Guiding Reification in OWL through Aggregation

Paula Severi[1], José Fiadeiro[1], and David Ekserdjian[2]

[1] Department of Computer Science
[2] Department of History of Art and Film
University of Leicester, United Kingdom

Abstract. We put forward a methodological approach aimed at guiding ontology modellers in choosing which relations to reify. Our proposal is based on the notion of aggregation as used in conceptual modelling approaches for representing situations that, normally, would require non-binary relations or complex integrity constraints. The feedback received from using the method in a real-word situation is that if offers a more controlled use of reification and a closer fit between the resulting ontology and the application domain as perceived by an expert.

1 Introduction

A well-known limitation of OWL 2 (Web Ontology Language) is that only binary relations between classes can be represented [1–3]. In practice, relations of arbitrary arity are quite common and they have to be represented in OWL in an indirect way by coding them as classes[3]. In the literature of Description Logic (DL) [4], the class codifying an n-ary relation ρ is called *the reification of ρ* [4].

As any codification, reification requires extra work in addition to 'simple' modelling, which can make it quite impractical (and unintuitive), especially when performed by people who are not 'experts': extra classes, predicates, individuals and axioms [5] need to be introduced and, as the number of classes increases, ontologies can become very difficult to read and understand, mainly because this additional information (which is encoded) is not directly visible. That is, there is a mismatch between the layer of abstraction at which domain modellers work and that of the representation where information is encoded, which is particularly harmful when we want to extend and reuse ontologies.

In this paper, we propose the use of a methodological construction that has been devised many years ago in the database community, which is based on the notion of aggregation as proposed in [6]. Aggregation is an abstraction that

---

[3] Similarly for RDF (Resource Description Framework)

[4] The term *reification* can have several meanings and uses in Logic in general, and the Semantic Web in particular. In this paper, we use it as a synonym for encoding n-ary relations as classes. We do not use it to refer to the usage of RDF as a metalanguage to describe other logics, or in situations in which a statement can be assigned a URI and treated as a resource, or the use of classes as individuals.

was offered therein for increasing the "understandibility of relational models by the imposition of additional semantic structure". Although, in ontologies, the technical problems that arise are not necessarily the same as those of relational databases, the methodological issues are similar in the sense that the solution to our problem lies first of all in helping modellers to conceptualize the real world in a way that can lead to a better representation, and then offering them a mechanism for implementing these semantic structures in ontologies. By 'better' we mean a more controlled use of reification and a closer fit between the resulting ontology and the real-world domain as perceived by an expert.

Having this in mind, we start by motivating the problem using the case study that led us to investigate the representation of n-ary relationships — an ontology of 16th-century Italian altarpieces. In Section 3, we discuss a formal, set-theoretical, notion of aggregation and the way that it can be implemented in ontologies through reification. Then, in Section 4, we discuss how aggregation as a modelling abstraction can be used effectively in a number of situations that are recurrent in domains such as that of altarpieces.

2   Motivation

In order to illustrate some of the problems that may arise from the limitations of having to encode n-ary relations through reification and the method that we propose to minimize them, we use the Ontology of Altarpieces [7] — a joint project between the Departments of Computer Science and History of Art and Film at the University of Leicester. This case study is a good example of a domain in which n-ary relations arise quite naturally and frequently.

Suppose that we want to express the following knowledge as produced in natural language by an art expert:

1. *The altarpiece painted by Raphael called "Sistine Madonna"*[5] *has the figure of the Virgin on it.*
2. *The altarpiece painted by Raphael called "Sistine Madonna" has the figure of the Christ on it.*
3. *The altarpiece painted by Raphael called "The Marriage of the Virgin"*[6] *has the figure of the Virgin on it.*

The above sentences can be represented by a ternary relation $hasFigure$ between the sets $Painters$, $PictureNames$ and $Figures$.

$$hasFigure = \{(\mathsf{raphael}, \mathsf{sistine\ madonna}, \mathsf{virgin}),$$
$$(\mathsf{raphael}, \mathsf{sistine\ madonna}, \mathsf{christ}),$$
$$(\mathsf{raphael}, \mathsf{marriage\ of\ virgin}, \mathsf{virgin})\}$$

Figure 1 shows an entity relation (ER) diagram for the relationship $hasFigure$ of which the set above is an extension. This relation cannot be represented in

---

[5] See http://en.wikipedia.org/wiki/Sistine_Madonna.
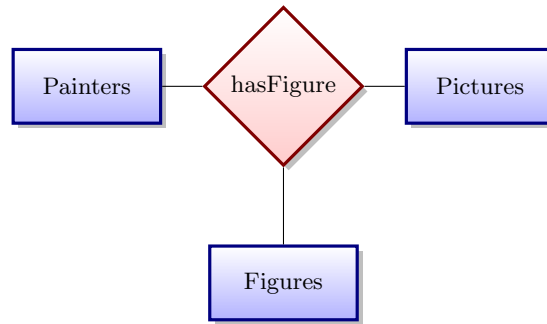[6] See http://en.wikipedia.org/wiki/The_Marriage_of_the_Virgin_(Raphael).

Fig. 1. ER diagram: hasFigure as a relationship of arity 3

OWL unless we code it as a class $C_{hasFigure}$ of individuals that represent the tuples — the reification of the relation [4]. For example, we create an individual r_1 that represents the tuple

$$\langle \mathsf{raphael}, \mathsf{sistine\ madonna}, \mathsf{virgin} \rangle$$

and we connect r_1 to each component in the tuple using the role names painter, picturename and figure as shown in Figure 2.
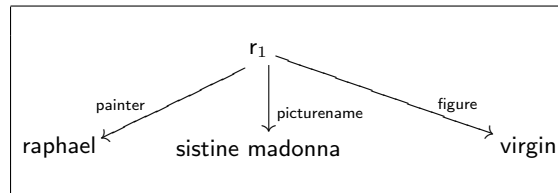

Fig. 2. Connecting r_1 to the components of the tuple

However, reifying *hasFigure* is not necessarily the right decision that a modeller should make. This is because Figure 1 shows the relationship *hasFigure* isolated from the rest of the ontology. A diagram that shows other relationships between these entities in a wider conceptual model of the domain of altarpieces is depicted in Figure 3 In this diagram, we can see another relationship involving *Painters* and *PictureNames* and a number of 'descriptive attributes' (functional relationships involving a data type) that apply to that relationship. Naturally, one cannot take a blind approach to the representation of these aspects of the domain and reify relations as they come: the complexity of the ontologies thus generated would be even beyond skilled computer scientists, let alone domain experts.

Suppose we want to express that *The Virgin is holding Christ in the altarpiece called "Sistine Madonna" by Raphael.* To represent the above sentence, we need a relation *holds* of arity 4 where

$$holds = \{(\mathsf{raphael}, \mathsf{sistine\ madonna}, \mathsf{virgin}, \mathsf{christ})\}$$

In the Ontology of Altarpieces we have about 20 relations of arity 3 such as *hasFigure* and more than 4000 relations such as *holds* of arity 4. It would not make sense to blindly reify all the relations of arity strictly greater than 2. Given that each relation may have an average of 1000 tuples, doing so would mean 1000 individuals for coding the tuples and 1000 x 4 pairs connecting the individuals with their components. If we consider that the details of those figures and other attributes of the altarpieces need to be represented, it is easy to see that the whole ontology would become quite unwieldy.

In other words, basic questions that a modeller needs to consider very carefully is: "Can I reduce the number of reifications in my ontology?", "Which relations are more convenient to reify?". Our answer in this paper is given in methodological terms, inspired by similar problems faced by the relational database community 30 years ago.
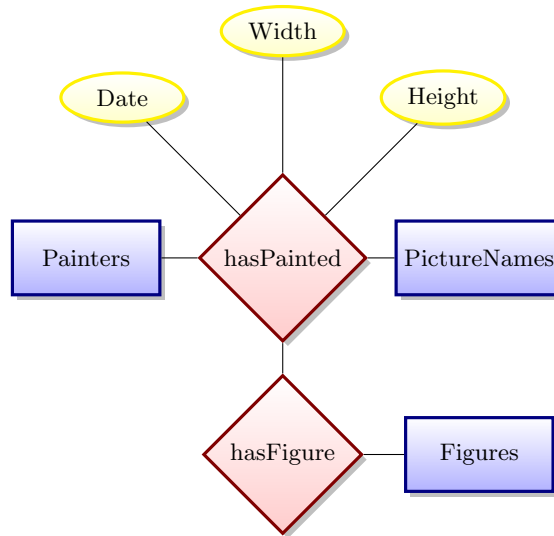

Fig. 3. ER diagram: how the hasFigure relationship interacts with other relationships

The examples presented in this paper are very simple and try to extract the main concepts behind the method. However, we have applied aggregations to more complex relations in the Ontology of Altarpieces.

3   Aggregation in Set Theory vs Reification in OWL

Aggregation as defined in [6] refers to an abstraction in which a relationship between objects is regarded as a higher-level object. The intention, as stated therein, was to adapt cartesian product structures (as proposed by T. Hoare for record structures in programming languages) to be used in the context of relational models. Although a formal definition was not given as a semantics for the abstraction, we found it useful to advance one so that, on the one hand, we can be precise about our usage of the term and, on the other hand, we can relate it to the mechanism of reification. Throughout the paper, we use the Greek alphabet for entities that we define in Set Theory.

Definition 1. *Let $\Delta_1, \Delta_2 \subseteq \Delta$ and $\rho \subseteq \Delta_1 \times \Delta_2$ be a binary relation. An aggregation of ρ is a set $\Delta_\rho \subseteq \Delta$ together with two (total) functions π_1 and π_2 (called* projections*) from Δ_ρ to Δ_1 and Δ_2, respectively, such that:*

1. *For all $r \in \Delta_\rho$, $\langle \pi_1(r), \pi_2(r) \rangle \in \rho$ — i.e., there is no 'junk' in Δ_ρ.*
2. *For all $\langle x_1, x_2 \rangle \in \rho$, there exists $r \in \Delta_\rho$ such that $\pi_1(r) = x_1$ and $\pi_2(r) = x_2$ — the aggregation covers the whole relation ρ.*
3. *For all $r_1, r_2 \in \Delta_\rho$, if $\pi_1(r_1) = \pi_1(r_2)$ and $\pi_2(r_1) = \pi_2(r_2)$ then $r_1 = r_2$ — i.e., there is no 'confusion': every tuple of the relation has a unique representation as an aggregate.*

It is trivial to prove the following result:

Proposition 1. *Δ_ρ is isomorphic to ρ.*

That is, an aggregation is indeed offering a 'faithfull' representation of the relation. We denote this isomorphism by Ψ_ρ or just Ψ where $\Psi(r) = \langle \pi_1(r), \pi_2(r) \rangle$. Its inverse defines the encoding of the relation, i.e. it assigns to each tuple in the relation ρ a unique element (aggregate) of the set Δ_ρ.

Informally, the reification of a relation ρ is a class C_ρ representing the tuples of ρ [4, 8]. This representation should be as close as possible to the relation itself in order to avoid any possible mismatch between the representation and the model that the expert has in mind. In order to be able to analyse this relationship, we have found it useful to provide a concrete definition of how we are using the notion of reification:

Definition 2. *Let $\Delta_1, \Delta_2 \subseteq \Delta$ and $\rho \subseteq \Delta_1 \times \Delta_2$ be a binary relation. A* reification *of ρ in OWL is a concept C_ρ together with two roles P_1 and P_2, called* projections*, two domains D_1 and D_2, and the following collection of axioms:*

$$
\begin{array}{ll}
\text{(proj func)} & \top \sqsubseteq\, \leq 1P_1 \sqcap\, \leq 1P_2 \\[2pt]
\text{(proj domain)} & \exists P_1.\top \sqcap \exists P_2.\top \sqsubseteq C_\rho \\[2pt]
\text{(proj range)} & \top \sqsubseteq \forall P_1.D_1 \sqcap \forall P_2.D_2 \\[2pt]
\text{(proj totality)} & C_\rho \sqsubseteq \exists P_1.D_1 \sqcap \exists P_2.D_2 \\[6pt]
\text{(unique rep)} & C_\rho\ \mathsf{hasKey}(P_1, P_2)
\end{array}
$$

These definitions can be generalized to relations of arbitrary arity. We can now define more precisely how a reification relates to the relation:

Definition 3. *Let $\rho \subseteq \Delta_1 \times \Delta_2$ be a binary relation. Given an interpretation I, we say that the reification $(C_\rho, D_1, D_2, P_1, P_2)$ is* faithful *to ρ in relation to I iff $D_1^I = \Delta_1$, $D_2^I = \Delta_2$, and (C_ρ^I, P_1^I, P_2^I) is an aggregation of ρ.*

Unfortunately, the axioms that are part of the reification are not sufficient to guarantee that it is faithfull to ρ in relation to every interpretation:

- The first four axioms state that the role names P_1 and P_2 are total functions from C_ρ to D_1 and D_2, respectively. However, a limitation of OWL is that the reasoner does not show any inconsistency if we forget to define P_1 or P_2 for some element of C_ρ (see [9]). This type of mistake could obviously avoided if OWL provided us with relations of arity n.
- The axiom (unique rep) states that two named individuals in C_ρ that have the same projections should be equal. This axiom is weaker than the third condition of Definition 1 in the sense that unicity of the representation is not enforced for *all* individuals but only on those that are explicitly *named* in the ontology. This is because the hasKey constructor of OWL-2 is a weak form of key representation (so-called "EasyKey constraints") that is valid only for individuals belonging to the Herbrand Universe [10].

Summarising, reification is not only hard work (in the sense that it requires the modeller to introduce a number of roles and axioms that are 'technical', i.e. more related to the limitations of the formalism and less specific to the domain of application) but also prone to errors. Essentially, errors may arise if the modeller forgets to enforce the properties that cannot be expressed in OWL: totality, 'no junk' or coverage.

Notice that, in the specific case of binary relations, we can add an atomic role R to the ontology and add the following axiom to the reification, which corresponds to the first condition of Definition 1 — 'no junk':

$$(R\text{-contains}) \quad (P_1)^{-1} \circ P_2 \sqsubseteq R$$

This axiom states that the relation R can be recovered from the reification C_ρ through the projections P_1 and P_2. In this case, faithfulness would require that $R^I = C_\rho^I$. The ability to work with an atomic role R also has methodological advantages as illustrated in the next section.

Also note that, in the binary case, the converse of (R-contains), which would correspond to the second condition of Definition 1, is as follows

$$(R\text{-inclusion}) \quad R \sqsubseteq (P_1)^{-1} \circ P_2$$

However, this axiom cannot be expressed in OWL in the above form [7] because the right-hand side of the inclusion is not a role name (see [3]).

---

[7] This is not a proof that the axiom cannot be expressed in the logic which would be more involved.

These shortcomings show why methodological support is necessary when using reification in OWL: one should make sure that abstraction mechanisms are available through which a modeller can keep a close fit between the representation and the domain and that these mechanisms are supported, as much as possible, by tools. The aim of the techniques put forward in the next section is precisely to overcome the gap that may exist between the perception of the relationships that exist in the domain of discourse and the use of reification to encode them in OWL.

4   Guiding the Use of Reifications in Ontologies

In this section, we put forward a methodological approach aimed at guiding the modeller in the use of reification. The method is based on the usage of the semantic primitive of aggregation as used in conceptual modelling precisely for representing situations that, normally, would require non-binary relations or complex integrity constraints [11]. We illustrate the approach with some examples that are representative of the situations that we have encountered in the altarpieces project.
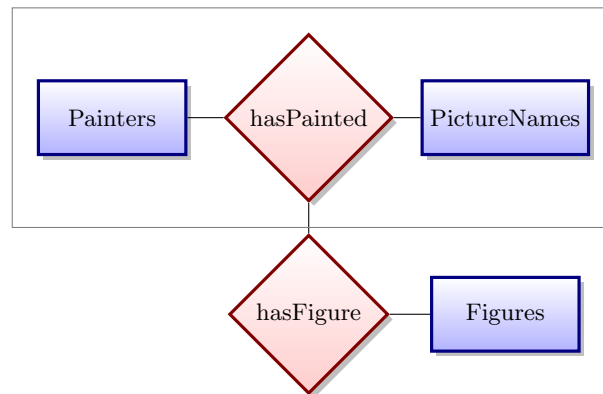

Fig. 4. ER diagram: hasPainted as an aggregate of hasFigure

4.1   Relationships amongst Relationships

A recurrent situation in database modelling is the use of aggregation in order to reduce certain ternary relationships to binary ones [11]. Using ER diagrams, the method can be explained in terms of evolving situations such as the one depicted in Figure 1 to the one depicted in Figure 4. More specifically, the method consists in identifying a binary relationship — *hasPainted* — such that the ternary

relationship — $hasFigure$ — can be expressed as a binary relationship between the aggregation of the former — $hasPainted$ — and the remaining domain — $Figures$. The aggregation of a relationship is indicated by the box that surrounds the relationship diagram. Following this method, instead of reifying the whole relation $hasFigure$, we reify $hasPainted$. Since $hasPainted$ is a binary relation, we represent it by the role hasPainted and consider the reification of $hasPainted$ as in Definition 2. For this, we introduce the class Altarpieces as the reification $C_{hasPainted}$ and the roles painter and picturename as the projections. The relation $hasFigure$ is represented as an object property whose domain is $C_{hasPainted}$ and whose range is Figures as shown in Figure 5.
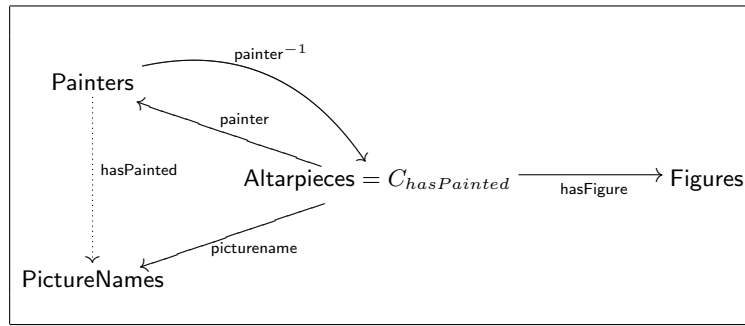

Fig. 5. Representation of hasFigure that considers hasPainted as an aggregate

We have chosen the simplest example from the ontology to illustrate the method. In this case it is clear from the use of the noun in the sentences in Section 2 that we should have chosen to model altarpieces as a class Altarpieces right from the start. The point is that, from the point of view of conceptual modelling, altarpieces are an aggregation of a relation: art experts identify altarpieces precisely through the name of the painter and the designation of the picture. Therefore, the class Altarpieces corresponds, in a natural way, to the reification of the relation $hasPainted$. In the case of other examples in our ontology, for instance the relations $holds$ and $wears$, among others, the class does not arise so naturally (indeed, they do not correspond to nouns), which explains why using reification to represent them is somewhat artificial, i.e. driven by technical, not conceptual concerns.

Definition 4. *Let $\Delta_1, \Delta_2, \Delta_3 \subseteq \Delta$, $\rho \subseteq \Delta_1 \times \Delta_2$ be a binary relation and $\rho' \subseteq \Delta_1 \times \Delta_2 \times \Delta_3$ be a ternary relation. We say that ρ participates in ρ' if the following condition is satisfied:*

- *For all $x \in \Delta_1, y \in \Delta_2, z \in \Delta_3$, $(x,y) \in \rho$ whenever $(x,y,z) \in \rho'$.*

If ρ participates in ρ' then ρ' "can be seen" as a binary relation between the aggregation Δ_ρ and Δ_3.

The relationship *hasPainted* 'participates' in the relationship *hasFigure* since the following constraint is satisfied:

$$\text{if } (x, y, z) \in hasFigure \text{ then } (x, y) \in hasPainted. \tag{1}$$

The above constraint is enforced in OWL by the axiom that states that the domain of hasFigure is $C_{hasPainted}$, and the axiom (hasPainted-contains) from the binary-relation extension of Definition 2 (see Figure 5).

The method that we propose for guiding reification consists in analysing which relations participate in other relations: if ρ *participates in* ρ' then, instead of reifying the whole relation ρ', we should consider reifying the participating relation ρ and represent ρ' as a role whose domain is C_ρ. If ρ participates in yet another relation, say ρ'', that relation does not need to be reified either but reuse instead the reification of ρ. Indeed, *hasPainted* participates in many relations other than *hasFigure* — e.g. *hasField*, for representing polyptych altarpieces that have many fields. All the corresponding relations can be represented in OWL as object properties whose domain is $C_{hasPainted}$ as in Figure 5.

Another important aspect of this representation (which is another reason why it is better than the reified ternary relation) is that we now have the relation *hasFigure* represented as a property hasFigure and not as a class $C_{hasFigure}$. Reifications represent properties but they cannot be used in the syntax as properties because they are actually classes. We cannot use constructors for roles on $C_{hasFigure}$ such as composition, quantification or transitive closure, which may restrict the ability of the modeller to capture important aspects of the domain. Whilst the representation of *hasFigure* as a property allows us to use the role name hasFigure in quantifications or in compositions. For instance, we can use an existential quantifier over the role hasFigure to express that all altarpieces must have some religious figure on it as follows:

$$\text{Altarpieces} \sqsubseteq \exists \text{hasFigure.Religious}$$

4.2 Descriptive attributes

Another related methodological guideline for the use of reification arises from what in [11] are called descriptive attributes. Descriptive attributes are used to record information about a relationship rather than about one of the participating entities, again using an aggregation. From a conceptual modelling point of view, they allow us to capture typical situations in which a functional dependency exists on a ternary relation as an attribute of the aggregation of a binary relation. For example, it would be intuitive to represent *height*, *width* and *date* in Figure 6 as descriptive attributes associated with the relationship *hasPainted*.

Definition 5. *Let $\rho \subseteq \Delta_1 \times \Delta_2$ and $\rho' \subseteq \Delta_1 \times \Delta_2 \times \Delta_3$. We say that ρ' is descriptive attribute of ρ if the following conditions holds:*

1. *ρ' is a function from $\Delta_1 \times \Delta_2$ into Δ_3.*
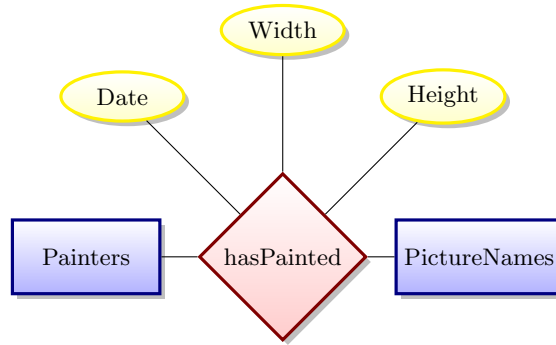2. *ρ participates in ρ' (see Definition 4).*

Fig. 6. ER diagram: hasPainted and descriptive attributes associated with it

This property is, indeed, satisfied by *height*:

1. There is a functional dependency between the height of the altarpiece and the pair given by the painter and the picture name. In other words, the ternary relation *height* is actually a function

$$height \in Painters \times PictureNames \rightarrow Int$$

2. *hasPainted* participates in *height*, i.e.:

$$\text{if } (x, y, z) \in height \text{ then } (x, y) \in hasPainted. \tag{2}$$

Given that descriptive attributes involve a participating relation, the methodological guidelines that we discussed in 4.1 suggest that descriptive attributes be represented as (functional) roles of the reification of the participating relation. For instance, using the reification $C_{hasPainted}$, the descriptive attribute *height* can be represented in OWL by a data type property height and two axioms

$$\top \sqsubseteq \leq 1.\text{height}$$

$$\geq 1\text{height} \sqsubseteq C_{hasPainted}$$

The constraints associated with the descriptive attribute *height* are deduced from the above two axioms and the axiom (hasPainted-contains).

5   Related Work and Concluding Remarks

The use of conceptual modelling primitives in the context of ontologies is not new. For instance, [12] and [13] show how to transform ER diagrams into Description Logic. However, this transformation does not include relationships involving relationships or descriptive attributes as illustrated in Section 4, nor does it address aggregation as a modelling abstraction. A paper that focuses specifically on

aggregation is [14]. However, the author represents aggregations using union of classes, which does not correspond in any way to their original meaning [6]. Our use of aggregation (based on cartesian products) adheres to its use in databases and explores its methodological advantages for conceptual modelling [11].

Our approach is also related with proposals that, like [15], put forward patterns for representing relations $\rho \subseteq A \times B \times C$. The third case of Pattern 1 in that note does the reification of the whole relation and the remaining cases do the reification of $B \times C$ and represent ρ as a property whose range is the reification $C_{B \times C}$. Our method is based on semantic abstractions and, therefore, goes beyond simple patterns. In fact, it deepens the study of these patterns in the sense that it guides the application of reification by the identification of relations that, like *hasPainted*, participate in other relations.

On the subject of representing non-binary relations, [16] provides a trivial extension of the syntax of OWL with n-ary properties. Decidability is not studied and the extension contemplates only properties: there are no other constructors to deal with predicates of arity n as in [4, 8]. On the other hand, OWL 2 provides the possibility of defining n-ary datatype predicates F, albeit in a restricted way [17]. We can use an n-ary predicate F in expressions of the form $\forall P_1 \ldots P_n.F$ or $\exists P_1 \ldots P_n.F$ where $P_1 \ldots P_n$ are binary data type predicates. The n-ary predicate F is actually a functional proposition defined implicitly by a formula of the form $\lambda(x_1 \ldots x_n).\mathsf{comp}(p, q)$ where $\mathsf{comp} \in \{\leq, =, \geq, <, >, \neq\}$ and p and q are linear polynomials on $x_1, \ldots, x_n$. However, OWL does not support the definition of n-ary predicates by listing the tuples as for object and datatype properties.

Our plans for future work include further study of the extensions of DL for n-ary relations [4, 8, 18]. In particular, we have in mind to investigate the formal mechanisms that, from a DL point of view, can support the constructions illustrated in Section 4. For instance, following the argument in Section 10.6.1 of [5], the class Altarpieces can be seen as a binary relation with two attributes painter and picturename or as a ternary relation with three attributes painter, picturename and height. This is possible in the case of a descriptive attribute because of the fact that there is a functional dependency. However, in the case of general ternary relations such as hasFigure, this is not possible: the class Altarpieces cannot be seen as a ternary relation with attributes painter, picturename and hasFigure. The relation hasFigure is being represented by the role name hasFigure and not by Altarpieces. This change of point of view is important when we consider aggregations of aggregations, which is another topic that we are exploring.

The examples presented in this paper are very simple and try to extract the main concepts behind the method. However, we have applied aggregations to more complex relations in the Ontology of Altarpieces. For instance, there are cases where we need to add another layer of aggregations and, therefore, consider aggregations of aggregations. This is the case of many relations such as *holds* and *wears* where the relation *hasFigure* itself needs to be reified.

References

1. W3C: OWL 2 Overview. http://www.w3.org/TR/owl2-overview/
2. Horrocks, I., Patel-Schneider, P.F., McGuinness, D.L., Welty, C.A.: OWL: a Description Logic Based Ontology Language for the Semantic Web. In Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P.F., eds.: The Description Logic Handbook. CUP (2007)
3. Horrocks, I., Kutz, O., Sattler, U.: The Even More Irresistible $\mathcal{SROIQ}$. In: Proc. of the 10th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR 2006), AAAI Press (2006) 57–67
4. Calvanese, D., Giacomo, G.D.: Expressive Descriptions Logics. In: The Description Logic Handbook. CUP 193–236
5. Borgida, A., Brachman, R.J.: Conceptual Modeling with Description Logics. (2003) 349–372
6. Smith, J., Smith, D.: Database abstractions: aggregation. Communications of the ACM **20**(6) (June 1977) 405–413
7. Ekserdjian, D., P.Severi: Ontology of Altarpieces. Technical report, University of Leicester (2009)
8. Giacomo, G.D., Lenzerini, M.: Description Logics with Inverse Roles, Functional Restrictions, and N-ary Relations. In: JELIA '94, Springer (1994) 332–346
9. Motik, B., Horrocks, I., Sattler, U.: Bridging the Gap Between OWL and Relational Databases. J. of Web Semantics **7**(2) (April 2009) 74–89
10. Parsia, B., Sattler, U., Schneider, T.: Easy Keys for OWL. In: OWLED. (2008)
11. Ramakrishna, R., Gehrke, J.: Database Management Systems (3rd edition). Mc Graw Hill (2003)
12. Sattler, U., Calvanese, D., Molitor, R.: Relation with Other Formalisms. In: The Description Logic Handbook. CUP 149–192
13. Borgida, A., Lenzerini, M., Rosati, R.: Description Logic for Databases. In: The Description Logic Handbook. CUP 149–192
14. Veres, C.: Aggregation in Ontologies: Practical Implementations in OWL. In: ICWE 2005. LNCS, Springer (2005)
15. Noy, N., Rector, A.: N-ary relations. http://www.w3.org/TR/swbp-n-aryRelations/
16. Salguero, A., Delgado, C., Araque, F.: Easing the Definition of N-Ary Relations for Supporting Spatio-Temporal Models in OWL. In: EUROCAST. (2009) 271–278
17. B.Parsia, Stattler, U.: OWL 2: Data Range Extension: Linear Equations. http://www.w3.org/TR/2009/NOTE-owl2-dr-linear-20091027/
18. Giacomo, G.D., Lenzerini, M.: What's in an Aggregate: Foundations for Description Logics with Tuples and Sets. In: IJCAI (1). (1995) 801–807