Capturing Relational Schemas and Functional Dependencies in RDFS

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

Wolfgang Fischl, Reinhard Pichler, Emanuel Sallinger, Mantas Šimkus
Institute of Information Systems
Vienna Univ. of Technology, Austria
{wfischl, pichler, sallinger, simkus}@dbai.tuwien.ac.at

Abstract

Mapping relational data to RDF is an important task for the development of the Semantic Web. To this end, the W3C has recently released a Recommendation for the so-called direct mapping of relational data to RDF. In this work, we propose an enrichment of the direct mapping to make it more faithful by transferring also semantic information present in the relational schema from the relational world to the RDF world. We thus introduce expressive identification constraints to capture functional dependencies and define an RDF Normal Form, which precisely captures the classical Boyce-Codd Normal Form of relational schemas.

Introduction

Over the past years, we have been witnessing an enormous growth of the Semantic Web through initiatives like Open Linked Data (Bizer, Heath, and Berners-Lee 2009) and Open Government Data (HM Government 2014; US Government 2014). As was noted by He et al. (2007) and Madhavan et al. (2009), to a large extent, the data accessible on the web still originates from relational databases. The World Wide Web Consortium (W3C) has thus recognized a standardized mapping of relational data to the Semantic Web data format RDF as an important task. For this purpose, the so-called *direct mapping* has been recently released as a W3C Recommendation (Arenas et al. 2012a). Note that the direct mapping of relational data to RDF does not transfer the semantic information present in the relational schema to the RDF graph.

The goal of this paper is to study an enrichment of the direct mapping to make it more faithful by transferring also important semantic information, e.g., functional dependencies, from relational to RDF data. As a first attempt to transfer such information to RDF, Sequeda, Arenas, and Miranker (2012) have proposed to extend the mapping by the use of OWL vocabulary. They have succeeded in transferring information such as primary and foreign keys to an RDF graph with OWL. They also proved several important properties of their mapping, such as query-preservation, i.e., SQL queries over the relational data can be rewritten to equivalent SPARQL queries over the mapped data.

However, if the RDF graph resulting from such a mapping is later changed (through update, delete, or insert operations), then the correspondence between the relational and the RDF data may get lost. We therefore propose a further extension of the direct mapping that uses *DL-Lite*$_{RDFS}$ (Arenas et al. 2012b) – extended with disjointness – as basis. *DL-Lite*$_{RDFS}$ is a variant of *DL-Lite*$_{\mathcal{A}}$ (Calvanese et al. 2006) and captures the Description Logic (DL) fragment of RDFS (Brickley and Guha 2004). While this DL is simple and allows for efficient reasoning, it naturally captures conceptual modeling constructs, and hence can express dependencies over RDF graphs. We present a mapping that produces an RDF graph (corresponding to the direct mapping) and a DL TBox constraining the RDF graph (making use of DL-features such as functionality and disjointness). We thus keep good properties of the mapping proposed by Sequeda, Arenas, and Miranker (2012) – such as query preservation. In addition, we are able to prove that this mapping satisfies a desired one-to-one correspondence between relational databases and *legal* RDF graphs (i.e., RDF graphs satisfying the constraints of the TBox).

An important part of the semantic information present in relational schemas is due to various forms of *constraints* imposed on the relational data, such as key dependencies, functional dependencies, foreign key dependencies, inclusion dependencies, etc. The focus of our work is on *functional dependencies* (fds). Intuitively, for a relation R, an fd $\{A_1, \ldots, A_n\} \rightarrow_R A_0$ expresses that if two tuples of R agree on all attributes $A_1, \ldots, A_n$, they also have to agree on attribute A_0. We will see how to extend this notion to the DL and RDF setting through the use of paths. Fds are a crucial building block in database design (Mannila and Räihä 1992). Indeed, they form the basis of the definition of normal forms to eliminate redundancies and to avoid update anomalies. Sequeda, Arenas, and Miranker (2012) have extended the direct mapping by constraints such as primary and foreign keys,

while fds have not been in the scope of their work. Calvanese, De Giacomo, and Lenzerini (2001) enrich DLs with fds and with a generalization of DL functionality assertions, called identification constraints (ids). The latter are extended by Calvanese et al. (2008) to path-based ids (pids)[1]. Toman and Weddell (2008) introduce path-based fds in DLs (containing the expressive DL $\mathcal{ALCN}$); however, they are not interested in the correspondence to relational data and thus do not take the direct mapping into account (Toman and Weddell 2005).

As we will show, even the more expressive pids fail to capture fds for the direct mapping of relational data to RDF. We shall therefore introduce a further extension of such ids, which we call *tree-based ids* (tids). With this new class of ids, we shall restore the desired one-to-one relationship between *legal* databases (i.e., databases satisfying a given set of fds) and *legal* RDF graphs.

As mentioned above, the usual strategy to avoid update anomalies of relational data is to introduce normal forms of relational schemas. Several normal forms of relational schemas have been proposed since the very beginnings of relational database research, see, e.g., (Codd 1971a; 1971b). One of the most important and most intuitive normal forms of relational schemas is *Boyce-Codd Normal Form* (BCNF). A relational schema is in BCNF if the following condition holds: Whenever there is an fd from a subset S of the attributes to some attribute $A \notin S$, then S is a super-key, i.e., there is an fd from S to *every* attribute of this schema. Our goal is to transfer the favorable properties of BCNF to the RDF world. To this end, we first analyze how update anomalies can arise in the presence of tids. We identify several paths (stemming from the same tid) identifying the same object as a crucial source of redundancy and hence of update anomalies. This observation inspires the definition of an *RDF Normal Form* (RNF) to avoid such anomalies. Returning to the direct mapping, we prove that a relational schema is in BCNF if and only if the corresponding TBox with its constraints guarantees RNF. As a kind of surprise, it turns out that – for relational schemas in BCNF – the additional expressive power of tree-based ids is not needed to capture fds. Indeed, under the restriction to BCNF, the original form of ids introduced by Calvanese, De Giacomo, and Lenzerini (2001) are expressive enough to transfer fds from the relational schema to the DL TBox.

The remainder of this paper is structured as follows: after recalling some basic notions and results, we introduce our extended direct mapping of relational data to RDF. To capture also fds, we then extend path-based ids to tree-based ids. Finally, we define RNF and prove important properties of this normal form. We conclude with a brief discussion of our results and an outlook to future work.

Preliminaries

Databases. Let Δ_r and Δ_v be countably infinite, disjoint sets of *relation symbols* and *values*, respectively. Each $R \in \Delta_r$ has an associated positive integer value $ar(R)$, called *the arity of* R. An *atom* is an expression of the form $R(c_1, \ldots, c_{ar(R)})$, where $R \in \Delta_r$ and $c_1, \ldots, c_{ar(R)} \in \Delta_v$. A *database (DB)* I is a*finite* set of atoms. Let $dom(I)$ denote the set of values that occur in the atoms of I. We let $Inst(R)$ be the set of all DBs I with $I \subseteq \{R(t) \mid t \in (\Delta_v)^{ar(R)}\}$, i.e., all DBs over the relation symbol R. We let $Inst(\Delta_r)$ be the set of all DBs over the relation symbols in Δ_r.

Functional Dependencies. A *functional dependency* (fd) is an expression of the form $X \rightarrow_R Y$, where $R \in \Delta_r$ and $X, Y \subseteq \{1, \ldots, ar(R)\}$. A DB $\mathcal{I}$ *satisfies* $X \rightarrow_R Y$, denoted as $\mathcal{I} \models X \rightarrow_R Y$, if for all $\{R(c_1, \ldots, c_l), R(c'_1, \ldots, c'_l)\} \subseteq \mathcal{I}$, it holds that $c_i = c'_i$ for all $i \in X$ implies $c_j = c'_j$ for all $j \in Y$. Given a set Σ of fds, we write $\mathcal{I} \models \Sigma$, if $\mathcal{I} \models \varphi$ for all $\varphi \in \Sigma$. Given an fd φ, we write $\Sigma \models \varphi$ if for all $\mathcal{I} \models \Sigma$, it holds that $\mathcal{I} \models \varphi$. We say that Σ is in *Boyce-Codd Normal Form* (BCNF) if for every $X \rightarrow_R Y$ with $\Sigma \models X \rightarrow_R Y$, it holds that $Y \subseteq X$ or $\Sigma \models X \rightarrow_R \{1, \ldots, ar(R)\}$.

DL-Lite$_{RDFS,pid}$**.** Unary (resp., binary) relation symbols are also called *concept names* (resp., *role names*). The role name id is called the *identity role*. For simplicity, we use the following abstraction for RDF triples: A DB I is called an *RDF graph (RG)* if it only consists of atoms with concept names or role names as relation symbols, i.e., all atoms in I are unary or binary. RGs can be easily converted to RDF triples. We use calligraphic letters $\mathcal{I}, \mathcal{J}$ to denote RGs.

A *basic role* is either a role name or an expression of the form R^-, where R is a role name. A *basic concept* is either a concept name or an expression of the form $\exists P$, where P is a basic role. We write $R^-(c, d) \in \mathcal{I}$ if $R(d, c) \in \mathcal{I}$, and $\exists P(c) \in \mathcal{I}$ if there exists d with $P(c, d) \in \mathcal{I}$. In the following let A denote a concept name, B, B_1, B_2 basic concepts and P, P_1, P_2 basic roles. A *positive concept inclusion* is an expression of the form $B \sqsubseteq A$. We write $\mathcal{I} \models B \sqsubseteq A$ if $B(c) \in \mathcal{I}$ implies $A(c) \in \mathcal{I}$ for every value c. A *positive role inclusion* has the form $P_1 \sqsubseteq P_2$. We write $\mathcal{I} \models P_1 \sqsubseteq P_2$ if $P_1(c, d) \in \mathcal{I}$ implies $P_2(c, d) \in \mathcal{I}$ for all values c, d. A *negative concept inclusion* is an expression of the form $B_1 \sqsubseteq \neg B_2$. We write $\mathcal{I} \models B_1 \sqsubseteq \neg B_2$ if $B(c) \in \mathcal{I}$ implies $B_2(c) \notin \mathcal{I}$ for every value c. A *negative role inclusion* has the form $P_1 \sqsubseteq \neg P_2$. We write $\mathcal{I} \models P_1 \sqsubseteq \neg P_2$ if $P_1(c, d) \in \mathcal{I}$ implies $P_2(c, d) \notin \mathcal{I}$ for all values c, d. A *functionality assertion* has the form $(\mathsf{funct}\ P)$. We write $\mathcal{I} \models (\mathsf{funct}\ P)$ whenever $P(a, b) \in \mathcal{I}$ and $P(a, c) \in \mathcal{I}$ implies $b = c$.

A *test role* is an expression of the form $B?$. We write $B?(c, d) \in \mathcal{I}$ in case $c = d$ and $B(c) \in \mathcal{I}$. A *path* is a (possibly empty) word $\pi = \sigma_1 \cdots \sigma_m$, where each σ_i is a basic or test role. A *path-based identification constraint* (pid) (Calvanese et al. 2008) has the form $B?(\pi_1, \ldots, \pi_n)$, where $\pi_1, \ldots, \pi_n$ are paths. The function $(\cdot)^{\mathcal{I}}$ assigns to each path π a binary relation based on the RG $\mathcal{I}$ as follows:

- $\varepsilon^{\mathcal{I}} = \{(e, e) \mid e \in \Delta_v\}$
- $(\sigma \cdot \pi)^{\mathcal{I}} = \{(e_1, e_2) \mid \exists (e'_1, e'_2) \in \pi^{\mathcal{I}}$ s.t. $\sigma(e_1, e'_1) \in \mathcal{I} \wedge \sigma(e_2, e'_2) \in \mathcal{I}\}$,

[1] We explore the connection between pids and fds in the Section "Capturing Functional Dependencies".

Figure 1: An RG $\mathcal{I}$ with data on buildings.

We write $\mathcal{I} \models B?(\pi_1, \ldots, \pi_n)$, if $(e_1, e_2) \in \bigcap_{i=1}^{n} (B? \cdot \pi_i)^{\mathcal{I}}$ implies $e_1 = e_2$. We often write π to denote a pid, e.g., $\pi = B?(\pi_1, \ldots, \pi_n)$. We then denote by $\pi^{\mathcal{I}}$ the binary relation given by $\bigcap_{i=1}^{n} (B? \cdot \pi_i)^{\mathcal{I}}$. A pid is called *fully local* if every path π_i contains at most one role.

A *TBox* $\mathcal{T}$ is a finite set of inclusions and pids. We write $\mathcal{I} \models \mathcal{T}$ if $\mathcal{I} \models \alpha$ for all $\alpha \in \mathcal{T}$, and $id(a, b) \in \mathcal{I}$ implies $a = b$. Let $Mod(\mathcal{T})$ be the set of all RGs $\mathcal{J}$ s.t. $\mathcal{J} \models \mathcal{T}$.

Example 1. Consider an RG $\mathcal{I}$ that stores buildings, their rooms and addresses, as depicted in Figure 1. All rooms in the same building must have the same address, which is expressed using the pid $\sigma = address?(located\_at^- \cdot is\_in)$. The pid σ is not satisfied in $\mathcal{I}$, since $(address? \cdot located\_at^- \cdot is\_in)^{\mathcal{I}} = \{(\texttt{1500\_PA\_Av}, \texttt{1600\_PA\_Av})\}$. In order to satisfy σ we can merge the nodes with values "1500_PA_Av" and "1600_PA_Av", which is illustrated in Figure 3.

Direct Mapping of Relational Data to RDFS Graph Data

Here we propose a mapping from DBs to RGs. Note that, our mapping is similiar to the so-called direct mapping proposed by the W3C (Arenas et al. 2012a) and Sequeda, Arenas, and Miranker (2012). At the end of this section we will discuss the differences between the direct mapping introduced here and by Sequeda, Arenas, and Miranker (2012).

We translate relational symbols and DBs to RGs conforming to a TBox. Since our focus is on fds, we do not discuss primary and foreign keys. We show that our mapping is semantics preserving with respect to fds. First, we will define a mapping from relation symbols without fds to a TBox, s.t. all databases conforming to such a TBox are those that can be mapped to databases of the relation symbols. Then we extend this definition with a mapping from fds to dependencies over RGs. We will call the combination of both mappings *Relational to RDF graph direct mapping* (R2RG).

For every $R \in \Delta_r$ we use the following: The concept T_R represents tuples of R. Let $i \in \{1, \ldots, ar(R)\}$. The role $R\#i$ associates a tuple with its value in column i, and the concept R_i represents the values in column i. We assume the domains of the columns to be disjoint.

Definition 1 (Schema mapping sm). *The function $sm(\Delta_r)$ outputs a TBox $\mathcal{T}$ as follows. For each $R \in \Delta_r$ we introduce the following set of assertions:*

1. *All concepts are disjoint from each other, e.g., an attribute cannot also denote a tuple:*

$$T_R \sqsubseteq \neg R\_i, \quad R\_i \sqsubseteq \neg R\_j, \quad \text{for } 1 \leq i < j \leq ar(R)$$

2. *Every role has an associated domain and range:*

$$\exists R\#i \sqsubseteq T_R, \quad \exists R\#i^- \sqsubseteq R\_i, \quad \text{for } 1 \leq i \leq ar(R)$$

3. *Functionality assertions express that each tuple can only have one of each attribute:*

$$(\mathsf{funct}\ R\#i), \quad \text{for } 1 \leq i \leq ar(R)$$

4. *Identification constraints ensure that no tuple can occur twice (the relational model assumes set semantics):*

$$T_R?(R\#1, \ldots, R\#ar(R)).$$

The function sm maps relation symbols to TBoxes. Given this mapping we can translate RGs conforming to $\mathcal{T}$ into DBs and vice-versa. We now define such translations. The function $d2r(I)$ translates a DB I into an RG satisfying the TBox created by the function sm.

Definition 2 (DB to RG mapping). *Let I be a finite set of atoms. Let $\mathcal{T} = sm(\Delta_r)$. The function $d2r(I)$ returns the RG $\mathcal{J}$, defined as follows. For each atom $R(c_1, \ldots, c_{ar(R)})$ we add to $\mathcal{J}$: $T_R(t_{c_1,\ldots,c_{ar(R)}})$, where $t_{c_1,\ldots,c_{ar(R)}}$ is a new value not in $dom(I)$ and called* tuple identifier*; $R\_i(c_i)$ and $R\#i(t_{c_1,\ldots,c_{ar(R)}}, c_i)$, where $1 \leq i \leq ar(R)$.*

We now show that the RG $\mathcal{J}$ returned by $d2r(I)$ satisfies all assertions in $\mathcal{T}$.

Proposition 1. *Let $\mathcal{T} = sm(\Delta_r)$, let I be a database and $\mathcal{J}$ the RG returned by $d2r(I)$. Then $\mathcal{J} \models \mathcal{T}$.*

Proof. In order to show that $\mathcal{J} \models \mathcal{T}$ we need to show that $\mathcal{J}$ satisfies all assertions in $\mathcal{T}$. That is, the following holds for each relation $R \in \Delta_r$.

- $T_R \sqsubseteq \neg R\_i$: Every T_R atom is built using new domain elements not yet in I.
- $R\_i \sqsubseteq \neg R\_j$: All columns domains are disjoint.
- $\exists R\#i \sqsubseteq T_R$: All tuple identifiers are in both relations.
- $\exists R\#i^- \sqsubseteq R\_i$: All values are in both relations.
- $(\mathsf{funct}\ R\#i)$: For each atom we add one value to $R\#i$.
- $T_R?(R\#1, \ldots, R\#n)$: Since each tuple is unique, also the identification assertion is satisfied by $\mathcal{J}$. □

The function $r2d(\mathcal{I})$ translates an RG $\mathcal{I}$ satisfying a TBox created by $sm(\Delta_r)$ into a corresponding DB.

Definition 3 (RG to DB mapping). *Let $\mathcal{I}$ be an RG, s.t. $\mathcal{I} \models \mathcal{T}$, where $\mathcal{T} = sm(\Delta_r)$. Let $\oplus$ denote exclusive or. The function $r2d(\mathcal{I})$ returns $J = \{R(c_1, \ldots, c_{ar(R)}) \mid \exists T_R(t) \in \mathcal{I}$ s.t. $\bigwedge_{i=1}^{ar(R)} ((R\#i(t, c_i) \in \mathcal{I}) \oplus c_i = NULL)\}$.*

Given the construction of Definition 3, it is easy to see that the DB $r2d(\mathcal{I})$ has only atoms of the relation symbols in Δ_r. That is, the following holds.

Proposition 2. *Let Δ_r be a set of relation symbols, $\mathcal{T} = sm(\Delta_r)$, and let $\mathcal{I}$ be an RG, s.t. $\mathcal{I} \models \mathcal{T}$. Let $J = r2d(\mathcal{I})$. Then J has only atoms of the relation symbols in Δ_r.*

We have established a direct mapping of relational DBs to RDF graphs. We are now ready to prove a one-to-one correspondence between such DBs and RGs.

Theorem 1. *Let Δ_r be a set of relation symbols and let $\mathcal{T} = sm(\Delta_r)$. Then, we have*

$$(a) \quad I = d2r(r2d(I)), \quad \text{and}$$
$$(b) \quad \mathcal{I} \text{ is isomorphic to } r2d(d2r(\mathcal{I})).$$

Proof sketch. (a): Let I be a DB. We translate I into an RG $\mathcal{J} = d2r(I)$ and back into a DB $I' = r2d(\mathcal{J})$. Then, utilizing Propositions 1 and 2, we have that $I' = I$.

(b) Similarly, let $\mathcal{I}$ be an RG, s.t. $\mathcal{T} \models \mathcal{I}$. We translate $\mathcal{I}$ into a DB and back, i.e., $\mathcal{I}' = d2r(r2d(\mathcal{I}))$. Again using Propositions 1 and 2, we get that $\mathcal{I}'$ and $\mathcal{I}$ are isomorphic. □

As a corollary of Theorem 1 we get that $Inst(\Delta_r) = \{r2d(\mathcal{I}) \mid \mathcal{I} \models \mathcal{T}\}$ and $Mod(\mathcal{T})$ is isomorphic to $\{d2r(I) \mid I \in Inst(\Delta_r)\}$. Thus, the set $Inst(\Delta_r)$ is equal to the translated RGs satisfying the TBox $\mathcal{T} = sm(\Delta_r)$ and vice versa.

The direct mapping $\mathcal{DM}$ introduced by Sequeda, Arenas, and Miranker (2012) translates a relational schema into OWL (similar to sm) and a database instance into RDF (similar to $d2r$). Note that our direct mapping $d2r$ is a notational variant of the translation from DBs to RGs established by the direct mapping $\mathcal{DM}$. Thus, all good properties, e.g. information and query preservation, of the direct mapping $\mathcal{DM}$ that rely solely on the direct mapping of DBs to RGs are also kept by the direct mapping $d2r$. Therefore, in case of information preservation the proof of part (a) in Theorem 1 is similiar to the proof of Theorem 1 by Sequeda, Arenas, and Miranker (2012). In contrast, the proof of part (b) in Theorem 1 does not hold for $\mathcal{DM}$. The direct mapping $\mathcal{DM}$ only generates a set of domain and range assertions of object- and datatype properties from the relational schema, whereas our direct mapping sm does the same (see 2. of Definition 1) and adds disjointness of concepts, functionality and identification assertions. These additional assertions allow us to introduce a direct mapping $r2d$ of RGs to DBs and establish information preservation also for this direct mapping (part (b) of Theorem 1).

Capturing Functional Dependencies

So far we have shown that we can map DBs to RDF graphs and vice-versa. Now, we extend this mapping with functional dependencies. Path-based identification constraints (Calvanese et al. 2008) are an ideal candidate to express fds over RDF graphs. Here, we investigate the expressivity of pids and show that, surprisingly, they are not sufficient to capture fds under the direct mapping. The following example gives the idea why this is not the case.

Example 2. Consider the fd $\sigma := \{1,2\} \to 3$. The naive translation following the spirit of the direct mapping into a pid is $\delta := R\_3?(R\#3^- \cdot R\#1, R\#3^- \cdot R\#2)$. We will now show that σ and δ distinguish different DBs and RGs. In Figure 2 we give two DBs with relation symbol R and arity 4. The DB in Figure 2a does not satisfy σ, i.e. $c_1 = c_2$, whereas the DB in Figure 2b does satisfy σ. If we translate σ into the pid δ and also use $d2r$ to map the DBs in Figures 2a and 2b to RGs, depicted in Figures 2d and 2e, respectively, we have two RGs that both violate the pid δ, i.e. $c_1 = c_2$ and $c_1' = c_2'$.

The above example shows that there exist fds for which there is no intuitive translation into pids, such that the DBs that (do not) satisfy these fds translated into RGs also correspondingly (do not) satisfy the translated pids. Actually, we will show in Theorem 2 that there are fds, which cannot be translated to any set of pids, such that the previous property holds. But first, we define a special *simulation* relation to analyze the existence of paths in pairs of RGs. Recall that B denotes a basic concept and P a basic role.

Definition 4. *Let $\mathcal{I}_1$, $\mathcal{I}_2$ be a pair of RGs and let c_1 in $\mathcal{I}_1$ and c_2 in $\mathcal{I}_2$. A* (c_1, c_2)-simulation of $\mathcal{I}_1$ by $\mathcal{I}_2$ *is a relation $\sim \subseteq \Delta_v \times \Delta_v$ such that:*

- $c_1 \sim c_2$*;*
- *for every o_1 in $\mathcal{I}_1$, there is o_2 in $\mathcal{I}_2$ with $o_1 \sim o_2$;*
- *if $o_1 \sim o_2$ and $B(o_1) \in \mathcal{I}_1$, then $B(o_2) \in \mathcal{I}_2$;*
- *if $o_1 \sim o_2$ and $P(o_1, o_1') \in \mathcal{I}_1$, then there exists o_2' such that $P(o_2, o_2') \in \mathcal{I}_2$ and $o_1' \sim o_2'$;*
- *if $o_1 \sim o_2$ and $P(o_1, c_1) \in \mathcal{I}_1$, then $P(o_2, c_2) \in \mathcal{I}_2$.*

By using a (c_1, c_2)-simulation of $\mathcal{I}_1$ by $\mathcal{I}_2$ we show that once a pid is violated in $\mathcal{I}_1$, it is also violated in $\mathcal{I}_2$.

Proposition 3. *Let $\sim$ be a (c_1, c_2)-simulation of $\mathcal{I}_1$ by $\mathcal{I}_2$, and assume that $b_1 \sim b_2$. Let π be a pid. Then $(b_1, c_1) \in \pi^{\mathcal{I}_1}$ implies $(b_2, c_2) \in \pi^{\mathcal{I}_2}$.*

Proof sketch. By induction on the length, we show that every path from b_1 to c_1 in $\mathcal{I}_1$ also exists from b_2 to c_2 in $\mathcal{I}_2$. □

Utilizing the direct mapping, we now show that there is an fd that is not expressible by any set of pids. That is, the fd is able to distinguish two different DBs, whereas there is no set of pids that distinguishes the translated DBs.

Theorem 2. *There exists a set Σ of fds such that there* does not exist *a set Φ of pids such that: $I \models \Sigma$ iff $d2r(I) \models \Phi$ for all databases I.*

Proof. Let R be a relation symbol with arity 4 and let $\Sigma = \{\{1,2\} \to_R 3\}$. Let Φ be an arbitrary set of pids. To prove the claim it suffices to define DBs I and I' such that $I \not\models \Sigma$ and $I' \models \Sigma$ but $d2r(I) \not\models \Phi$ and $d2r(I') \not\models \Phi$.

Let $I = \{R(a_1, b_1, c_1, d_1), R(a_1, b_1, c_2, d_2)\}$ (Figure 2a). Clearly, $I \not\models \Sigma$. Suppose $d2r(I) \not\models \Phi$, i.e., there exists $\pi \in \Phi$ and a pair $(d, e) \in \pi^{d2r(I)}$ with $d \neq e$. Note that for $(d, e) \in \pi^{d2r(I)}$ to be true, there must exist d', e' and a role P such that $P(d, d') \in d2r(I)$ and $P(e, e') \in d2r(I)$. Due to the structure of $d2r(I)$, the latter is true only (c_1, c_2), (d_1, d_2) and (t_1, t_2) (cf. Figure 2d). I.e. $(d, e) \in \{(c_1, c_2), (d_1, d_2), (t_1, t_2)\}$. Therefore, we consider those three cases:

1. Assume $(d, e) = (c_1, c_2)$. Let $I' = \{R(a_1', b_2', c_1', d_1'), R(a_2', b_1', c_1', d_1'), \quad R(a_3', b_1', c_2', d_2'), \quad R(a_1', b_3', c_2', d_2')\}$ (Figure 2b). Clearly, $I' \models \Sigma$. Next, we need to show that there is a (c_2, c_2')-simulation $\sim$ of $d2r(I)$ by $d2r(I')$. We do so, by exhbiting such a simulation in the following table (every pair of objects connected by $\sim$ below is in the (c_2, c_2')-simulation $\sim$):

$\mathcal{M}$ / $\mathcal{M}'$	t_1, t_2	a_1	b_1	c_1, c_2	d_1, d_2
t_1', t_2', t_3', t_4'	$\sim$				
a_1', a_2', a_3'		$\sim$			
b_1', b_2', b_3'			$\sim$		
c_1', c_2'				$\sim$	
d_1', d_2'					$\sim$

	1	2	3	4
t_1	a_1	b_1	c_1	d_1
t_2	a_1	b_1	c_2	d_2

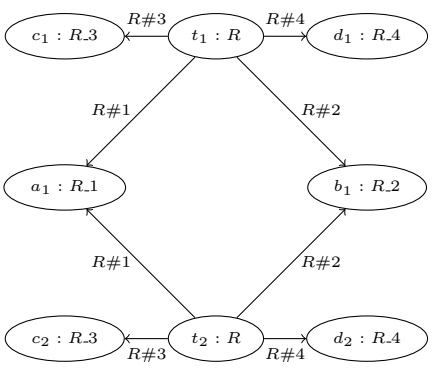
(a) DB I.

	1	2	3	4
t'_1	a'_1	b'_2	c'_1	d'_1
t'_2	a'_2	b'_1	c'_1	d'_1
t'_3	a'_3	b'_1	c'_2	d'_2
t'_4	a'_1	b'_3	c'_2	d'_2

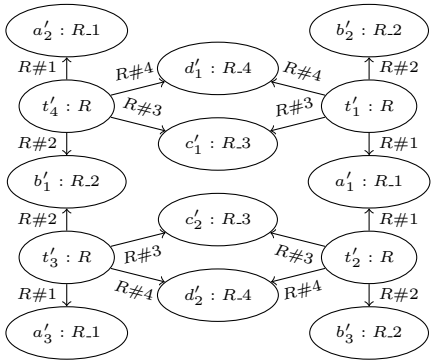
(b) DB I' used in Example 2, Case 1 and Case 2 of Theorem 2.

	A	B	C	D
t'_1	a'_1	b'_2	c'_1	d'_1
t'_2	a'_1	b'_1	c'_1	d'_2

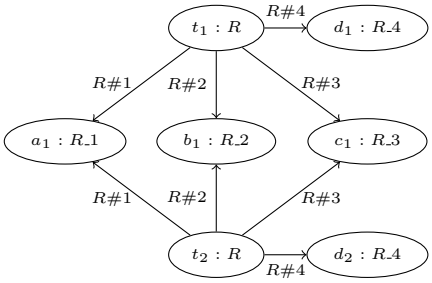
(c) DB I' used in Case 3 of Theorem 2.



(d) RG $\mathcal{M} = i2m_{R[U]}(I)$.



(e) RG $\mathcal{M}' = i2m_{R[U]}(I')$ used in Example 2, Case 1 and Case 2 of Theorem 2.



(f) RG $\mathcal{M}' = i2m_{R[U]}(I')$ used in Case 3 of Theorem 2.

Figure 2: DBs and their corresponding RGs used in the proof of Theorem 2.

Therefore, $c_1 \sim c'_1$. By Proposition 3, $(c'_1, c'_2) \in \pi^{d2r(I')}$. Since $c'_1 \neq c'_2$, we get $d2r(I') \not\models \pi$ and thus $d2r(I') \not\models \Phi$.

2. Assume $(d, e) = (d_1, d_2)$. The argument is identical to the above case. Simply replace c_1 by d_1 and c_2 by d_2.
3. Assume $(d, e) = (t_1, t_2)$. Let $I' = \{R(a'_1, b'_1, c'_1, d'_1), R(a'_1, b'_1, c'_1, d'_2)\}$ (Figure 2c). Clearly, $I' \models \Sigma$. Next, we need to show that there is a (t_2, t'_2)-simulation $\sim$ of $d2r(I)$ by $d2r(I')$. Such a simulation is given in the following table (every pair of objects connected by $\sim$ below is in the (t_2, t'_2)-simulation $\sim$):

$\mathcal{M}'$ \ $\mathcal{M}$	t_1, t_2	a_1	b_1	c_1, c_2	d_1, d_2
t'_1, t'_2	$\sim$				
a'_1		$\sim$			
b'_1			$\sim$		
c'_1				$\sim$	
d'_1, d'_2					$\sim$

Therefore, $t_1 \sim t'_1$. By Proposition 3, $(t'_1, t'_2) \in \pi^{d2r(I')}$. Since $t'_1 \neq t'_2$, we get $d2r(I') \not\models \pi$ and thus $d2r(I') \not\models \Phi$. □

As we have seen, pids do not properly capture fds in RGs. We now extend path-based to tree-based identification constraints. They allow us to overcome the limitations of pids.

Definition 5. *A* tree-based identification constraint (tid) *is an expression constructed using the following grammar:*

$$\tau ::= \varepsilon \mid \sigma \cdot \tau \mid (\tau, \tau)$$

where σ is an ordinary role or a test role of the form $C?$, and C is a concept. We usually write $C?(\tau_1, \tau_2 \ldots, \tau_n)$ instead of $(C? \cdot \tau_1, (\tau_2, (\ldots (\tau_{n-1}, \tau_n) \ldots))$.

The function $\pi^{\mathcal{I}}$ is easily extended from pids to tids.

Definition 6. *Let $\mathcal{I}$ be an RG. We define the function $\tau^{\mathcal{I}}$ that assigns to each tid τ a binary relation based on $\mathcal{I}$ as follows:*

- $(\varepsilon)^{\mathcal{I}} = \{(e, e) \mid e \in \Delta_v\}$,
- $(\sigma \cdot \tau)^{\mathcal{I}} = \{(e_1, e_2) \mid \exists (e'_1, e'_2) \in \tau^{\mathcal{I}} \quad s.t. \quad \sigma(e_1, e'_1) \in \mathcal{I} \wedge \sigma(e_2, e'_2) \in \mathcal{I}\}$,
- $(\tau_1, \tau_2)^{\mathcal{I}} = \tau_1^{\mathcal{I}} \cap \tau_2^{\mathcal{I}}$.

We say that $\mathcal{I} \models \tau$, if $(e_1, e_2) \in \tau^{\mathcal{I}}$ implies $e_1 = e_2$.

Notice that each pid is also a tid, but obviously the converse is not the case. From now on, we assume that a TBox $\mathcal{T}$ also allows for tids, which is called a *DL-Lite*$_{RDFS,tid}$ TBox.

Example 3. Following Example 2 we can express the fd $\sigma := \{1, 2\} \rightarrow 3$ using the tid $\tau := R\_3?(R\#3^- \cdot (R\#1, R\#2))$. Clearly, considering the RG $\mathcal{M}$ depicted in Figure 2d, we have $(c_1, c_2) \in \tau^{\mathcal{M}}$. Since $c_1 \neq c_2$, τ is violated in $\mathcal{M}$. The RG $\mathcal{M}'$ depicted in Figure 2e does not violate τ. Thus, τ captures the fd σ over $\mathcal{M}$ and $\mathcal{M}'$.

We can generalize the translation used in Example 3 to arbitrary fds.

Definition 7 (Fds to tids direct mapping). *Given a relational symbol R and a set Σ of fds, the function $dm(\Sigma)$ outputs a set of tids Φ as follows: For each fd in Σ of the form*

$$\{i_1, \ldots i_k\} \rightarrow i,$$

we add the following tid to Φ:

$$R\_i?(R\#i^- \cdot (R\#i_1, \ldots, R\#i_k))$$

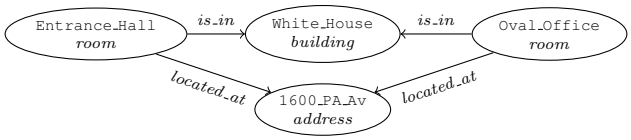


Figure 3: An RG $\mathcal{J}$ with data about buildings.

We now combine the fds to tids direct mapping (Definition 7) with the relation symbols to TBox direct mapping (Definition 1) to obtain the following:

Definition 8 (Relational to RDF graph direct mapping (R2RG)). *Given a set Δ_r of relation symbols and a set Σ of FDs over the relation symbols in Δ_r, the function $rdm(\Delta_r, \Sigma)$ outputs a TBox $\mathcal{T} = sm(\Delta_v) \cup dm(\Sigma)$.*

We now show that the tids generated by this direct mapping indeed captures functional dependencies in RGs.

Theorem 3. *Let I be a database with relation symbols Δ_r, let $\mathcal{T} = sm(\Delta_v)$, and let Σ be a set of fds over the relation symbols in Δ_r. Then,*

$$I \models \Sigma \qquad \textit{iff} \qquad d2r(I) \models dm(\Sigma)$$

Proof sketch. Both directions are proved by contraposition.

($\Rightarrow$) We assume that there is a tid $\tau \in dm(\Sigma)$ that is violated by $d2r(I)$. Hence, there is a witness, i.e. two objects, for the violation of τ in $d2r(I)$. We then show, that the corresponding two values in $I = r2d(d2r(I))$ are a witness for the violation of the translated fd.

($\Leftarrow$) We assume that there is an fd $\sigma \in \Sigma$ that is violated by I. Hence, there is a witness, i.e. two values, for the violation of σ in I. We then show, that the corresponding two objects in $d2r(I)$ are a witness for the violation of the translated tid. □

Normal Form

In relational databases, normal forms are used to avoid update anomalies originating from redundancies induced by functional dependencies. Such redundancies are also possible in RDF graphs as we will see in the following example.

Example 4. Consider the RG from Example 1. Remember that the address is the same for all rooms in one building, which is expressed using the tid $address?\left(located\_at^- \cdot is\_in\right)$. Now consider the DB illustrated in Figure 3. Clearly, the information that the "White_House" is located in "16000_PA_Av" is stored redundantly. If we change the address of one room, we violate the above tid (see Figure 1). This is what is called an update anomaly. We can avoid such a redundancy by storing the address directly connected to the building concept.

In order to detect a redundancy as illustrated in Example 4, we need to ensure that the properties specified by each tid are stored as local as possible (e.g., directly connected as in our example). This is captured as follows:

Definition 9 (RDF Normal Form (RNF)). *Let $\mathcal{T}$ be a TBox with tids, and let Φ denote the tids in $\mathcal{T}$. Then $\mathcal{T}$ is in* RDF Normal Form *(RNF) if and only if there is a set Φ' of fully local tids, s.t. $Mod(\mathcal{T}) = Mod((\mathcal{T} \setminus \Phi) \cup \Phi')$.*

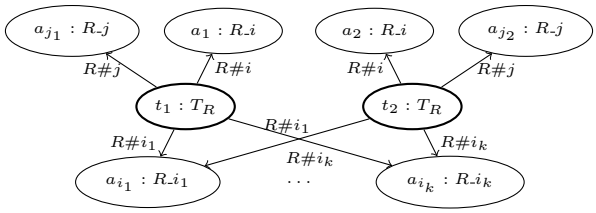


Figure 4: The RG $\mathcal{A}$.

We now show that RNF captures BCNF in RGs using the relational to RDF graph direct mapping.

Theorem 4. *Let R be a relation symbol and Σ a set of fds over R. Then (R, Σ) is in BCNF iff $rdm(R, \Sigma)$ is in RNF.*

Proof. Let $\mathcal{T} = rdm(R, \Sigma)$ and let Φ be the set of tids in $\mathcal{T}$.

($\Rightarrow$) Suppose (R, Σ) is in BCNF. We need to show that there exists a set Φ', s.t. each $\sigma \in \Phi'$ is fully local and $Mod(\mathcal{T}) = Mod((\mathcal{T} \setminus \Phi) \cup \Phi')$. We show this by substituting each non-local $\sigma \in \Phi$ by an equivalent fully local tid σ' as follows: The tid σ is of the form $R\_i?(R\#i^- \cdot (R\#i_1, \ldots, R\#i_k))$ and corresponds to the fd $\{i_1, \ldots, i_k\} \rightarrow_R i$. Since (R, Σ) is in BCNF, the attributes $i_1, \ldots, i_k$ must be a super-key, i.e., $\Sigma \models \{i_1, \ldots, i_k\} \rightarrow_R \{1, \ldots, ar(R)\}$. Therefore, we add to Φ' the tid $\sigma' = T_R?(R\#i_1, \ldots, R\#i_k)$. It can be verified that $\{\sigma', (\text{funct } R\#i)\} \models \sigma$.

($\Leftarrow$) We proof the contrapositive. Suppose (R, Σ) is not in BCNF. We need to show that there does not exist a set Φ'_Σ, s.t. each $\sigma \in \Phi'_\Sigma$ is fully local. Since (R, Σ) is not in BCNF, there exists an fd $\sigma = \{i_1 \ldots i_k\} \rightarrow_R i$ in Σ, s.t. $\{i_1 \ldots i_k\}$ is not a super-key. Therefore, there exists an attribute j, s.t. $\{i_1 \ldots i_k\} \not\rightarrow_R j$. The fd σ is translated by the R2RG direct mapping into a tid $\varphi = R\_i?(R\#i^- \cdot (R\#i_1, \ldots, R\#i_k))$. Now consider the witness RG $\mathcal{A}$ for the violation of φ depicted in Figure 4, i.e., $(a_1, a_2) \in \varphi^{\mathcal{A}}$. Note that since $i_1, \ldots, i_k$ is not a super-key, the tuple identifiers t_1 and t_2 are distinct. Suppose there is a set of fully local tids Θ, s.t. $(a_1, a_2) \in \Theta^{\mathcal{A}}$. Since a_1 and a_2 are only reachable by $R\#i$, also $(t_1, t_2) \in \Theta^{\mathcal{A}}$. This contradicts that t_1 and t_2 are distinct. Hence, there is no such set of fully local tids Θ. □

Note that the proof of Theorem 4 relies only on the tids and functionality assertions in the TBox $\mathcal{T}$. Hence, to establish that RNF captures BCNF, a direct mapping of fds together with functionality assertions suffices. Clearly, for such a direct mapping Property (b) of Theorem 1 is no longer true.

CHECKRNF Next, we provide a polynomial time algorithm to check whether a given TBox $\mathcal{T}$ is in RNF. To this end, we adapt the well-known database *chase* technique (Maier, Mendelzon, and Sagiv 1979; Beeri and Vardi 1984).

Definition 10. *Let $\mathcal{I}$ be an RG and $\mathcal{T}$ a TBox. An RG $\mathcal{I}'$ is called a* chase *of $\mathcal{I}$ w.r.t. $\mathcal{T}$ if $\mathcal{I}'$ can be obtained from $\mathcal{I}$ by exhaustively applying the following rules:*

- *If $id(a, b) \in \mathcal{I}$ or $\mathcal{T}$ has τ such that $(a, b) \in \tau^{\mathcal{I}}$, and $a \neq b$, then replace every occurrence of a in $\mathcal{I}$ by b.*

- *If $B \sqsubseteq A \in \mathcal{T}$ and $B(c) \in \mathcal{I}$, then add $A(c)$ to $\mathcal{I}$.*
- *If $P \sqsubseteq R \in \mathcal{T}$ and $P(c,d) \in \mathcal{I}$, then add $R(c,d)$ to $\mathcal{I}$.*
- *If $P \sqsubseteq R^- \in \mathcal{T}$ and $P(c,d) \in \mathcal{I}$, then add $R(d,c)$ to $\mathcal{I}$.*

Let $chase(\mathcal{I}, \mathcal{T})$ denote a chase of $\mathcal{I}$ w.r.t. $\mathcal{T}$. Note that $chase(\mathcal{I}, \mathcal{T})$ is unique up to renaming of values.

Our next step is to show how the structure of a given tid τ can be represented as an RG.

Definition 11 (Characteristic RG). *Let τ be a tid, and a, b a pair of values. Then $rg(\tau, a, b)$ is an RG defined inductively on the structure of τ as follows:*

- *If $\tau = id$, then $rg(\tau, a, b) = \{id(a, b)\}$.*
- *If $\tau = A? \cdot \tau'$, then $rg(\tau, a, b) = rg(\tau', a, b) \cup \{A(a), A(b)\}$.*
- *If $\tau = (\exists R)? \cdot \tau'$, then $rg(\tau, a, b) = rg(\tau', a, b) \cup \{R(a, a'),\ R(b, b')\}$, where a', b' are fresh values, i.e., $a', b' \notin dom(rg(\tau', a, b))$*
- *If $\tau = (\exists R^-)? \cdot \tau'$, then $rg(\tau, a, b) = rg(\tau', a, b) \cup \{R(a', a), R(b', b)\}$, where a', b' are fresh values.*
- *If $\tau = R \cdot \tau'$, then $rg(\tau, a, b) = \{R(a, a'), R(b, b')\} \cup \mathcal{I}$, where $\mathcal{I}$ is obtained from $rg(\tau', a', b')$ by replacing every occurrence of a, b with fresh values.*
- *If $\tau = (\tau_1, \tau_2)$, then $rg(\tau, a, b) = rg(\tau_1, a, b) \cup \mathcal{I}$, where $\mathcal{I}$ is obtained from $rg(\tau_2, a, b)$ by renaming every value c, $c \notin \{a, b\}$, with a fresh value.*

Note that for every $(a', b') \in \tau^{\mathcal{I}}$, there is a homomorphism h from $rg(\tau, a, b)$ to $\mathcal{I}$ with $h(a) = a'$ and $h(b) = b'$. The following shows how $\mathcal{T} \models \tau$ can be decided by employing the chase of a characteristic RG.

Proposition 4. *Let τ be a tid and $\mathcal{T}$ a TBox. Let $\mathcal{I} = rg(\tau, a, b)$ and $\mathcal{J} = chase(\mathcal{I}, \mathcal{T})$. Then $\mathcal{T} \models \tau$ iff (a) there is a negative inclusion $\alpha \in \mathcal{T}$ s.t. $\mathcal{J} \not\models \alpha$, or (b) $\mathcal{J} \models \tau$.*

Proof sketch. ($\Rightarrow$) Observe that $\mathcal{J} \models \alpha$ for every tid and positive inclusion α in $\mathcal{T}$. If in addition $\mathcal{J} \models \alpha$ for all negative inclusions α of $\mathcal{T}$, i.e., $\mathcal{J} \models \mathcal{T}$, then by assumption $\mathcal{T} \models \tau$ we must have $\mathcal{J} \models \tau$.

($\Leftarrow$) Assume towards a contradiction that $\mathcal{T} \not\models \tau$, i.e., there is an RG $\mathcal{M}$ s.t. $\mathcal{M} \not\models \tau$. That is, there exist $a' \neq b'$ with $(a', b') \in \tau^{\mathcal{M}}$. Then there is homomorphism h from $\mathcal{I}$ to $\mathcal{M}$ with $h(a) = a'$ and $h(b) = b'$. In other words, $\mathcal{I}$ is a counter-example for τ. It is easy to check that the counter-example is preserved after applying any chase rule. That is, we get $chase(\mathcal{I}, \mathcal{T}) \models \mathcal{T}$ but still $chase(\mathcal{I}, \mathcal{T}) \not\models \tau$. □

It is not difficult to see that the method of Proposition 4 allows to check $\mathcal{T} \models \tau$ in polynomial time in the size of the input. Given that there are only finitely many, yet double exponentially, different sets of fully local tids over a given set of basic concepts and roles, the decidability of RNF can already be inferred. However, to obtain a polynomial time algorithm we need to reduce the search space significantly.

Let $\mathcal{I}$ be an RG and a, b a pair of values. We next show how to construct the "maximal" tid that may be used to collapse a and b in $\mathcal{I}$ using the *types* (Tp) of values in an RG. In the following, given a set $\Gamma = \{B_1, \ldots, B_n\}$ of basic concepts, we write $\Gamma?$ instead of $B_1? \cdots B_n?$.

Definition 12. *Let $\mathcal{I}$ be an RG and a, b a pair of values. Let $Tp(c) = \{A \mid A(c) \in \mathcal{I}\} \cup \{\exists R \mid R(c, c') \in \mathcal{I}\} \cup \{\exists R^- \mid R(c', c) \in \mathcal{I}\}$. We let $Tp(a, b) = Tp(a) \cap Tp(b)$. Then $\tau(\mathcal{I}, a, b) = Tp(a, b)?(\tau_1, \ldots, \tau_n)$, where $\{\tau_1, \ldots, \tau_n\} = \{R \cdot (Tp(c))? \mid \{R(a, c), R(b, c)\} \subseteq \mathcal{I}\}$.*

In Algorithm 1 we present our procedure to check whether a given TBox $\mathcal{T}$ is in RNF. If the answer is "yes", the procedure returns the desired normalized TBox. Intuitively, it checks whether each tid $\tau \in \mathcal{T}$ can be simulated by a set of fully local tids. To this end, the algorithm computes a characteristic RG of τ and then chases it using the inclusions of $\mathcal{T}$ and tids τ' extracted from the current RG according to the method in Definition 12. To ensure soundness, the algorithm checks that each applied τ' is in fact entailed from $\mathcal{T}$.

Theorem 5. *The algorithm* CHECKRNF *in Algorithm 1 is a decision procedure for RNF.*

Proof. Let $\mathcal{T}$ be a TBox and let Φ the set of tids in $\mathcal{T}$. Let Φ^* be a set of all fully local tids τ such that $\mathcal{T} \models \tau$. Observe that $\mathcal{T}$ is in RNF iff $Mod(\mathcal{T}) = Mod((\mathcal{T} \setminus \Phi) \cup \Phi^*)$. Indeed, assume $\mathcal{T}$ is in RNF, i.e., there is a set of fully local tids Φ' with $Mod(\mathcal{T}) = Mod((\mathcal{T} \setminus \Phi) \cup \Phi')$. Hence $\Phi' \subseteq \Phi^*$ must be true, and thus $Mod((\mathcal{T} \setminus \Phi) \cup \Phi^*) \subseteq Mod((\mathcal{T} \setminus \Phi) \cup \Phi') = Mod(\mathcal{T})$. Moreover, $Mod(\mathcal{T}) \subseteq Mod((\mathcal{T} \setminus \Phi) \cup \Phi^*)$ because $\mathcal{T} \models \tau$ for each $\tau \in \Phi^*$. This shows that deciding if $\mathcal{T}$ is in RNF can be reformulated to deciding $Mod(\mathcal{T}) = Mod((\mathcal{T} \setminus \Phi) \cup \Phi^*)$, which is equivalently formulated as:

1. $(\mathcal{T} \setminus \Phi) \cup \Phi^* \models \tau$, for all $\tau \in \Phi$, and
2. $\mathcal{T} \models \tau$, for all $\tau \in \Phi^*$.

Clearly, (2) holds by the definition of Φ^*. For (1), let $\tau \in \Phi$ and $\mathcal{I} = rg(\tau, a, b)$. By employing Proposition 4, it suffices to check that either there is a negative inclusion $\alpha \in \mathcal{T}$ with $chase(\mathcal{I}, (\mathcal{T} \setminus \Phi) \cup \Phi^*) \not\models \alpha$ or $chase(\mathcal{I}, (\mathcal{T} \setminus \Phi) \cup \Phi^*) \models \tau$, for all $\tau \in \Phi$. Algorithm 1 implements this by computing $chase(\mathcal{I}, (\mathcal{T} \setminus \Phi) \cup \Phi^*))$. To this end, it traverses pairs $a \neq b$ of $\mathcal{I}$ and checks whether Φ^* has a tid τ' that would collapse a and b in the current RG. Clearly, such a τ' exists iff the tid $\tau(\mathcal{I}, a, b)$ collapses a and b in the current RG. If the previous succeeds for all $\tau \in \Phi$, the algorithm has the set $\hat{\Phi} \subseteq \Phi^*$ of all tids that were used to compute each $chase(\mathcal{I}, (\mathcal{T} \setminus \Phi) \cup \Phi^*)$. Thus $(\mathcal{T} \setminus \Phi) \cup \hat{\Phi})$ is the desired normalized TBox. □

Theorem 6. *Deciding whether $\mathcal{T}$ is in RNF is feasible in polynomial time.*

Proof sketch. To prove the claim we analyze Algorithm 1. The loop in line 2 has at most $|\Phi|$ iterations, which is linear in the size of Φ. The RG $\mathcal{I}$ in line 3 can be constructed in linear time in the size of Φ. The number of iterations of the "repeat" loop is bounded by a polynomial in the size of Φ and $\mathcal{T}$. The condition in the "if" statement on line 6 can be decided in polynomial time in the size of $\mathcal{I}$ and $\mathcal{T}$. Observe that $\tau(\mathcal{I}, a, b)$ can be constructed in polynomial time in the size of $\mathcal{I}, a, b$. Moreover, using Proposition 4, $\mathcal{T} \models \tau(\mathcal{I}, a, b)$ can be decided by computing $chase(\tau(\mathcal{I}, a, b), \mathcal{T})$, which is

Algorithm 1: CHECKRNF

Input: TBox $\mathcal{T}$ with Φ the set of tids in $\mathcal{T}$

Output: If exists, a set $\hat{\Phi}$ of fully local tids s.t.
$Mod(\mathcal{T}) = Mod((\mathcal{T} \setminus \Phi) \cup \hat{\Phi})$

1 $\hat{\Phi} \leftarrow \emptyset$;
2 **foreach** $\tau \in \Phi$ **do**
3 $\quad \mathcal{I} \leftarrow rg(\tau, c, d)$ (c, d are arbitrary values) ;
4 $\quad$ **repeat**
5 $\quad\quad$ *change* $\leftarrow$ *false* ;
6 $\quad\quad$ **if** *exist* $a \neq b$ *in* $\mathcal{I}$ *s.t.* $\mathcal{T} \models \tau(\mathcal{I}, a, b)$ **then**
7 $\quad\quad\quad \hat{\Phi} \leftarrow \hat{\Phi} \cup \{\tau(\mathcal{I}, a, b)\}$;
8 $\quad\quad\quad$ Identify a and b in $\mathcal{I}$;
9 $\quad\quad\quad$ *change* $\leftarrow$ *true* ;
10 $\quad\quad$ **if** $\mathcal{I} \neq chase(\mathcal{I}, \mathcal{T} \setminus \Phi)$ **then**
11 $\quad\quad\quad \mathcal{I} \leftarrow chase(\mathcal{I}, \mathcal{T} \setminus \Phi)$;
12 $\quad\quad\quad$ *change* $\leftarrow$ *true* ;
13 $\quad$ **until** *change* $=$ *false*;
14 $\quad$ **if** $\mathcal{I} \models \mathcal{T}$ *and* $\mathcal{I} \not\models \tau$ **then**
15 $\quad\quad$ **return** "not in RNF" ;
16 **return** $(\mathcal{T} \setminus \Phi) \cup \hat{\Phi}$;

feasible in polynomial time. Finally, the condition in line 14 can be computed in polynomial time. □

We are now going to illustrate in two examples the algorithm CHECKRNF.

Example 5. Consider the tid $\tau := address?(located\_at^- \cdot is\_in)$. from Example 4. There we have argued that τ leads to a violation of RNF. We now show that CHECKRNF returns "not in RNF" on τ. First, we can use as characteristic database $\mathcal{I} = rg(\tau, \texttt{1500\_PA\_Av}, \texttt{1600\_PA\_Av})$ the RG depicted in Figure 1. We now check the condition in line 6 of CHECKRNF. For example, consider the pair `Entrance_Hall` and `Oval_Office` in $\mathcal{I}$. Then, $\tau(\mathcal{I}, \texttt{Entrance\_Hall}, \texttt{Oval\_Office}) = \exists located\_at?(is\_in)$. Clearly, $\mathcal{T} \not\models \exists located\_at?(is\_in)$. Note that there does not exist any other pair $a \neq b$ in $\mathcal{I}$ s.t. $\mathcal{T} \models \tau(\mathcal{I}, a, b)$. Since then the condition in line 10 is also false, the inner loop terminates. It is easy to see that $\mathcal{I} \models \mathcal{T}$ and $\mathcal{I} \not\models \tau$, hence the algorithm returns "not in RNF".

Example 6. Consider the following set Φ of tids:

$$R\_2?(R\#2^- \cdot T_R? \cdot \exists R\#3? \cdot R\#1), \quad (1)$$

$$R\_3?(R\#3^- \cdot T_R? \cdot (R\#1, R\#2)), \quad (2)$$

$$T_R?(R\#1, R\#2, R\#3), \quad (3)$$

$$R\_1?(R\#1^-), \quad R\_2?(R\#2^-), \quad R\_3?(R\#3^-) \quad (4)$$

Note that the tids (1) and (2) in Φ correspond to the fds $1 \rightarrow_R 2$ and $\{1, 2\} \rightarrow_R 3$. The tid (3) guarantees distinct tuples and the tids in (4) correspond to functionality assertions. If the arity of R is 3, then these fds are in BCNF. We will now show that Φ is also in RNF. The algorithm CHECKRNF computes for every tid $\tau \in \Phi$ the following (illustrated at the tid (1)):

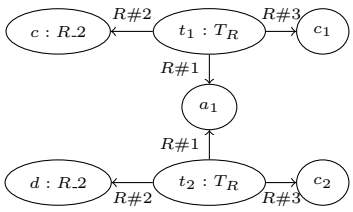


Figure 5: The characteristic RG $\mathcal{I} = rg(\tau, c, d)$.

- First, it builds the characteristic RG $\mathcal{I} = rg(\tau, c, d)$, which is depicted in Figure 5.
- We now have, that there exists $t_1 \neq t_2$, s.t. $\mathcal{T} \models \tau(\mathcal{I}, t_1, t_2)$. Note that $\tau(\mathcal{I}, t_1, t_2) = \exists R\#2^-?(T_R? \cdot \exists R\#3? \cdot R\#1)$. We add $\tau(\mathcal{I}, t_1, t_2)$ to $\hat{\Phi}$ and identify t_1 and t_2 in $\mathcal{I}$. Since the $chase(\mathcal{I}, \mathcal{T} \setminus \Phi)$ does not change $\mathcal{I}$ we continue with another loop.
- Next, we have that there exists $c \neq d$, s.t. $\mathcal{T} \models \tau(\mathcal{I}, c, d)$, where $\tau(\mathcal{I}, c, d) = R\_2?(R\#2^-)$. Hence, we add $R\_2?(R\#2^-)$ to $\hat{\Phi}$ and identify c and b in $\mathcal{I}$.

The algorithm CHECKRNF does the same for the tids (2) - (4) and then returns the following set $\hat{\Phi}$ of fully local tids:

$$\exists R\#2^-?(T_R? \cdot \exists R\#3? \cdot R\#1)$$
$$\exists R\#3^-?(T_R? \cdot (R\#1, R\#2))$$
$$T_R?(R\#1, R\#2, R\#3),$$
$$R\_1?(R\#1^-), \quad R\_2?(R\#2^-), \quad R\_3?(R\#3^-)$$

Discussion and Conclusions

In this paper we have proposed an extensions of the direct mapping of relational data to RDF to ensure a one-to-one correspondence between relational databases and RDF graphs. We started by incorporating disjointness and functionality constraints into the direct mapping. To transfer also functional dependencies, we extended previous forms of identification constraints to tree-based ids. Finally, we introduced an RDF Normal Form that turned out to capture precisely the well-known BCNF of relational schemas.

On top of our agenda for future research is the extension of our work on RNF. So far, we have concentrated on preserving BCNF of a relational schema under the direct mapping of relational data to RDF. However, normal forms for eliminating redundancies in the data would be an interesting topic for the design of TBoxes in general. We thus see two main directions to continue our work. On the one hand, we would like to extend the definition of our RNF to more expressive DLs than *DL-Lite*$_{RDFS,tid}$. Note that this raises highly non-trivial questions concerning the recognizability of the normal form, since our PTIME-membership result in Theorem 6 crucially depends on the language restrictions of *DL-Lite*$_{RDFS,tid}$. On the other hand, we also want to investigate relaxations of our definition of RNF. In our current definition, we request that a set of tids must be equivalent to a set of fully local ids. This allows us to capture BCNF in *DL-Lite*$_{RDFS,tid}$. However, for the definition of a normal form of more expressive DLs,

the equivalence of tids to a richer class of ids – such as local pids considered by Calvanese et al. (2008) may be more appropriate.

Acknowledgments

We thank the anonymous AAAI-2014 referees for their very helpful comments and suggestions.

Diego Calvanese has been partially supported by the Wolfgang Pauli Institute Vienna, and by the EU IP project Optique (*Scalable End-user Access to Big Data*), grant agreement n. FP7-318338. Wolfgang Fischl, Reinhard Pichler, Emanuel Sallinger and Mantas Šimkus have been partially supported by the Austrian Science Fund (FWF):P25207-N23, (FWF):P25518-N23 and by the Vienna Science and Technology Fund (WWTF) project ICT12-15.

References

Arenas, M.; Bertails, A.; Prud′hommeaux, E.; and Sequeda, J. 2012a. A direct mapping of relational data to RDF. W3C Recommendation, W3C. Available at http://www.w3.org/TR/rdb-direct-mapping.

Arenas, M.; Botoeva, E.; Calvanese, D.; Ryzhikov, V.; and Sherkhonov, E. 2012b. Exchanging description logic knowledge bases. In *Proc. of the 13th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR 2012)*, 308–318. AAAI Press.

Beeri, C., and Vardi, M. Y. 1984. A proof procedure for data dependencies. *J. of the ACM* 31(4):718–741.

Bizer, C.; Heath, T.; and Berners-Lee, T. 2009. Linked data - the story so far. *Int. J. on Semantic Web Information Systems* 5(3):1–22.

Brickley, D., and Guha, R. V. 2004. RDF vocabulary description language 1.0: RDF Schema. W3C Recommendation, World Wide Web Consortium. Available at http://www.w3.org/TR/rdf-schema/.

Calvanese, D.; De Giacomo, G.; Lembo, D.; Lenzerini, M.; Poggi, A.; and Rosati, R. 2006. Linking data to ontologies: The description logic DL-Lite$_A$. In *Proc. of the 2nd Int. Workshop on OWL: Experiences and Directions (OWLED 2006)*, volume 216 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Calvanese, D.; De Giacomo, G.; Lembo, D.; Lenzerini, M.; and Rosati, R. 2008. Path-based identification constraints in description logics. In *Proc. of the 11th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR 2008)*, 231–241. AAAI Press.

Calvanese, D.; De Giacomo, G.; and Lenzerini, M. 2001. Identification constraints and functional dependencies in description logics. In *Proc. of the 17th Int. Joint Conf. on Artificial Intelligence (IJCAI 2001)*, 155–160. Morgan Kaufmann.

Codd, E. F. 1971a. Further normalization of the data base relational model. IBM Research Report RJ909, IBM, San Jose, California.

Codd, E. F. 1971b. Normalized data structure: A brief tutorial. In *Proc. of the SIGFIDET Workshop*, 1–17. ACM.

He, B.; Patel, M.; Zhang, Z.; and Chang, K. C.-C. 2007. Accessing the deep web. *Communications of the ACM* 50(5):94–101.

HM Government. 2014. data.gov.uk. http://data.gov.uk.

Madhavan, J.; Afanasiev, L.; Antova, L.; and Halevy, A. Y. 2009. Harnessing the deep web: Present and future. In *Proc. of the 4th Biennial Conf. on Innovative Data Systems Research (CIDR 2009)*. www.cidrdb.org.

Maier, D.; Mendelzon, A. O.; and Sagiv, Y. 1979. Testing implications of data dependencies. *ACM Trans. on Database Systems* 4(4):455–469.

Mannila, H., and Räihä, K.-J. 1992. *The Design of Relational Databases*. Addison Wesley Publ. Co.

Sequeda, J.; Arenas, M.; and Miranker, D. P. 2012. On directly mapping relational databases to RDF and OWL. In *Proc. of the 21st Int. World Wide Web Conf. (WWW 2012)*, 649–658. ACM.

Toman, D., and Weddell, G. E. 2005. On the interaction between inverse features and path-functional dependencies in description logics. In *Proc. of the 19th Int. Joint Conf. on Artificial Intelligence (IJCAI 2005)*, 603–608.

Toman, D., and Weddell, G. E. 2008. On keys and functional dependencies as first-class citizens in description logics. *J. of Automated Reasoning* 40(2–3):117–132.

US Government. 2014. data.gov. http://www.data.gov.