches carry out this process from relational OLTP (*On-Line Transaction Processing*) systems, assuming that a RDBMS is the most common kind of data source we may find, and taking as a starting point a relational (i.e., logical) schema.

Furthermore, it is widely accepted that the conceptual schema of the data warehouse must be structured according to the *multidimensional model*. As a consequence, the most common way to automatically look for subjects (i.e., facts) and dimensions of analysis is by discovering functional dependencies (since dimensions functionally depend on the fact) over the data sources [22].

Starting from a logical schema may present some inconveniences, however. A logical schema is tied to the design decisions made when devising the system and these decisions either made to fulfill the system requirements (for instance, improve query answering, avoid insertion / deletion anomalies, preserve features inherited from legacy systems, etc.) or naively made by non-expert users, have a big impact on the quality of the multidimensional schemas obtained by current automatable approaches. In fact, these approaches require a certain degree of normalization in the input logical schema to guarantee that it captures as much as possible the to-one relationships (i.e., functional dependencies) existing in the domain. Discovering this kind of relationships is crucial in the design of the data warehouse [22], and the most common way to represent them at the logical level is by means of "foreign" (FK) and "candidate key" (CK) constraints. Therefore, the accuracy of the obtained results (and specially, the quality of the found dimension hierarchies) depends on the degree of normalization of the logical schema, since some FK's and CK's are lost if we do not consider a schema up to third normal form.

This scenario can be avoided by modeling the data warehouse from a conceptual formalization of the domain. The role of a con-

ceptual layer on the top of information systems has been discussed in depth in the literature and in case of reengineering processes, like the data warehouse conceptual design, the benefits are clear: the conceptual layer provides more and better knowledge about the domain to carry out this task. In [26] we introduced AMDO (*Automating Multidimensional Design from Ontologies*), our approach for automatically deriving the multidimensional schema from a domain ontology. Our goals are mainly two: (i) to improve the quality of the obtained output (by working over a conceptual formalization of the domain instead of a logical one) and (ii) to automate the process. Because of this second assumption we choose ontologies as input for our method.

Nowadays ontology languages are widely used in different areas like data integration and the Semantic Web, but in other areas, like software engineering, UML and ER are the most common choices. Nevertheless, UML / ER are graphical languages that do not automate reasoning *per se*. Consequently, current automated reasoning tools and techniques for UML / ER class diagrams perform an explicit or implicit translation to languages that do allow automates reasoning. There are two main approaches for automated reasoning over UML / ER: translating the class-diagram into *first order logic* or into *description logics* (DL). The first approach is able to reason with very expressive schemas but it is not *decidable* whereas the second one provides *decidable* reasoning by restricting the language expressivity. We follow the latter, and in those scenarios where the domain is formalized by a UML / ER class-diagram we automatically translate it to DL (see Section 2 for further details).

About AMDO, it considers all the multidimensional concepts in depth by analyzing their semantics and how they should be identified from an ontology. As a result, it proposes new and original design patterns. These patterns demand to compute the *closure* of the functional dependencies stated in the conceptual domain. Due to space limitations, we refer to [26] for a detailed discussion on how to use the functional dependencies for discovering every multidimensional concept. As a major drawback, AMDO was not able to take advantage of generic reasoning algorithms provided by DL to compute these patterns and therefore, was not exploiting the reasoning capabilities provided by ontology languages.

To match our approach to traditional reasoning over ontology languages and better exploit their reasoning capabilities, in this paper we introduce an approach for discovering functional dependencies from a domain ontology. Therefore, we propose to redesign AMDO's core. Unlike in our previous approach, we do use generic reasoning algorithms so that no ad-hoc techniques and tools are needed but just a generic reasoner such as FaCT++ [20] or Racer [18].

The paper is structured as follows. Section 2 introduces *DL-Lite<sub>A</sub>*, a well-behaved DL with a good expressive power, that we will use as AMDO's input ontology language. Thus, AMDO will be able to handle any ontology expressible in *DL-Lite<sub>A</sub>* as well as UML / ER class-diagrams. For the latter, we also discuss briefly how they should be translated to *DL-Lite<sub>A</sub>*. Section 3 presents our approach for discovering fd's for *DL-Lite<sub>A</sub>* ontologies, and Section 4 introduces the algorithm for computing them. In Section 5, we consider the case of a *DL-Lite<sub>A</sub>* ontology that has been derived from a UML / ER schema. Finally, Section 6 wraps up the discussion with a practical use of our approach.

## 2. CONCEPTUAL MODELING USING DESCRIPTION LOGICS

Description Logics (DLs) [3] originate in the mid '80s to provide a formal basis to structured knowledge representation languages.

We make use of *DL-Lite<sub>A</sub>*, a DL of the *DL-Lite* family [7, 6], which is particularly well suited for conceptual modeling due its ideal trade-off between expressive power and computational properties. We first present *DL-Lite<sub>A</sub>*, and then illustrate its modeling capabilities by means of an example.

### 2.1 A Tractable Description Logic: *DL-Lite<sub>A</sub>*

In DLs, objects with common properties are grouped into *concepts*, and the properties are represented through *roles*, denoting binary relations over the domain of interest. Complex concepts and roles are built inductively by starting from atomic ones (i.e., simply concept and role names) and applying a set of constructs. Different from traditional DLs, and following what done in other conceptual modeling formalisms such as UML class diagrams, *DL-Lite<sub>A</sub>* distinguishes between (abstract) objects and (data) values. Hence, it distinguishes concepts, denoting sets of objects, from *value-domains*, denoting sets of values, and roles, denoting binary relations between objects, from *attributes*, denoting binary relations between objects and values. More precisely, concepts, roles, value-domains, and attributes in *DL-Lite<sub>A</sub>* are formed starting from atomic elements according to the following syntax (where the distinction between *basic* and *arbitrary* elements is relevant in what follows):

	atomic	basic	arbitrary
concept	$A$	$B \rightarrow A \mid \exists Q \mid \delta(U)$	$C \rightarrow B \mid \neg B$
role	$P$	$Q \rightarrow P \mid P^-$	$R \rightarrow Q \mid \neg Q$
value-domain		$E \rightarrow \rho(U)$	$F \rightarrow \top_D \mid T_1 \mid \dots \mid T_n$
attribute	$U$	$V \rightarrow U$	$W \rightarrow V \mid \neg V$

Above,  $\delta(U)$  denotes the *domain* of  $U$ , i.e., the set of objects that  $U$  relates to values;  $\rho(U)$  denotes the *range* of  $U$ , i.e., the set of values that  $U$  relates to objects;  $\top_D$  is the universal value-domain;  $T_1, \dots, T_n$  are  $n$  pairwise disjoint unbounded value-domains, corresponding to data types, such as `string`, `integer`, etc. In the following, let  $\text{Inv}(Q) = P^-$  when  $Q = P$ , and  $\text{Inv}(Q) = P$  when  $Q = P^-$ .

In *DL-Lite<sub>A</sub>*, knowledge about the domain is represented by means of an *ontology* (or knowledge base), consisting of a TBox, encoding intensional knowledge, and an ABox, encoding extensional knowledge on specific objects. Specifically, a *DL-Lite<sub>A</sub>* TBox is constituted by a set of assertions of the form

$$B \sqsubseteq C, \quad Q \sqsubseteq R, \quad E \sqsubseteq F, \quad V \sqsubseteq W, \quad (\text{funct } Q), \quad (\text{funct } U),$$

which respectively denote an inclusion between a basic and an arbitrary concept, role, value-domain, and attribute, and functionality on a role and on an attribute. As for the ABox, we introduce two disjoint alphabets,  $\Gamma_O$  of *object constants* denoting objects, and  $\Gamma_V$  of *value constants* denoting data values. A *DL-Lite<sub>A</sub>* ABox is a finite set of *membership assertions* of the form (where  $a, b \in \Gamma_O$  and  $c \in \Gamma_V$ ):

$$A(a), \quad P(a, b), \quad U(a, c).$$

*Definition 1.* A *DL-Lite<sub>A</sub>* ontology  $\mathcal{O}$  is a pair  $\langle \mathcal{T}, \mathcal{A} \rangle$ , where  $\mathcal{T}$  is a *DL-Lite<sub>A</sub>* TBox,  $\mathcal{A}$  is a *DL-Lite<sub>A</sub>* ABox, and the following conditions are satisfied:

- (1) for each atomic role  $P$ , if either  $(\text{funct } P)$  or  $(\text{funct } P^-)$  occur in  $\mathcal{T}$ , then  $\mathcal{T}$  does not contain assertions of the form  $Q \sqsubseteq P$  or  $Q \sqsubseteq P^-$  (for  $Q$  a basic role);
- (2) for each atomic attribute  $U$ , if  $(\text{funct } U)$  occurs in  $\mathcal{T}$ , then  $\mathcal{T}$  does not contain assertions of the form  $V \sqsubseteq U$  (for  $V$  an atomic attribute).

Intuitively, these two conditions say that, in a  $DL\text{-Lite}_A$  TBox, roles and attributes occurring in functionality assertions cannot be specialized. These conditions are crucial for the tractability of reasoning [24].

The semantics of  $DL\text{-Lite}_A$  is given in terms of FOL interpretations. An *interpretation*  $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$  consists of a first order structure over the interpretation domain  $\Delta^{\mathcal{I}}$  that is the disjoint union of  $\Delta_O^{\mathcal{I}}$  and  $\Delta_V^{\mathcal{I}}$ , and of an *interpretation function*  $\cdot^{\mathcal{I}}$  such that  $a^{\mathcal{I}} \in \Delta_O^{\mathcal{I}}$  for all  $a \in \Gamma_O$ ,  $c^{\mathcal{I}} \in \Delta_V^{\mathcal{I}}$  for all  $c \in \Gamma_V$ , and such that the following conditions are satisfied (below,  $o, o' \in \Delta_O^{\mathcal{I}}$ , and  $v \in \Delta_V^{\mathcal{I}}$ ):

$$\begin{aligned} A^{\mathcal{I}} &\subseteq \Delta_O^{\mathcal{I}} \\ (\exists Q)^{\mathcal{I}} &= \{o \mid \exists o'. (o, o') \in Q^{\mathcal{I}}\} & P^{\mathcal{I}} &\subseteq \Delta_O^{\mathcal{I}} \times \Delta_O^{\mathcal{I}} \\ (\delta(U))^{\mathcal{I}} &= \{o \mid \exists v. (o, v) \in U^{\mathcal{I}}\} & (P^-)^{\mathcal{I}} &= \{(o, o') \mid (o', o) \in P^{\mathcal{I}}\} \\ (\neg B)^{\mathcal{I}} &= \Delta_O^{\mathcal{I}} \setminus B^{\mathcal{I}} & (\neg Q)^{\mathcal{I}} &= (\Delta_O^{\mathcal{I}} \times \Delta_O^{\mathcal{I}}) \setminus Q^{\mathcal{I}} \\ \frac{T^{\mathcal{I}}}{\perp} &\subseteq \Delta_V^{\mathcal{I}} & U^{\mathcal{I}} &\subseteq \Delta_O^{\mathcal{I}} \times \Delta_V^{\mathcal{I}} \\ \frac{\perp}{D} &= \Delta_V^{\mathcal{I}} & (\neg V)^{\mathcal{I}} &= (\Delta_O^{\mathcal{I}} \times \Delta_V^{\mathcal{I}}) \setminus V^{\mathcal{I}} \\ (\rho(U))^{\mathcal{I}} &= \{v \mid \exists o. (o, v) \in U^{\mathcal{I}}\} \end{aligned}$$

We assume that the *unique name assumption* holds, i.e., different (object and value) constants are interpreted as different domain elements.

We define now when an interpretation  $\mathcal{I}$  satisfies a TBox or ABox assertion. Specifically,  $\mathcal{I}$  *satisfies*:

- $\alpha_1 \sqsubseteq \alpha_2$ , if  $\alpha_1^{\mathcal{I}} \subseteq \alpha_2^{\mathcal{I}}$ ;
- (funct  $\beta$ ), where  $\beta$  is either  $P$ ,  $P^-$ , or  $U$ , if  $(o, e_1) \in \beta^{\mathcal{I}}$  and  $(o, e_2) \in \beta^{\mathcal{I}}$  implies  $e_1 = e_2$ , for each  $o \in \Delta_O^{\mathcal{I}}$ , and  $e_1, e_2$  in either  $\Delta_O^{\mathcal{I}}$  or  $\Delta_V^{\mathcal{I}}$ ;
- $A(a)$  if  $a^{\mathcal{I}} \in A^{\mathcal{I}}$ ,  $P(a, a')$  if  $(a^{\mathcal{I}}, a'^{\mathcal{I}}) \in P^{\mathcal{I}}$ , and  $U(a, c)$  if  $(a^{\mathcal{I}}, c^{\mathcal{I}}) \in U^{\mathcal{I}}$ .

A *model* of an ontology  $\mathcal{O}$  (resp., TBox  $\mathcal{T}$ ) is an interpretation  $\mathcal{I}$  that satisfies all assertions in  $\mathcal{O}$  (resp.,  $\mathcal{T}$ ). An ontology  $\mathcal{O}$  (resp., TBox  $\mathcal{T}$ ) is *satisfiable* if it has at least one model, and  $\mathcal{O}$  (resp.,  $\mathcal{T}$ ) *logically implies* an assertion  $\alpha$ , denoted  $\mathcal{O} \models \alpha$  (resp.,  $\mathcal{T} \models \alpha$ ), if  $\alpha$  is satisfied in all models of  $\mathcal{O}$  (resp.,  $\mathcal{T}$ ).

A conjunctive query (CQ)  $q$  over an ontology  $\mathcal{O}$  is an expression of the form  $q(\vec{x}) \leftarrow \text{body}(\vec{x}, \vec{y})$ , where  $\vec{x}$  are the so-called *distinguished variables*,  $\vec{y}$  are the *non-distinguished variables*, and  $\text{body}(\vec{x}, \vec{y})$  is a set of atoms of the form  $A(x_o)$ ,  $P(x_o, x'_o)$ ,  $T_i(x_v)$ , or  $U(x_o, x_v)$ , where  $x_o, x'_o$  are variables in  $\vec{x}$  or  $\vec{y}$  or constants in  $\Gamma_O$ , and  $x_v$  is a variable in  $\vec{x}$  or  $\vec{y}$  or a constant in  $\Gamma_V$ . Notice that CQs corresponds to SELECT-PROJECT-JOIN SQL queries, and hence are the queries most commonly posed to relational DBs. The query  $q(\vec{x}) \leftarrow \text{body}(\vec{x}, \vec{y})$  is interpreted in  $\mathcal{I}$  as the set  $q^{\mathcal{I}}$  of tuples  $\vec{e} \in \Delta^{\mathcal{I}} \times \dots \times \Delta^{\mathcal{I}}$  such that, when we assign  $\vec{e}$  to the variables  $\vec{x}$ , the first-order logic formula  $\exists \vec{y}. \varphi(\vec{x}, \vec{y})$ , where  $\varphi(\vec{x}, \vec{y})$  is the conjunction of atoms in  $\text{body}(\vec{x}, \vec{y})$ , evaluates to true in  $\mathcal{I}$ . The reasoning service we are interested in is (*conjunctive query answering*): given an ontology  $\mathcal{O}$  and a query  $q$  over  $\mathcal{O}$ , return the *certain answers* to  $q$  over  $\mathcal{O}$ , i.e., all tuples  $\vec{t}$  of elements of  $\Gamma_V \cup \Gamma_O$  such that  $\vec{t}^{\mathcal{I}} \in q^{\mathcal{I}}$  for every model  $\mathcal{I}$  of  $\mathcal{K}$ , denoted  $\mathcal{K} \models q(\vec{t})$ .

As shown in [7, 24], all forms of inference over a  $DL\text{-Lite}$  ontology (e.g., satisfiability, logical implication, and CQ answering) can be done in polynomial time in the size of the TBox, and in  $AC_0^1$  in the size of the ABox (i.e., w.r.t. data complexity [32]). In particular, to compute the certain answers of a CQ  $q$ , we can: (i) by using the assertions in the TBox, rewrite  $q$  to a union  $Q$  of

<sup>1</sup> $AC_0$  is the complexity class that corresponds to the complexity in the size of the data of evaluating a first-order (i.e., SQL) query over a relational database (see, e.g., [1]).

CQs (which is directly translatable to an SQL query), and (ii) evaluate  $Q$  over the database corresponding to the ABox assertions. In this way, all forms of inference in  $DL\text{-Lite}_A$  can be carried out by exploiting standard commercial relational DB technology for manipulating the data (i.e., the ABox).

## 2.2 DLs and Conceptual Schemas

DLs share many similarities with representation formalisms used in different contexts, such as UML class diagrams<sup>2</sup> (UML-CDs), and the correspondence with these formalisms has been analyzed in detail [11, 4]. By virtue of this correspondence, automated inference algorithms developed for DLs can be used to reason over UML-CDs. Specifically,  $DL\text{-Lite}_A$  has been designed so as to capture the most important features of such diagrams, while keeping the complexity of inference low. We illustrate how a  $DL\text{-Lite}_A$  TBox can capture a UML-CD by means of an example, and refer to [9] for more details.

Consider the UML-CD in Figure 1. Intuitively, each UML class in the diagram is represented by an atomic concept in the TBox, each UML (binary)<sup>3</sup> association by an atomic role, and each UML attribute by an atomic attribute (we assume here that the name of the class is part of the attribute name). We describe how suitable TBox assertions capture the constraints imposed on the domain by the UML-CD.

- A generalization between two classes is represented by means of an inclusion assertion between the corresponding concepts, e.g.,  $\text{Reservation} \sqsubseteq \text{RentalAgreement}$ .
- To represent the domain (i.e., first component) and range (i.e., second component) of an association  $P$ , we use  $\exists P$  and  $\exists P^-$ , respectively. E.g., to represent that the domain of the  $\text{LocatedAt}$  association is contained in  $\text{Branch}$  we use  $\exists \text{LocatedAt} \sqsubseteq \text{Branch}$ , and to represent that the range is contained in  $\text{Country}$ , we use  $\exists \text{LocatedAt}^- \sqsubseteq \text{Country}$ .
- To represent domain and range of an attribute  $U$ , we use  $\delta(U)$  and  $\rho(U)$ , respectively. E.g., to represent that  $\text{bestPrice}$  is an attribute with domain  $\text{RentalAgreement}$  and range  $\text{Money}$ , we use  $\delta(\text{bestPrice}) \sqsubseteq \text{RentalAgreement}$  and  $\rho(\text{bestPrice}) \sqsubseteq \text{Money}$ .
- To capture mandatory participation in an association (i.e., a min. multiplicity 1), we use e.g.,  $\text{Branch} \sqsubseteq \exists \text{IsOfType}$  or  $\text{Customer} \sqsubseteq \exists \text{Makes}^-$ .
- To capture functionality of an association (i.e., a max. multiplicity 1), we use e.g., (funct  $\text{LocatedAt}$ ) or (funct  $\text{Makes}^-$ ).
- Finally, to capture disjointness between classes, we use negation on concepts, as e.g., in  $\text{Car} \sqsubseteq \neg \text{Branch}$ .

By virtue of the reasoning capabilities of  $DL\text{-Lite}_A$  and of the encoding described above, inference over UML-CDs can be carried out by relying on the reasoning services provided by reasoners for  $DL\text{-Lite}_A$ , e.g., by the system QuOnto [2, 25].

## 3. DISCOVERING FDS

In this section, we present our approach to discovering functional dependencies by relying on the assertions in an ontology.

<sup>2</sup>Our considerations apply, *mutatis mutandis*, also to ER schemas [12].

<sup>3</sup>Associations of arity greater than 2 can be handled through *reification* [11, 9].

We first recall some basic definitions regarding functional dependencies in the standard relational model (see, e.g., [1]). Consider a relation schema  $R$ , i.e., a relation symbol with an associated set of attributes, each denoting one component of  $R$ . A *functional dependency* (fd) over  $R$  has the form  $R: X \rightarrow Y$ , where  $X$  and  $Y$  are sets of attributes of  $R$ . We say that a relation  $r$  for  $R$  satisfies such a dependency if for each pair  $t_1, t_2$  of tuples in  $r$  such that  $\pi_X(t_1) = \pi_X(t_2)$ , we have  $\pi_Y(t_1) = \pi_Y(t_2)$  (where, as usual,  $\pi_X(t)$  denotes the projection of tuple  $t$  on the attributes in  $X$ ).

It is well known (cf. [1]) that the following set of inference rules is sound and complete for implication of fds over a relation schema  $R$  (below,  $X, Y$ , and  $Z$  are sets of attributes of  $R$ , and juxtaposition of two sets stands for their union):

- If  $Y \subseteq X$ , then  $R: X \rightarrow Y$  (*reflexivity*).
- If  $R: X \rightarrow Y$ , then  $R: XZ \rightarrow YZ$  (*augmentation*).
- If  $R: X \rightarrow Y$  and  $R: Y \rightarrow Z$ , then  $R: X \rightarrow Z$  (*transitivity*).

In other words, all fds derived from a set  $\mathcal{F}$  of fds over  $R$ , i.e., the  $\mathcal{F}$ -closure, can be computed by starting from  $\mathcal{F}$  and exhaustively applying the above inference rules.

### 3.1 Functional dependencies over ontologies

We would like now to carry over to the conceptual level the standard notion of fd defined at the logical level. To this aim, we introduce the notion of fd over an ontology. We observe that previous work has already considered fds in the context of ontologies, see e.g., [10, 30, 31]. In these works, mimicking the notion in the relational model, a fd ensures that, if two objects that are instances of some concept share the same values for a set of attributes (or of attribute chains), then they share also the value of an additional attribute (or attribute chain), namely the attribute (chain) that functionally depends on the former attributes (chains). Instead, for the purposes described in Section 1, a fd should capture the intuition that the instances of one concept *functionally depend* on the instances of another concept. In other words, given two concepts<sup>4</sup>  $C_1$  and  $C_2$ , we are interested in establishing whether each instance of  $C_1$  allows one to determine a unique instance of  $C_2$ . We will denote this by  $C_1 \rightarrow C_2$ . Several observations are in order:

(i) The dependency between the two concepts  $C_1$  and  $C_2$  needs to be established explicitly, and this can be done by means of some role that relates  $C_1$  to  $C_2$ .<sup>5</sup>

(ii) Since each instance of  $C_1$  should determine a unique instance of  $C_2$ , and such a dependency is established through a role, we need to require such a role to be functional.

(iii) If we want to ensure a property analogous to transitivity (i.e., if  $C_1 \rightarrow C_2$  and  $C_2 \rightarrow C_3$ , then also  $C_1 \rightarrow C_3$ ), we need to allow the dependency to be established not only by atomic roles, but also by composite roles (i.e., role chains).

(iv) In an ontology, roles are not necessarily typed, i.e., they do not necessarily have a specified concept as domain and a specified concept as range. Therefore, one cannot establish in general that a role relates one concept to another concept. As a consequence, every untyped role would potentially allow one to establish that two arbitrary concepts are functionally dependent on each other, provided that the role relates one object to a single other object, i.e., that it is functional. This is clearly unsatisfactory, and therefore we need to enforce some stricter condition for a functional dependency  $C_1 \rightarrow C_2$  to hold. Specifically, we will require not only that the

<sup>4</sup>All our considerations can be easily extended to the case where  $C_2$  is a value-domain instead of concept. We stick to pairs of concepts for space reasons.

<sup>5</sup>Note that in the relational model, attributes that functionally depend on other attributes are implicitly related through the relation schema to which the attributes belong.

role is functional, but also that the instances of  $C_1$  mandatorily participate to the role, and that the role necessarily relates them to an instance of  $C_2$ .

The above observations lead us to the following definition of when a fd between two  $DL\text{-Lite}_A$  concepts holds in a given interpretation. We make use of the notion of *role chain*  $Q_1 \circ \dots \circ Q_n$  of basic roles, interpreted as the composition of the binary relations corresponding to the roles. Formally, for an interpretation  $\mathcal{I}$ , we have that

$$(Q_1 \circ \dots \circ Q_n)^{\mathcal{I}} = Q_1^{\mathcal{I}} \circ \dots \circ Q_n^{\mathcal{I}}.$$

We can then apply concept and role constructs to role chains (instead of basic roles), and the semantics naturally extends the one for the case of basic roles. When we talk about a role chain  $Q_1 \circ \dots \circ Q_n$  over a TBox  $\mathcal{T}$ , we intend that for each  $i \in \{1, \dots, n\}$ , at least one of  $Q_i$  or  $\text{Inv}(Q_i)$  appears in  $\mathcal{T}$ . Similarly, a basic concept *over*  $\mathcal{T}$  is any basic concept that can be constructed from atomic concepts and roles in  $\mathcal{T}$ .

*Definition 2.* Given a  $DL\text{-Lite}_A$  TBox  $\mathcal{T}$  and two concepts  $C_1$  and  $C_2$  over  $\mathcal{T}$ , the expression  $C_1 \rightarrow C_2$  is called a *functional dependency* (over  $\mathcal{T}$ ). Given an interpretation  $\mathcal{I}$  of  $\mathcal{T}$ , we say that  $C_1 \rightarrow C_2$  is satisfied in  $\mathcal{I}$ , denoted  $\mathcal{I} \models C_1 \rightarrow C_2$ , if there is a role chain  $S = Q_1 \circ \dots \circ Q_n$  over  $\mathcal{T}$ , with  $n > 0$  and such that for each object  $o_1 \in C_1^{\mathcal{I}}$  there is exactly one object  $o_2 \in C_2^{\mathcal{I}}$  such that  $(o_1, o_2) \in S^{\mathcal{I}}$ .

Intuitively, the definition requires that each instance of  $C_1$  determines a unique instance of  $C_2$  by means of some chain of roles in  $\mathcal{T}$ . Note that the inverse of such a role chain corresponds to a path in the path-based identification constraints in [8].

From the above definition, it is immediate to verify that the following properties hold for fds over  $\mathcal{T}$  involving concepts  $C_1, C_2, C_3$ , and for every interpretation  $\mathcal{I}$ :

$$\text{Asserted:} \quad \text{If } \mathcal{I} \models (\text{funct } Q), \text{ then } \mathcal{I} \models \exists Q \rightarrow \exists \text{Inv}(Q). \quad (1)$$

$$\text{Transitivity:} \quad \text{If } \mathcal{I} \models C_1 \rightarrow C_2 \text{ and } \mathcal{I} \models C_2 \rightarrow C_3, \text{ then } \mathcal{I} \models C_1 \rightarrow C_3. \quad (2)$$

$$\text{Left-inclusion:} \quad \text{If } \mathcal{I} \models C_1 \rightarrow C_2 \text{ and } C_3^{\mathcal{I}} \subseteq C_1^{\mathcal{I}}, \text{ then } \mathcal{I} \models C_3 \rightarrow C_2. \quad (3)$$

$$\text{Right-inclusion:} \quad \text{If } \mathcal{I} \models C_1 \rightarrow C_2 \text{ and } C_2^{\mathcal{I}} \subseteq C_3^{\mathcal{I}}, \text{ then } \mathcal{I} \models C_1 \rightarrow C_3. \quad (4)$$

We are now interested in determining when an fd is logically implied by the assertions in the TBox, i.e., the fd is necessarily satisfied in all models of the TBox.

*Definition 3.* Given a  $DL\text{-Lite}_A$  TBox  $\mathcal{T}$ , we say that a fd  $C_1 \rightarrow C_2$  over  $\mathcal{T}$  is logically implied by  $\mathcal{T}$ , denoted  $\mathcal{T} \models C_1 \rightarrow C_2$ , if  $C_1 \rightarrow C_2$  is satisfied in every model of  $\mathcal{T}$ .

In the following, we will restrict the attention to functional dependencies between basic concepts only, since in  $DL\text{-Lite}_A$  negation is used only to assert disjointness. Exploiting the restrictions in the expressive power of  $DL\text{-Lite}_A$ , we can show the following property.

*Proposition 1.* Given a  $DL\text{-Lite}_A$  TBox  $\mathcal{T}$  and two basic concepts  $B_1, B_2$  over  $\mathcal{T}$ , we have that  $\mathcal{T} \models B_1 \rightarrow B_2$  if and only if there is a role chain  $S = Q_1 \circ \dots \circ Q_n$  over  $\mathcal{T}$ , where  $n > 0$ , such that (i)  $\mathcal{T} \models (\text{funct } Q_i)$ , for  $i \in \{1, \dots, n\}$ , (ii)  $\mathcal{T} \models B_1 \sqsubseteq \exists S$ , and (iii)  $\mathcal{T} \models \exists S^- \sqsubseteq B_2$ .

PROOF SKETCH. The “if” direction is a direct consequence of Definitions 2 and 3.

For the “only-if” direction, we observe that properties (ii) and (iii) follow from the canonical model property of the DLs of

the *DL-Lite* family [7], and in particular of *DL-Lite<sub>A</sub>* [24]. Instead, for property (i), we exploit the tree-model property of *DL-Lite<sub>A</sub>*. This property is shared by most DLs [3], and states that if a TBox admits a model, then it admits one that has the structure of a tree (where the nodes of the tree are the elements of the interpretation domain, and the edges are determined by the role instances). In a tree-model, it is ruled out that from a given object  $o_1$  there are two *different* paths labeled with the same roles that lead to the same object  $o_2$ . Hence, a chain of roles is forced to be functional in the TBox  $\mathcal{T}$  only if also all of the component roles are forced to be functional.  $\square$

We can now exploit Proposition 1 to obtain a simple technique that derives pairs of concepts  $B_1, B_2$  such that  $\mathcal{T} \models B_1 \rightarrow B_2$ . The technique is based on turning the properties (1)–(4) above into the following inference rules, which derive new fds from existing ones for a given TBox  $\mathcal{T}$  and for basic concepts  $B_1, B_2$ , and  $B_3$  over  $\mathcal{T}$ :

Asserted: If  $\mathcal{T} \models (\text{funct } Q)$ , then  $\mathcal{T} \models \exists Q \rightarrow \exists \text{Inv}(Q)$ . (5)

Transitivity: If  $\mathcal{T} \models B_1 \rightarrow B_2$  and  $\mathcal{T} \models B_2 \rightarrow B_3$ , then  $\mathcal{T} \models B_1 \rightarrow B_3$ . (6)

Left-inclusion: If  $\mathcal{T} \models B_1 \rightarrow B_2$  and  $\mathcal{T} \models B_3 \sqsubseteq B_1$ , then  $\mathcal{T} \models B_3 \rightarrow B_2$ . (7)

Right-inclusion: If  $\mathcal{T} \models B_1 \rightarrow B_2$  and  $\mathcal{T} \models B_2 \sqsubseteq B_3$ , then  $\mathcal{T} \models B_1 \rightarrow B_3$ . (8)

We consider these rules to be applied exhaustively to all basic concepts over  $\mathcal{T}$ . Soundness of the rules follows directly from the corresponding properties above, while completeness is a consequence of Proposition 1. Moreover, since the number of basic concepts over  $\mathcal{T}$  is finite, rule application clearly terminates.

We observe that the “left-inclusion” and “right-inclusion” rules propagate fds according to the TBox inclusion assertions, and as such they provide an interaction between functional and inclusion dependencies. In general, implication is undecidable when combining functional and inclusion dependencies [1], our setting is much simpler, since we only consider unary inclusion<sup>6</sup> and functional dependencies [13]. Note also that there is no counterpart of the augmentation rule for fd implication in the relational setting, since we deal only with unary functional dependencies.

#### 4. DISCOVERING FDS IN *DL-Lite<sub>A</sub>*

In this section, we propose an algorithm to discover all the fd’s logically implied by a *DL-Lite<sub>A</sub>* TBox  $\mathcal{T}$ , and which exploits the reasoning capabilities of a *DL-Lite<sub>A</sub>* reasoner. Our algorithm starts from the asserted fds (see inference rule (5)), and then computes the closure of the asserted fds w.r.t. the remaining rules. We recall that the asserted fds are simply  $\exists Q \rightarrow \exists \text{Inv}(Q)$ , for each functional role  $Q$  in the TBox.

The closure of the asserted fd’s is computed as follows. First, we identify the sets  $\mathcal{B}_d$  and  $\mathcal{B}_r$  of all basic concepts that appear respectively in the domain and range of a functional basic role. To do so, we scan all functional basic roles, and for each such role  $Q$  and each basic concept  $B$  over  $\mathcal{T}$ , if  $\mathcal{T} \models B \sqsubseteq \exists Q$  then we add  $B$  to  $\mathcal{B}_d$ , and if  $\mathcal{T} \models B \sqsubseteq \exists \text{Inv}(Q)$  then we add  $B$  to  $\mathcal{B}_r$ .

Then, for each pair of basic concepts  $B_d \in \mathcal{B}_d$  and  $B_r \in \mathcal{B}_r$ , we need to check whether  $\mathcal{T} \models B_d \sqsubseteq \exists S.B_r$ , for some chain  $S$  of functional basic roles<sup>7</sup>. To perform such a check, we have

<sup>6</sup>Note that, in *DL-Lite*, role inclusions are restricted so as not to interact with functionality.

<sup>7</sup>The concept  $\exists S.B_r$  is called a qualified existential and is interpreted as  $\{o \mid \exists o'.(o, o') \in S^{\mathcal{I}} \wedge o' \in B_r^{\mathcal{I}}\}$ . It is not a *DL-Lite<sub>A</sub>* concept.

to face the difficulty that in principle we have to try all possible lengths  $n$  of the chain  $S$ , and all the possible ways of composing it by means of functional basic roles. To tackle the latter issue, we introduce a new atomic role  $U$  in  $\mathcal{T}$ , and for each basic role  $Q$  such that (funct  $Q$ ) is in  $\mathcal{T}$ , we add to  $\mathcal{T}$  the assertion  $Q \sqsubseteq U^8$ . Hence,  $U$  acts as a super-role of all functional basic roles in  $\mathcal{T}$  and it is sufficient to consider  $S$  as a chain of  $U$   $n$  times with itself, for suitable values of  $n$ . We then iterate over  $n$  until we have tried a sufficiently large value (see below).

In *DL-Lite<sub>A</sub>* we cannot directly check the logical implication  $\mathcal{T} \models B_d \sqsubseteq \exists S.B_r$ , with  $S = U \circ \dots \circ U$  (for some fixed length  $n$  of the chain). However, we can easily encode such a check into the problem of computing the certain answers to the CQ

$$Q_{B_d, B_r}^n() \leftarrow B_d(a), U(a, x_1), U(x_1, x_2), \dots, U(x_{n-1}, x_n), B_r(x_n)$$

over the ABox constituted only by the assertion  $B_d(a)$ . Indeed, since the only fact in the ABox is one involving  $B_d$ , it is not possible to satisfy the atoms  $U(x, x')$  and  $B_r(x_n)$  with facts in the ABox. Hence, the only case in which the answer to the query could anyway be positive, is when the whole body of  $Q_{B_d, B_r}^n$  can be rewritten to just  $B_d(a)$  [7]. And this is precisely the case when  $\mathcal{T} \models B_d \sqsubseteq \exists S.B_r$ . Notice that we are taking advantage of the query rewriting technique for *DL-Lite<sub>A</sub>*, which exploits the knowledge contained in the TBox of the ontology to actually compute the right-inclusion and left-inclusion inference rules with the *DL-Lite<sub>A</sub>* reasoner.

The question that we still need to address is which is the maximum bound for the length  $n$  of the role chain  $S$ . If the ontology does not contain functional cycles, we should stop when no new answer is retrieved. However, it is not uncommon to find functional cycles in a real world ontology. In this case, we should stop looking for functional dependencies originating at a concept  $B_d$  when (i) for a given length no results are provided, or (ii) no new concepts are proposed with regard to previous iterations. Intuitively, the reason is that in *DL-Lite* all the roles involved in a functional path must be functional as well, and hence at each step we must get, at least, one new concept of the longest path. Otherwise we are looping in a cycle.

More precisely, let  $B_d$  be the concept from where we start looking for functional dependencies and  $\mathcal{D}_i$  the set of concepts that we have already identified up to iteration  $i$ . Let  $B_r$  be a concept functionally dependent on  $B_d$  and not yet identified.

- If  $B_d$  functionally identifies  $B_r$ , then there must be a role chain  $S'$  that connects  $\mathcal{D}_i$  and  $B_r$ . Let  $\mathcal{D}_?$  be the concepts along  $S'$ . Note that  $\mathcal{D}_i$  and  $\mathcal{D}_?$  are disjoint, since the  $\mathcal{D}_i$  contains concepts already visited, while  $\mathcal{D}_?$  not.
- At least, one concept of  $\mathcal{D}_i$  and one concept of  $\mathcal{D}_?$  must be directly related. Let’s call them  $B_i$  and  $B_?$ , respectively.
- If along the  $i + 1$ -th iteration we do not identify any new concept, then,  $B_? \in \mathcal{D}_i$ , which contradicts our initial assumption that  $\mathcal{D}_i$  and  $\mathcal{D}_?$  are disjoint.

An upper bound for the maximum number of queries we will have to pose is  $\Theta(n \cdot |\mathcal{B}_d| \cdot |\mathcal{B}_r|)$ , where  $n$  is the length of the maximum chain of functional roles. However, this is an upper bound not reachable in practice because those concepts that do not get any new solution for paths of length  $i$ , are not queried in the next iterations. In most cases (considering ontologies in real applications), most of the concepts will end with  $n$  being rather small, as discussed in Section 6.1. Furthermore, if in a previous iteration we

<sup>8</sup>Note that this is compatible with the conditions in Definition 1.

have shown that  $\mathcal{T} \models B_d \rightarrow B_r$ , then, this pair will not be checked again in next iterations.

We note that the computational complexity of the rewriting algorithm of  $DL\text{-Lite}_A$  is exponential in the size of the query. However, this turns out to be manageable for real ontologies, given that the number of times we have to concatenate role  $U$  is relatively small, cf. Section 6.1.

## 5. $DL\text{-Lite}_A$ ONTOLOGIES FROM CONCEPTUAL SCHEMAS

The fds as introduced in Section 3 are conceived for arbitrary  $DL\text{-Lite}_A$  ontologies. Nevertheless, there are some interesting additional considerations to be made regarding  $DL\text{-Lite}_A$  ontologies derived from conceptual schemas. Consider the UML diagram depicted in Figure 1 and let  $\mathcal{T}_{eu}$  be the corresponding  $DL\text{-Lite}_A$  TBox. Specifically, the `hasAssigned` association results in the following assertions:

$$\begin{aligned} \exists \text{hasAssigned} &\sqsubseteq \text{RentalAgreement} && (\text{funcnt } \text{hasAssigned}) \\ \exists \text{hasAssigned}^- &\sqsubseteq \text{Assignment} && (\text{funcnt } \text{hasAssigned}^-) \\ \text{Assignment} &\sqsubseteq \exists \text{hasAssigned}^- && \end{aligned}$$

According to Definition 3, we have that  $\mathcal{T}_{eu} \models \text{Assignment} \rightarrow \text{RentalAgreement}$ , (but  $\mathcal{T}_{eu} \not\models \text{RentalAgreement} \rightarrow \text{Assignment}$ ), since `RentalAgreement` does not have a mandatory participation in `hasAssertion`. As discussed in Section 3.1, the mandatory participation is needed in arbitrary  $DL\text{-Lite}_A$  ontologies to avoid discovering meaningless functional dependencies. For instance, consider the following TBox  $\mathcal{T}$ :

$$\exists P_1 \sqsubseteq A_1, \exists P_1^- \sqsubseteq A_2, (\text{funcnt } P_1), \exists P_2 \sqsubseteq A_3 \exists P_2^- \sqsubseteq A_4 (\text{funcnt } P_2)$$

Without requiring the mandatory participation we would have that  $\mathcal{T} \models A_1 \rightarrow A_4$ . Indeed, both  $P_1$  and  $P_2$  are functional, and therefore, in every model of  $\mathcal{T}$ , every instance of  $A_1$  is connected to at most one instance of  $A_4$  via the role chain  $P_1 \circ P_2$ .

However this scenario cannot happen in ontologies derived from conceptual schemas. In an UML-CD (or ER schema) two classes are supposed to be disjoint unless they are related by a generalization relationship and furthermore, strict role-typing is assumed (i.e., exactly the opposite assumptions to those in arbitrary  $DL\text{-Lite}_A$  ontologies). Hence, when translating UML-CDs to  $DL\text{-Lite}_A$  it makes sense to identify functional dependencies from non mandatory relationships. With this aim, we redefine the functional property definition presented in Section 3.1 for  $DL\text{-Lite}_A$  ontologies derived from conceptual schemas:

*Definition 4.* Given a  $DL\text{-Lite}_A$  TBox  $\mathcal{T}$ , an atomic role  $P$  in  $\mathcal{T}$  is *strict role-typed* in  $\mathcal{T}$  if there is a single atomic concept  $A_1$  such that  $\exists P \sqsubseteq A_1$  is in  $\mathcal{T}$ , and a single atomic concept  $A_2$  such  $\exists P^- \sqsubseteq A_2$  is in  $\mathcal{T}$ . The concepts  $A_1$  and  $A_2$  are called respectively the *domain* and *range* of  $P$ . A  $DL\text{-Lite}_A$  TBox  $\mathcal{T}_c$  is called  *$DL\text{-Lite}_A$  conceptual schema* if each atomic role  $P$  is strict role-typed in  $\mathcal{T}$  and for each pair of atomic concepts  $A_1, A_2$ , either  $A_1$  and  $A_2$  are disjoint (i.e.,  $\mathcal{T}_c \models A_1 \sqsubseteq \neg A_2$ ) or  $A_1$  and  $A_2$  participate in the same concept taxonomy (i.e., there is an atomic concept  $A$  such that  $\mathcal{T}_c \models A_1 \sqsubseteq A$  and  $\mathcal{T}_c \models A_2 \sqsubseteq A$ ).<sup>9</sup>

*Definition 5.* Given a  $DL\text{-Lite}_A$  conceptual schema  $\mathcal{T}_c$ , two basic concepts  $B_1$  and  $B_2$  over  $\mathcal{T}_c$ , and an interpretation  $\mathcal{I}$  of  $\mathcal{T}$ , we say that  $\mathcal{I} \models B_1 \rightarrow B_2$  if there is a chain  $S = Q_1 \circ \dots \circ Q_n$ , with  $n > 0$ , of roles that are strict role-typed in  $\mathcal{T}$ , where  $B_1$  is

<sup>9</sup>Note that the concept  $A$  may coincide with  $A_1$  or  $A_2$ .

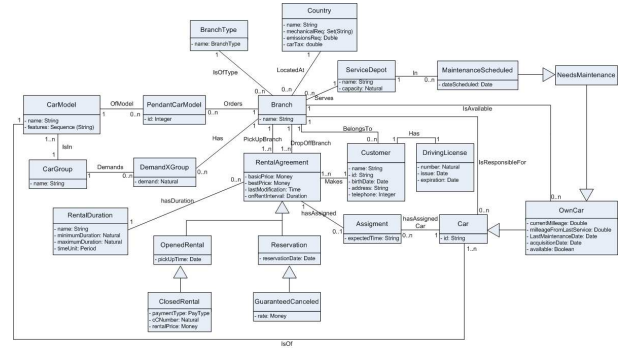


Figure 1: The EU-Car Rental Case Study

the domain of  $Q_1$ ,  $B_2$  is the range of  $Q_n$ , and such that for each object  $o_1 \in \exists Q_1^{\mathcal{I}}$  there is exactly one object  $o_2 \in C_2^{\mathcal{I}}$  such that  $(o_1, o_2) \in S^{\mathcal{I}}$ .

Roughly speaking, we may relax the mandatory participation of  $B_1$  in  $S$  thanks to the implicit constraints we may find in a  $DL\text{-Lite}$  conceptual schema.

We can take advantage of the algorithm presented in Section 4 to discover functional dependencies over  $DL\text{-Lite}_A$  conceptual schemas by adding the following two assertions for each functional role  $P$  with domain  $A_1$  and range  $A_2$ :

$$A_1 \sqsubseteq \exists P \quad A_2 \sqsubseteq \exists P^-$$

Indeed, we are adding a mandatory participation for the role to its domain and range. In terms of UML-CDs, we are modifying the cardinality of the relationship and making it mandatory. With this trick we fulfill Definition 2 of fd and despite this change, the semantics with regard to fd's will not change and the results we get are sound. Notice that these assertions are only needed while discovering functional dependencies and they have to be retracted once the algorithm has finished.

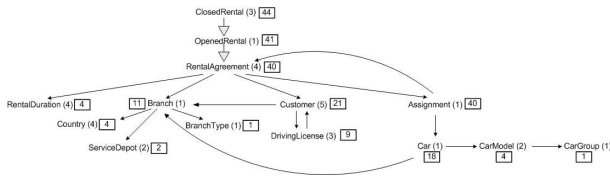
This trick cannot be applied for arbitrary  $DL\text{-Lite}$  ontologies. In an arbitrary  $DL\text{-Lite}_A$  ontology disjointness of concepts cannot be assumed and therefore, adding the domain and range assertion we would modify the semantics of the model also with respect to fds. As a consequence, we could identify false fds.

## 6. CASE STUDY

In this section we show results got after applying our algorithm over a real case study. Consider the conceptual diagram depicted in Figure 1. This diagram corresponds to a piece of the EU-Car Rental case study. It presents a car rental company with branches in several countries which provides typical rental services. This conceptual schema collects information about cars, branches, rental agreements, assignments, etc. A detailed specification of this case study may be found in [15].

Since we want to automate the fd's computation, we first need to transform the UML diagram into a  $DL\text{-Lite}_A$  ontology as explained in Section 2. Once it is done, we can use AMDO's patterns to discover the multidimensional concepts as presented in [26].

It is remarkable the relevance of automating this process as in this small example we are able to discover 426 functional dependencies. The reader will notice, though, that not all of them will be shown to the user but internally handled by AMDO. For instance, many functional dependencies identified are datatypes. Moreover, given a concept  $C$ , its set of functional dependencies may be depicted as a tree with  $C$  as root node, and most of these trees will



**Figure 2: A compacted view of the functional dependencies in the EU-Car Rental case study**

overlap. As an example, the fd tree of branch will be a subtree within the rentalagreement fd tree. Figure 2 takes advantage of these properties to show the functional dependencies of closedrental (the concept with most fds in our example) and the functional dependencies of all those concepts in its tree; giving rise to a directed graph. This figure must be read as follows. Starting from a concept, following all the directed arrows we may get to all its functional dependencies (for instance, branch and its subtree are functional dependencies of customer, and the whole three of rentalagreement is a subtree of assignment. Closedrental and openedrental have two special arrows that stand for inheritance; thus, they get all the functional dependencies their parents have). Numbers in brackets represent the number of functional datatypes that concept has (since by definition, datatypes do not have dependencies we can overlook them at first sight and only show them if the user asks for). The squared number stands for the functional dependencies that concept has. Notice that we cannot compute the number of fds of a concept by adding the squared numbers of its sons, as some fds are shared between them. Finally, the reader must notice that if rentalduration has 4 functional dependencies (not shared with any other concept), rentalagreement will get 5 fds from this edge (4 fds plus rentalduration itself).

By applying AMDO, each domain concept will be evaluated as a potential fact. In short, the fd's of each concept are divided in two main groups: those being potential dimensional concepts, and those being potential measures. The more measures and dimensions a concept has, the better will be ranked as a promising fact. For instance, ClosedRental (42 fd's), OpenedRental (41), RentalAgreement (40) and Assignment (40) would be considered promising subjects of study, which makes sense according to their semantics. AMDO applies additional design patterns for eventually shape the multidimensional conceptual schema but in essence, it uses fd's for identifying dimensional concepts (and also to give rise to the dimension hierarchies) and measures.

## 6.1 Statistics over the Case Study

For the sake of comprehension, we have discussed till now only a piece of the EU Car-Rental case study. We consider now the full EU Car-Rental case study of which we computed the closure of functional dependencies by applying the algorithm presented in Section 4. This case study has 65 concepts and 170 roles (30 of which are subsumption assertions between classes).

A total of 1908 functional dependencies were found. The total computation time was 2.332 seconds from which, 0.080 seconds were used by the reasoner to classify the ontology, 0.006 seconds were required to query for candidate domains and ranges for the functional paths (i.e., the  $\mathcal{B}_d$  and  $\mathcal{B}_r$  sets presented in Section 4, and the remaining time (2.246 seconds) was used to compute the functional dependencies.

To run these tests we used the FaCT++ reasoner. We note that FaCT++ doesn't support answering conjunctive queries, As a workaround, we had to devise a subsumption verification query

which was *true* iff the CQ of Section 4 is non-empty, and *false* otherwise. The query which complies with this specification is  $(B_1 \sqsubseteq \exists U. \exists U. \dots B_2)?$  were  $B_1$  corresponds to the current domain to be tested, the number of nested  $U$ 's corresponds to the number of  $U$  atoms in the original CQ and  $B_2$  corresponds to the range of the functional path to be tested.

Furthermore, we also computed the functional paths (i.e., the composition of roles) that verified the query shown above. Notice this is relevant because in the multidimensional model each functional path between two concepts must give rise to different perspectives of analysis (i.e., aggregation paths). In order to do this, we sent additional queries to the reasoner, whenever we had verified the existence of a path of length  $n$ . In these queries we replaced the  $n$ 'th occurrence of role  $U$  in the qualified existential chain with each of the sub-roles of  $U$ . In this case, 41.039 seconds were spent pinning-down the specific roles which triggered the existence of these paths.

The computer used in this test was equipped with an Intel Core 2 Duo 2.16 GHz processor, 3 GB of RAM. With respect to software, we used the 64 bit version of the Java Runtime Environment 1.6 and the 64 bit version of the FaCT++ runtime binaries.

## 7. RELATED WORK

On the one hand, previous approaches for discovering functional dependencies were devised for relational databases and they address this task either at the logical or physical level. Addressing this task at the logical level entails that results got are tied to the design decisions made when devising the system. A logical schema may lack semantics regarding a conceptual schema. Logical design decisions such as overlooking foreign keys in the relational schema or denormalizing data (i.e., collapsing relations) directly impact on the quality of results got. Some approaches try to overcome the logical schema lack of semantics with additional semantics or assumptions (for instance, [19, 29]) but most of them address this task directly at the physical level [14, 21, 23].

On the other hand, there have been other works proposing to model the data warehouse from a conceptual formalization of the domain. However, most of them must be carried out manually since they present a step-by-step guide to be followed by a data warehousing expert (see [27] for further details). Only a few of them automate this process somehow but the degree of automation achieved is rather low. Specifically, these approaches consist of a detailed requirement elicitation stage (to be performed manually) and an automated analysis of the data sources. Later, both stages are put in common, conciliating in this way the data sources and requirements (for instance, see [5, 16, 17]). In these methods the requirement elicitation stage leads the process and main design decisions are captured in this step whereas the analysis of the data sources is rather superficial (i.e., design patterns described do not exploit the whole semantics of the multidimensional concepts). Indeed, the analysis of the data sources can be thought as a complementary task to the requirement elicitation processes described in these approaches. For instance, facts are mostly identified manually from requirements and the automation of the process is reduced to discover to-one relationships from each fact identified. To our knowledge, there is only one approach working at the conceptual level and following a similar framework to AMDO [28]. This work presents a method to design the data warehouse from ER diagrams. and it achieves a fair degree of automation. Dimension hierarchies and measures, however, are overlooked. For a detailed related work on multidimensional design methods (and the benefits and main contributions of AMDO regarding other approaches) we address the reader to [27, 26].

## 8. CONCLUSIONS

In this paper we propose to redesign the core of the AMDO method [26] and take advantage of the well-known reasoning services provided by DL languages. Thus, our algorithms do exploit now the inference capabilities of *DL-Lite* for multidimensional design purposes. We have also shown the feasibility of our method in practice by means of the EU-Car Rental case study.

## 9. ACKNOWLEDGEMENTS

This work has been partly supported by the Ministerio de Educacion y Ciencia and FEDER under project TIN2008-03863.

## 10. REFERENCES

- [1] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison Wesley, 1995.
- [2] A. Acciari, D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, M. Palmieri, and R. Rosati. QUONTO: Querying ONTOlogies. In *Proc. of AAAI 2005*, pages 1670–1671, 2005.
- [3] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press, 2003.
- [4] D. Berardi, D. Calvanese, and G. De Giacomo. Reasoning on UML class diagrams. *Artificial Intelligence*, 168(1–2):70–118, 2005.
- [5] A. Bonifati, F. Cattaneo, S. Ceri, A. Fuggetta, and S. Paraboschi. Designing Data Marts for Data Warehouses. *ACM Trans. Softw. Eng. Methodol.*, 10(4):452–483, 2001.
- [6] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Data complexity of query answering in description logics. In *Proc. of KR 2006*, pages 260–270, 2006.
- [7] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Tractable reasoning and efficient query answering in description logics: The *DL-Lite* family. *J. of Automated Reasoning*, 39(3):385–429, 2007.
- [8] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Path-based identification constraints in description logics. In *Proc. of KR 2008*, pages 231–241, 2008.
- [9] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Conceptual modeling for data integration. In A. Borgida, V. Chaudhri, P. Giorgini, and E. Yu, editors, *Essays in Honour of John Mylopoulos*, volume 5600 of *LNCIS*, pages 173–197. Springer, 2009.
- [10] D. Calvanese, G. De Giacomo, and M. Lenzerini. Identification constraints and functional dependencies in description logics. In *Proc. of IJCAI 2001*, pages 155–160, 2001.
- [11] D. Calvanese, M. Lenzerini, and D. Nardi. Description logics for conceptual data modeling. In J. Chomicki and G. Saake, editors, *Logics for Databases and Information Systems*, pages 229–264. Kluwer Academic Publishers, 1998.
- [12] P. P. Chen. The Entity-Relationship model: Toward a unified view of data. *ACM Trans. on Database Systems*, 1(1):9–36, Mar. 1976.
- [13] S. S. Cosmadakis, P. C. Kanellakis, and M. Vardi. Polynomial-time implication problems for unary inclusion dependencies. *J. of the ACM*, 37(1):15–46, Jan. 1990.
- [14] P. A. Flach and I. Savnik. Database dependency discovery: A machine learning approach. *AI Communications*, 12(3):139–160, 1999.
- [15] L. Frías, A. Queralt, and A. Olivé. EU-Rent car rentals specification. Technical report, Departament de Llenguatges i Sistemes Informatics, 2003.
- [16] P. Giorgini, S. Rizzi, and M. Garzetti. Goal-oriented Requirement Analysis for Data Warehouse Design. In *Proc. of DOLAP 2005*, pages 47–56. ACM Press, 2005.
- [17] M. Golfarelli, D. Maio, and S. Rizzi. The Dimensional Fact Model: A Conceptual Model for Data Warehouses. *Int. Journal of Cooperative Information Systems*, 7(2-3):215–247, 1998.
- [18] V. Haarslev and R. Möller. RACER system description. In *Proc. of IJCAR 2001*, volume 2083 of *LNAI*, pages 701–705. Springer, 2001.
- [19] J.-L. Hainaut, M. Chandelon, C. Tonneau, and M. Joris. Contribution to a theory of database reverse engineering. In *Proc. of the 1st Working Conf. on Reverse Engineering*, pages 161–170. IEEE, 1993.
- [20] I. Horrocks. Using an expressive description logic: FaCT or fiction? In *Proc. of KR’98*, pages 636–647, 1998.
- [21] M. Kantola, H. Mannila, K.-J. Räihä, and H. Siirtola. Discovering functional and inclusion dependencies in relational databases. *Int. J. of Intelligent Systems*, 7:591–607, 1992.
- [22] R. Kimball, L. Reeves, W. Thornthwaite, and M. Ross. *The Data Warehouse Lifecycle Toolkit: Expert Methods for Designing, Developing and Deploying Data Warehouses*. John Wiley, 1998.
- [23] W. M. Lim. Discovery of constraints from data for information system reverse engineering. In *Proc. of the 2nd Australian Software Engineering Conf.*, pages 39–48. IEEE, 1997.
- [24] A. Poggi, D. Lembo, D. Calvanese, G. De Giacomo, M. Lenzerini, and R. Rosati. Linking data to ontologies. *J. on Data Semantics*, X:133–173, 2008.
- [25] A. Poggi, M. Rodriguez, and M. Ruzzi. Ontology-based database access with DIG-Mastro and the OBDA Plugin for Protégé. In K. Clark and P. F. Patel-Schneider, editors, *Proc. of OWLED 2008 DC*, 2008.
- [26] O. Romero and A. Abelló. Automating multidimensional design from ontologies. In *Proc. of DOLAP 2007*, pages 1–8, 2007.
- [27] O. Romero and A. Abelló. A survey of multidimensional modeling methodologies. *Int. J. of Data Warehousing and Mining*, 5(2):1–23, 2009.
- [28] I.-Y. Song, R. Khare, and B. Dai. SAMSTAR: A Semi-Automated Lexical Method for Generating STAR Schemas from an ER Diagram. In *Proc. of DOLAP 2007*, pages 9–16. ACM, 2007.
- [29] H. B. K. Tan and Y. Zhao. Automated elicitation of functional dependencies from source codes of database transactions. *Information & Software Technology*, 46(2):109–117, 2004.
- [30] D. Toman and G. E. Weddell. On the interaction between inverse features and path-functional dependencies in description logics. In *Proc. of IJCAI 2005*, pages 603–608, 2005.
- [31] D. Toman and G. E. Weddell. On keys and functional dependencies as first-class citizens in description logics. *J. of Automated Reasoning*, 40(2–3):117–132, 2008.
- [32] M. Y. Vardi. The complexity of relational query languages. In *Proc. of STOC’82*, pages 137–146, 1982.