Knowledge Transformation using a Hypergraph Data Model

Lama Al Khuzayem[1] and Peter McBrien[1]

1 **Imperial College London**
South Kensington Campus
London SW7 2AZ
l.al-khuzayem11@imperial.ac.uk
p.mcbrien@imperial.ac.uk

Abstract

In the Semantic Web, knowledge integration is frequently performed between heterogeneous knowledge bases. Such knowledge integration often requires the schema expressed in one knowledge modelling language be translated into an equivalent schema in another knowledge modelling language. This paper defines how schemas expressed in OWL-DL (the Web Ontology Language using Description Logic) can be translated into equivalent schemas in the Hypergraph Data Model (HDM). The HDM is used in the AutoMed data integration (DI) system. It allows constraints found in data modelling languages to be represented by a small set of primitive constraint operators. By mapping into the AutoMed HDM language, we are then able to further map the OWL-DL schemas into any of the existing modelling languages supported by AutoMed. We show how previously defined transformation rules between relational and HDM schemas, and our newly defined rules between OWL-DL and HDM schemas, can be composed to give a bidirectional mapping between OWL-DL and relational schemas through the use of the both-as-view approach in AutoMed.

1998 ACM Subject Classification H.2.5 Heterogeneous Databases

Keywords and phrases Knowledge Transformation, Hypergraph Data Model, BAV Mappings

Digital Object Identifier 10.4230/OASIcs.ICCSW.2012.1

1 Introduction

One of the crucial impediments that hinder the realisation of the Semantic Web vision is the integration of ontologies [1, 2]. Since ontologies are a form of knowledge representation, we use the terms ontology integration (OI) and knowledge integration (KI) interchangeably.

The increasing number of ontologies that were made publicly available on the Web, has evolved the Web into a global ontology [3]. The main purpose of this global ontology is to provide a unified query interface for the local ontologies. A crucial problem in this context is how to specify the mappings between the global ontology and the local ontologies [1]. The main mapping approaches cited in the literature are Global-As-View (GAV) [4], Local-As-View (LAV) [4], Global-Local-As-View (GLAV) [5], and Both-As-View (BAV) [6].

The problem of OI has been extensively investigated in the literature (e.g. [1, 2, 7, 8, 9, 10, 11]). By closely examining these OI proposals, we have identified two things. Firstly, while BAV is the most expressive mapping approach, none have used it. In contrast to GAV, LAV, and GLAV, BAV is not only capable of providing a complete mapping between schemas in both directions, but also the mappings between schemas are described as a pathway of primitive transformation steps applied in sequence in the form of add, delete, rename,

2012 Imperial College Computing Student Workshop (ICCSW'12).
Editor: Andrew V. Jones; pp. 1–7
OpenAccess Series in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

extend, and contract. Hence a further advantage of the approach, is that composition of data mappings may be performed such that mapping two schemas to one common schema will produce a bidirectional mapping between the original two data sources [12]. Secondly, current approaches integrate ontologies represented, for example, in the Resource Description Framework Schema (RDFS) [13] or the Web Ontology Language (OWL) [14] by choosing one of them as the Common Knowledge Model (CKM) and converting all the other modelling languages into that CKM. Using a high-level CKM such as RDFS or OWL greatly complicates the mapping process. This is because there is rarely a simple correspondence between their modelling constructs [15].

In this paper, we show how to integrate knowledge bases, represented in OWL-DL, using a low-level Hypergraph Data Model (HDM) as the CKM. Our approach has the advantage of clearly separating the modelling of data structure from the modelling of constraints on the data. Moreover, the HDM supports a very small set of low-level elemental modelling primitives (nodes, edges, and constraints) which makes it better suited for use as a CKM than higher-level modelling languages [15]. The HDM is the common data model of the AutoMed DI system [12]. The AutoMed system [12] is distinguished from other DI systems for handling a wide range of data modelling languages through representing their constraints as BAV transformations [16]. Furthermore, by mapping into AutoMed's HDM language, we are then able to map the OWL-DL schemas into any of the existing modelling languages supported by AutoMed.

The remainder of this paper is structured as follows. Section 2 gives a brief description about the HDM. In Section 3, we show some of the representations of OWL-DL axioms in HDM and in Section 4, we show how previously defined transformation rules between relational and HDM schemas [16], and our newly defined rules between OWL-DL and HDM schemas, can be composed to give a mapping between relational and OWL-DL schemas. Finally, we state our conclusions in Section 5.

2 HDM Overview

In this Section, we provide a brief overview over the HDM and we refer the reader to [16] for full details. An HDM schema is a structure in which data may be held and is defined as follows:

▶ **Definition 1. HDM Schema** Given a set of $Names$ that we may use for modelling the real world, an HDM schema, S, is a triple $Nodes, Edges, Cons$ where:

- $Nodes \subseteq \{\langle\langle \mathsf{n_n} \rangle\rangle \mid n_n \in Names\}$ $Nodes$ is a set of nodes in the graph, each denoted by its name enclosed in double chevron marks.
- $Schemes = Nodes \cup Edges$
- $Edges \subseteq \{\langle\langle \mathsf{n_e}, \mathsf{s_1}, \ldots, \mathsf{s_n} \rangle\rangle \mid n_e \in Names \cup \{\_\} \wedge s_1 \in Schemes \wedge \ldots \wedge s_n \in Schemes\}$ $Edges$ is a set of edges in the graph where each edge is denoted by its name, together with the list of nodes/edges that the edge connects, enclosed in double chevron marks.
- $Cons \subseteq \{c(s_1, \ldots, s_n) \mid c \in Funcs \wedge s_1 \in Schemes \wedge \ldots \wedge s_n \in Schemes\}$ $Cons$ is a set of boolean-valued functions (constraints) whose variables are members of $Schemes$ and where the set of functions $Funcs$ forms the HDM constraint language. In this paper we only use the following:
  1. $inclusion(s_1, s_2) \equiv s_1 \subseteq s_2$
  2. $mandatory(s_1, \ldots, s_m, s) \equiv < s_1, \ldots, s_m > \rhd s$
  3. $unique(s_1, \ldots, s_m, s) \equiv < s_1, \ldots, s_m > \lhd s$
  4. $reflexive(s_1, s) \equiv s_1 \stackrel{\text{id}}{\rightarrow} s$

▶ Example 1. We list in here the contents of an example HDM schema that we shall later, in Figure 2, show to be equivalent to a relational schema. Note how the names of edges are sometimes given as the character '_' representing an unnamed edge.

$Nodes = \{$⟨⟨ug⟩⟩, ⟨⟨ug:ppt⟩⟩, ⟨⟨student⟩⟩, ⟨⟨student:name⟩⟩, ⟨⟨student:sid⟩⟩,
⟨⟨result:grade⟩⟩, ⟨⟨course⟩⟩, ⟨⟨course:code⟩⟩, ⟨⟨course:dept⟩⟩$\}$

$Edges = \{$⟨⟨_,ug,ug:ppt⟩⟩, ⟨⟨_,student,student:sid⟩⟩, ⟨⟨_,student,student:name⟩⟩,
⟨⟨result,student,course⟩⟩, ⟨⟨_,⟨⟨result,student,course⟩⟩,result:grade⟩⟩,
⟨⟨_,course,course:dept⟩⟩, ⟨⟨_,course,course:code⟩⟩$\}$

$Cons = \{$⟨⟨ug⟩⟩ $\lhd$ ⟨⟨_,ug,ug:ppt⟩⟩, ⟨⟨ug⟩⟩ $\rhd$ ⟨⟨_,ug,ug:ppt⟩⟩,
⟨⟨ug:ppt⟩⟩ $\rhd$ ⟨⟨_,ug,ug:ppt⟩⟩, ⟨⟨ug⟩⟩ $\subseteq$ ⟨⟨student⟩⟩,
⟨⟨student⟩⟩ $\lhd$ ⟨⟨_,student,student:sid⟩⟩, ⟨⟨student⟩⟩ $\rhd$ ⟨⟨_,student,student:sid⟩⟩,
⟨⟨student:sid⟩⟩ $\rhd$ ⟨⟨_,student,student:sid⟩⟩, ⟨⟨student⟩⟩ $\lhd$ ⟨⟨_,student,student:name⟩⟩,
⟨⟨student⟩⟩ $\rhd$ ⟨⟨_,student,student:name⟩⟩, ⟨⟨student⟩⟩ $\xrightarrow{\text{id}}$ ⟨⟨_,student,student:name⟩⟩,
⟨⟨student:name⟩⟩ $\rhd$ ⟨⟨_,student,student:name⟩⟩,
⟨⟨result:grade⟩⟩ $\rhd$ ⟨⟨_,⟨⟨result,student,course⟩⟩,result:grade⟩⟩,
⟨⟨result,student,course⟩⟩ $\lhd$ ⟨⟨_,⟨⟨result,student,course⟩⟩,result:grade⟩⟩,
⟨⟨course⟩⟩ $\lhd$ ⟨⟨_,course,course:dept⟩⟩, ⟨⟨course⟩⟩ $\rhd$ ⟨⟨_,course,course:dept⟩⟩,
⟨⟨course:dept⟩⟩ $\rhd$ ⟨⟨_,course,course:dept⟩⟩, ⟨⟨course⟩⟩ $\lhd$ ⟨⟨_,course,course:code⟩⟩,
⟨⟨course⟩⟩ $\rhd$ ⟨⟨_,course,course:code⟩⟩, ⟨⟨course⟩⟩ $\xrightarrow{\text{id}}$ ⟨⟨_,course,course:code⟩⟩,
⟨⟨course:code⟩⟩ $\rhd$ ⟨⟨_,course,course:code⟩⟩$\}$

3 Representing OWL-DL in HDM

We now discuss how OWL-DL axioms and facts may be represented in the HDM, and hence translated into other modelling languages. For conciseness, we only discuss those OWL-DL constructs listed in Table 1, which are sufficient to describe how the OWL-DL ontology depicted in Figure 1 can be translated into the HDM shown in Figure 1(b).

Table 1 HDM Representations of Some OWL-DL Axioms.

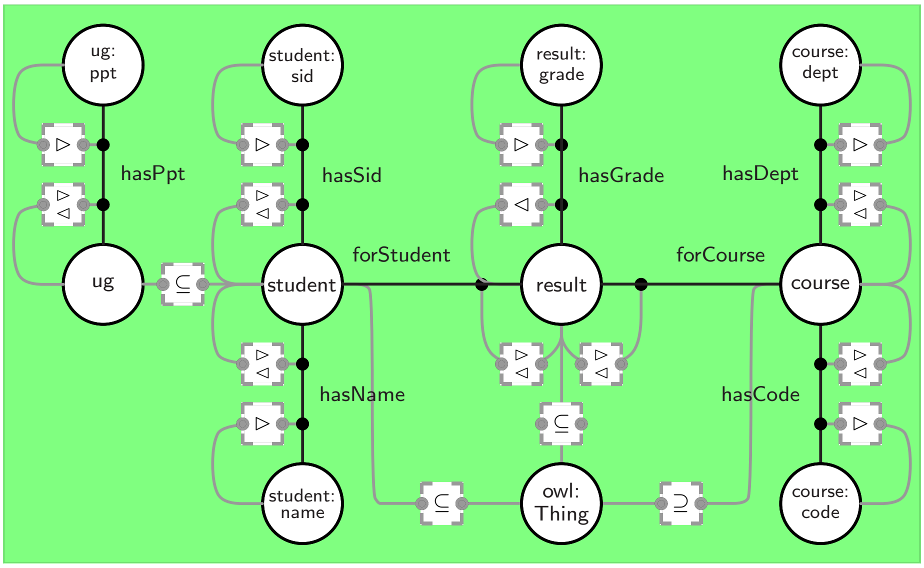
OWL-DL Name	DL Syntax	Scheme	HDM Representation
owl:Thing	$\top$	⟨⟨owl:Thing⟩⟩	Node ⟨⟨owl:Thing⟩⟩
owl:Nothing	$\bot$	⟨⟨owl:Nothing⟩⟩	Node ⟨⟨owl:Nothing⟩⟩
Class	C	⟨⟨C⟩⟩	Node ⟨⟨C⟩⟩
SubClassOf (C_1 C_2)	$C_1 \sqsubseteq C_2$	⟨⟨$\sqsubseteq$, $\mathsf{C_1}$, $\mathsf{C_2}$⟩⟩	Constraint ⟨⟨$\subseteq$, ⟨⟨$\mathsf{C_1}$⟩⟩, ⟨⟨$\mathsf{C_2}$⟩⟩⟩⟩
ObjectProperty	P	⟨⟨P, $\mathsf{C_1}$, $\mathsf{C_2}$⟩⟩	Edge ⟨⟨P, ⟨⟨$\mathsf{C_1}$⟩⟩, ⟨⟨$\mathsf{C_2}$⟩⟩⟩⟩
FunctionalProperty	T $\sqsubseteq \leq 1P$	⟨⟨P, $\mathsf{C_1}$, $\mathsf{C_2}$, func⟩⟩	Edge ⟨⟨P, ⟨⟨$\mathsf{C_1}$⟩⟩, ⟨⟨$\mathsf{C_2}$⟩⟩⟩⟩ Constraint ⟨⟨$\rhd$, ⟨⟨$\mathsf{C_1}$⟩⟩, ⟨⟨$\mathsf{C_2}$⟩⟩⟩⟩ Constraint ⟨⟨$\lhd$, ⟨⟨$\mathsf{C_1}$⟩⟩, ⟨⟨$\mathsf{C_2}$⟩⟩⟩⟩

▶ Example 2. Consider the OWL-DL schema illustrated in Figure 1(a) which represents concepts in a university Universe of Discourse and its relationships. Using the representations of OWL-DL axioms shown in Table 1, we present an equivalent HDM schema for the OWL-DL schema depicted in Figure 1(b). All classes such as owl:Thing, student, and course were represented as HDM Nodes. Functional properties such as hasName, hasSid, and hasPpt were represented as HDM edges with mandatory ($\rhd$) and unique ($\lhd$) constraints. The SubClassOf axioms such as (student $\sqsubseteq$ owl:Thing), (course $\sqsubseteq$ owl:Thing), and (ug $\sqsubseteq$ student) were represented as an inclusion constraint ($\subseteq$). Note that in the HDM diagram, HDM nodes are

$\text{student} \sqsubseteq \text{owl:Thing} \sqcap 1\ \text{hasName.name} \sqcap 1\ \text{hasSid.sid}$
$\text{course} \sqsubseteq \text{owl:Thing} \sqcap 1\ \text{hasCode.code} \sqcap 1\ \text{hasDept.dept}$
$\text{ug} \sqsubseteq \text{student} \sqcap \exists\ \text{hasPpt.ppt}$
$\text{result} \sqsubseteq \text{owl:Thing}$
$\text{result} \equiv 1.\text{forStudent}$
$\text{result} \equiv 1.\text{forCourse}$
$1.\text{hasGrade} \sqsubseteq \text{result}$
$\text{grade} \sqsubseteq \exists\ \text{hasGrade}$

$\top \sqsubseteq \text{hasGrade.grade}$
$\top \sqsubseteq \text{hasGrade}^{-}.\text{result}$
$\top \sqsubseteq \text{forStudent.student}$
$\top \sqsubseteq \text{forStudent}^{-}.\text{result}$
$\top \sqsubseteq \text{forCourse.course}$
$\top \sqsubseteq \text{forCourse}^{-}.\text{result}$
$\text{student} \sqcap \text{course} \sqsubseteq \bot$

(a) An OWL-DL schema of the student-course knowledge base



(b) HDM representation of the OWL-DL schema

Figure 1 An OWL-DL schema and its equivalent HDM schema.

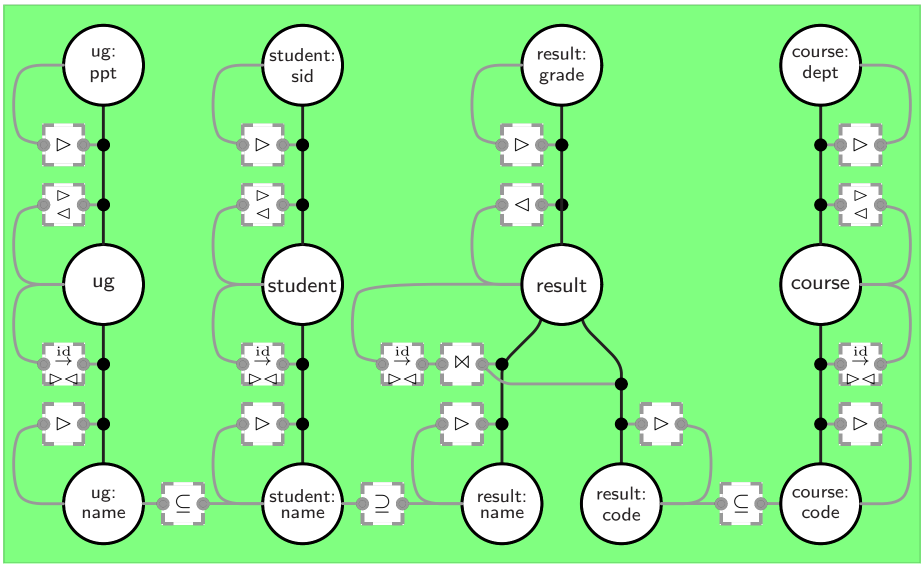
represented by white circles with thick outlines, and HDM edges are represented by thick black lines. The HDM constraint language is represented by grey dashed boxes connected by grey lines to the nodes and edges to which the constraint applies. Edges pass through black circles in a straight line, hence any edge or constraint applying to an edge meets that edge at an angle.

4 OWL-DL Knowledge Bases Transformation using the BAV Model

In [16], five general purpose equivalence mappings that allow the transformation between different modelling languages were proposed namely: Inclusion Merge, Identity Node Merge, Unique-Mandatory Redirection, Identity Edge Merge, and Node Reidentify. In this paper, we show how we can use them to transform between a knowledge model, the OWL-DL shown in Figure 1(a) and a data model, the relational shown in Figure 2(a). Taking the HDM equivalent schemas of these OWL-DL and relational schemas illustrated in Figure 1(b) and Figure 2(b) respectively and applying some of these BAV-defined mappings, we were able to transform the HDM relational schema into an HDM OWL-DL schema through 21 steps as shown below.

ug(name,ppt)	ug.name → student.name
student(name,sid)	result.name → student.name
course(code,dept)	result.code → course.code
result(code,name,grade?)	

(a) Relational schema for the student-course database



(b) HDM representation of the relational database schema

Figure 2 A relational schema and its equivalent HDM schema.

The first 5 steps are identical to those in transforming relational to ER HDM schemas shown in [16]. Applying these 5 steps results in Figure 3. When transforming from a key based model (such as relational) and a knowledge model that does not provide means to define keys (such as OWL-DL), we must overcome some fundamental differences which require, in our example, extending the object identifiers (OIDs) of ⟨⟨student⟩⟩, ⟨⟨course⟩⟩, and ⟨⟨result⟩⟩ respectively as illustrated in steps 6-8. The transformations associated with step 6 are illustrated in Example 3. Steps 7 and 8 are similar to step 6 thus, we do not explain them here. Step 9 is again similar to step 7 in relational and ER HDM schemas conversion given in [16]. Steps 10-13 illustrate adding the ⟨⟨owl:Thing⟩⟩ node along with three inclusion constraints (⊆) to it from the ⟨⟨student⟩⟩, ⟨⟨course⟩⟩, and ⟨⟨result⟩⟩ nodes. Finally, all we need to do to obtain the OWL-DL HDM schema is to rename the edges as shown in steps 14-21. The result of these 21 steps is the schema shown in Figure 1(b).

1. inclusion_merge (⟨⟨student:name⟩⟩,⟨⟨_,result:name, result⟩⟩)
2. inclusion_merge (⟨⟨course:code⟩⟩, ⟨⟨_, result:code, result⟩⟩)
3. identity_node_merge (⟨⟨_,ug:name, ug⟩⟩)
4. unique_mandatory_redirection (⟨⟨_, student:name, result⟩⟩, ⟨⟨_, student:name, student⟩⟩)
5. unique_mandatory_redirection (⟨⟨_, course:code, result⟩⟩, ⟨⟨_, course:code, course⟩⟩)
6. extend_OID (⟨⟨student⟩⟩ $\overset{\text{id}}{\rightarrow}$⟨⟨_,student,student:name⟩⟩)
7. extend_OID (⟨⟨course⟩⟩ $\overset{\text{id}}{\rightarrow}$⟨⟨_,course,course:code⟩⟩)
8. extend_OID (⟨⟨result⟩⟩ $\overset{\text{id}}{\rightarrow}$⟨⟨_,result,student:name⟩⟩ ⋈ ⟨⟨_,result,course:code⟩⟩)

9. move_dependants (⟨⟨student:name⟩⟩, ⟨⟨student⟩⟩, ⟨⟨_,student:name, student⟩⟩)
10. addNode (⟨⟨owl:Thing⟩⟩)
11. addCons (⟨⟨student⟩⟩ $\subseteq$ ⟨⟨owl:Thing⟩⟩)
12. addCons (⟨⟨course⟩⟩ $\subseteq$ ⟨⟨owl:Thing⟩⟩)
13. addCons (⟨⟨result⟩⟩ $\subseteq$ ⟨⟨owl:Thing⟩⟩)
14. renameEdge(⟨⟨_,course,course:dept⟩⟩,⟨⟨hasDept,course,course:dept⟩⟩)
15. renameEdge(⟨⟨_,course,course:code⟩⟩,⟨⟨hasCode,course,course:code⟩⟩)
16. renameEdge(⟨⟨_,result,result:grade⟩⟩,⟨⟨hasGrade,result,result:grade⟩⟩)
17. renameEdge(⟨⟨_,result,course⟩⟩,⟨⟨forCourse,result,course⟩⟩)
18. renameEdge(⟨⟨_,result,student⟩⟩,⟨⟨forStudent,result,student⟩⟩)
19. renameEdge(⟨⟨_,student,student:sid⟩⟩,⟨⟨hasSid,student,student:sid⟩⟩)
20. renameEdge(⟨⟨_,student,student:name⟩⟩,⟨⟨hasName,student,student:name⟩⟩)
21. renameEdge(⟨⟨_,ug,ug:ppt⟩⟩,⟨⟨hasPpt,ug,ug:ppt⟩⟩)

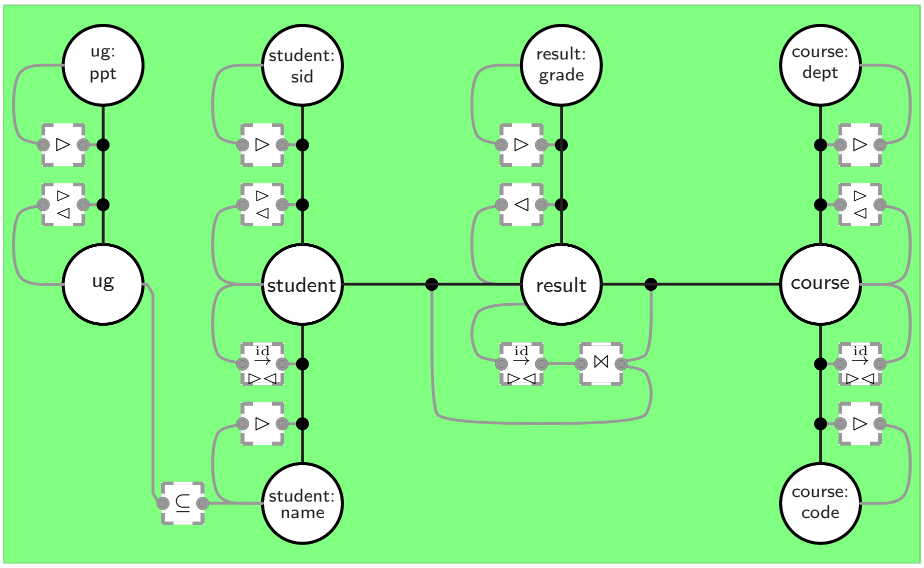


Figure 3 Intermediate HDM schema in relational to OWL-DL conversion, after steps 1–5.

▶ Example 3. Transformations associated with step 6:
1. inverse_identity_node_merge(⟨⟨student⟩⟩,⟨⟨student:oid⟩⟩)
2. deleteCons(⟨⟨student⟩⟩ $\xrightarrow{\text{id}}$⟨⟨_,student,student:oid⟩⟩)
3. node_reident(⟨⟨student⟩⟩, { $\langle x, y\rangle$ | ⟨o,x⟩ ∈ ⟨⟨_,student,student:oid⟩⟩∧⟨o,y⟩ ∈ ⟨⟨_,student,student:name⟩⟩})
4. deleteCons(⟨⟨student⟩⟩ $\xrightarrow{\text{id}}$⟨⟨_,student,student:name⟩⟩)
5. deleteCons(⟨⟨student⟩⟩ $\lhd$ ⟨⟨_,student,student:name⟩⟩)
6. deleteCons(⟨⟨student⟩⟩ $\rhd$ ⟨⟨_,student,student:name⟩⟩)
7. deleteCons(⟨⟨student:oid⟩⟩ $\lhd$ ⟨⟨_,student,student:name⟩⟩)
8. deleteCons(⟨⟨student:oid⟩⟩ $\rhd$ ⟨⟨_,student,student:name⟩⟩)
9. contractEdge(⟨⟨_,student,student:name⟩⟩)
10. contractNode(⟨⟨student:name⟩⟩)
11. renameNode(⟨⟨student:oid⟩⟩, ⟨⟨student:name⟩⟩)

Note that the inverse of identity node merge in transformation 1 generates a new node ⟨⟨student:oid⟩⟩, connected to ⟨⟨student⟩⟩ by a new edge ⟨⟨_,student,student:oid⟩⟩. Transformations 2-4 have the net effect of repopulating the ⟨⟨student⟩⟩ node with values of the ⟨⟨student:name⟩⟩ attribute, and deleting the keys from name and oid. Transformations 5-11 delete the ⟨⟨student:name⟩⟩ node (with its associated constraints and edge) and rename the node ⟨⟨student:oid⟩⟩ with ⟨⟨student:name⟩⟩.

5 Conclusions

In this paper, we have defined how schemas expressed in OWL-DL can be translated into equivalent schemas in HDM. We have also given an example, using the AutoMed system, that shows how to map between HDM OWL-DL schemas and HDM relational schemas which results in a bidirectional mapping between OWL-DL and relational schemas, and vice versa. Our future work will expand our approach by defining schemas expressed in other knowledge modelling languages such as OWL 2 in HDM. This might include extending the HDM constraint language in order to accomodate the richness of such modelling languages.

References

1. Calvanese, D., Giuseppe, G., and Lenzerini, M., 2001. Ontology of Integration and Integration of Ontologies. In: *DL*.
2. Noy, N., 2004. Semantic Integration: A Survey of Ontology-Based Approaches. *SIGMOD Record*, 33(4), pp. 65–70.
3. Kalfoglou, Y. and Schorlemmer, M. 2003. Ontology Mapping: The State of the Art. *The Knowledge Engineering Review*. 18(1), Publisher: Cambridge University Press, pp. 1-31.
4. Lenzerini, M. 2002. Data Integration: A Theoretical Perspective. In *Proc. PODS'02*, pp. 233-246. ACM.
5. Madhavan, J. and Halevy, A.Y. 2003. Composing Mappings Among Data Sources. In *Proc. VLDB'03*, pp. 572-583.
6. Boyd, M., Kittivoravitkul, S., Lazanitis, C., McBrien, P. and Rizopoulos, N. 2004. AutoMed: A BAV Data Integration System for Heterogeneous Data Sources. In *Proc. CAiSE'04*. LNCS.
7. Noy, N., 2003. What Do We Need for Ontology Integration on the Semantic Web. In: *ISWC'03*. Sanibel Island, Florida.
8. Udrea, O., Getoor, L., and Miller, R. 2007. Leveraging Data and Structure in Ontology Integration. In: *SIGMOD '07*. Beijing, China. pp. 449-460.
9. Lv, Y., and Xie, C. 2010. A Framework for Ontology Integration and Evaluation. *IEEE*. pp. 521-524.
10. Jimenez-Ruiz, E., Grau, B., Horrocks, I., and Berlanga, R. 2009. Ontology Integration Using Mappings: Towards Getting the Right Logical Consequences. In: *Proc. of ESWC'09*.
11. N.F. Noy and M. Musen. 2000. PROMPT: Algorithm and Tool for Automated Ontology Merging and Alignment. In *Proc. of AAAI'00*, Austin, TX, USA.
12. Smith, A., Rizopoulos, N., McBrien, P. 2008. AutoMed Model Management. In:*Proc. of ER'08*. LNCS, vol. 5231, pp. 542–543. Springer, Heidelberg (2008).
13. W3C. RDF Vocabulary Description Language 1.0: RDF Schema. W3C Recommendation 10 February 2004. Available at: http://www.w3.org/TR/rdf-schema/
14. W3C. Web Ontology Language Guide. W3C Recommendation 10 February 2004. Available at: http://www.w3.org/TR/2004/REC-owl-guide-20040210/
15. McBrien, P. and Poulovassilis, A. 1999. A Uniform Approach to Inter-Model Transformations. In: *Proc. CAiSE'99*. p.333-348, June 14-18, 1999.
16. Boyd, M. and McBrien, P. 2005. Comparing and Transforming Between Data Models via Intermediate Hypergraph Data Model. pp. 69-109, Springer-Verlag, ISBN-13 978-3-540-31001-3, ISSN 0302-9743.