
Volume 9 issue 2
July 2011
ISSN 1570-8268
ELSEVIER
JOURNAL OF
Web Semantics
SCIENCE,
SERVICES &
AGENTS ON THE
WORLD WIDE WEB

Web Semantics: Science, Services and Agents on the World Wide Web 9 (2011) 83–98

Contents lists available at ScienceDirect

Web Semantics: Science, Services and Agents on the World Wide Web

journal homepage: http://www.elsevier.com/locate/websem

Guiding the representation of *n*-ary relations in ontologies through aggregation, generalisation and participation

Paula Severi [a,*], José Fiadeiro [a,*], David Ekserdjian [b]

[a] Department of Computer Science, University of Leicester, Leicester LE1 7RH, United Kingdom
[b] Department of History of Art and Film, University of Leicester, Leicester LE1 7RH, United Kingdom

ARTICLE INFO

Article history:
Received 30 June 2010
Received in revised form 26 April 2011
Accepted 27 April 2011
Available online 5 May 2011

Keywords:
Aggregation
Generalisation
n-Ary relations
Ontology engineering
Participation dependency
Reification

ABSTRACT

We put forward a methodological approach aimed at guiding ontologists in choosing which relations to reify. Our proposal is based on the notions of aggregation, generalisation and participation as used in conceptual modelling approaches for database design in order to represent situations that, normally, would require non-binary relations or complex integrity constraints. In order to justify our approach, we provide mathematical definitions of the constructs that we propose and use them to analyse the extent to which they can be implemented in languages such as OWL. A number of results are also proved that attest to the soundness of the methodological guidelines that we propose. The feedback received from using the method in a real-word situation is that it 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 [1–4]. 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.[1] In the literature on Description Logic (DL) [5], the class codifying a relation ρ is called the *reification of* ρ.[2]

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 [6] 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 often masks the concepts and structures that it encodes. 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. Ontologies that are simple and easy to read are also more likely to be reused.

In this paper we detail and expand on a method of Ontology Engineering that we introduced in [7] for simplifying ontologies. Our method is inspired by notions previously proposed in database modelling and construction for increasing the "understandibility of relational models by the imposition of additional semantic structure" [8]: aggregation and generalisation [9] and dependencies between relations [10]. 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 conceptualise 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. Our method also makes ontologies easier to extend, in particular to reuse existing reifications when adding new relations to an ontology, which is essential for supporting an incremental process of ontology engineering.

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 [11]. 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 show how aggregation as a

* Corresponding authors.
E-mail addresses: ps56@mcs.le.ac.uk (P. Severi), jose@mcs.le.ac.uk (J. Fiadeiro), dpme1@leicester.ac.uk (D. Ekserdjian).

[1] Similarly for RDF (Resource Description Framework).

[2] 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.

1570-8268/$ - see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.websem.2011.04.003

modelling abstraction and a new notion of dependency called *participation* allows us to reduce the arity of a relation.[3] We also discuss how these concepts can be used effectively in a number of situations that are recurrent in domains such as that of altarpieces to decide which relations should be reified. Finally, in Section 5, we show how further simplification can be achieved through a mechanism of generalisation.

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 minimise them, we use the Ontology of Altarpieces [13] – 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. There are a fair number of 16th-century altarpieces in Italy [11], each with a rich set of variations in the way they depict the subject matters, which is precisely what motivated this project as the typical tool of the art expert – the spreadsheet – is not powerful enough to analyse their properties.[4]

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

> *Joseph is holding the flowering staff in the altarpiece called "The Marriage of the Virgin" by Raphael*

The natural representation of this domain property is in terms of a relation *holds* of arity 3:

$$(\texttt{raphael*marriageofvirgin},\\ \texttt{joseph},\\ \texttt{floweringstaff}) \in \mathit{holds}$$

where `raphael*marriageofvirgin` is an identifier for the altarpiece "The Marriage of Virgin" painted by Raphael. In the ontology of altarpieces, we follow the traditional practice of art historians (the domain experts) and use identifiers of the form `painter *title` for altarpieces where `painter` is the name of the painter and `title` is the designation of the picture.[5]

Fig. 1 shows an entity-relationship (ER) diagram for the relationship *holds* of which the triple above is an instance. The corresponding relation cannot be represented in OWL unless we code it as a class `Reifholds` of individuals that represent the tuples – the reification of the relation [5]. For example, we can create an individual `h1` that represents the tuple:

$$(\texttt{raphael*marriageofvirgin},\\ \texttt{joseph},\\ \texttt{flowering staff})$$

This individual then needs to be connected to each component of the tuple using the role names `Altarpiece`, `Figure` and `Object` as shown in Fig. 2, which in OWL Manchester syntax can be represented as follows:

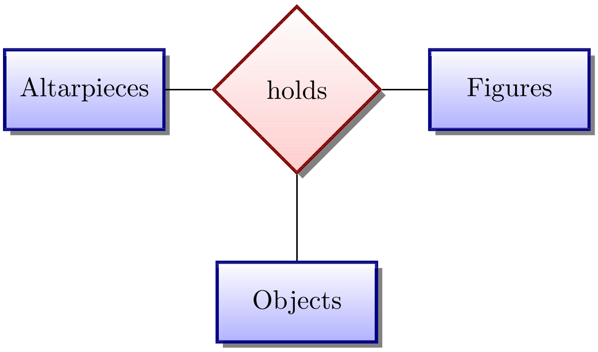

Fig. 1. ER diagram: holds as a relationship of arity 3.

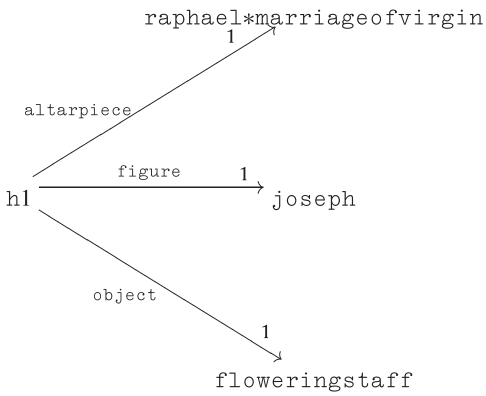

Fig. 2. Representation of a triplet through reification.

```
Class: Reifholds
ObjectProperty: altarpiece
ObjectProperty: figure
ObjectProperty: object

Individual: h1
    Types: Reifholds
    Facts: altarpiece raphael*marriageofvirgin,
           figure joseph,
           object floweringstaff
```

However, reifying *holds* is not necessarily the right decision that a modeller should make. In order to understand why, consider that subsequent additions of triples leads to the following extension of the relationship *holds*.

$$\begin{aligned}\mathit{holds} = \\ &\{(\texttt{raphael}*\texttt{marriageofvirgin}, \texttt{joseph}, \texttt{floweringstaff}),\\ &(\texttt{raphael}*\texttt{marriageofvirgin}, \texttt{joseph}, \texttt{ring}),\\ &(\texttt{corregio}*\texttt{foursaints}, \texttt{peter}, \texttt{keys}),\\ &(\texttt{corregio}*\texttt{foursaints}, \texttt{peter}, \texttt{book}),\\ &(\texttt{roselli}*\texttt{madonnaandsaints}, \texttt{catherine}, \texttt{palm}),\\ &(\texttt{roselli}*\texttt{madonnaandsaints}, \texttt{catherine}, \texttt{book}),\\ &(\texttt{dabrescia}*\texttt{madonnaandchild}, \texttt{catherine}, \texttt{sword}),\\ &(\texttt{dabrescia}*\texttt{madonnaandchild}, \texttt{catherine}, \texttt{palm})\}\end{aligned}$$

Through reification, we end up with 8 individuals, each coding one of the triples, and 24 assertions for establishing the connections depicted in Fig. 2. A simple inspection of the triples suggests that a simpler representation could be achieved by coding instead the pairs

[3] Our notion of participation differs from the notion of participation used in the ER model which relates an entity with a relation [12].

[4] One of the queries we are interested in is in finding out the names and painters of the altarpieces that satisfy certain description. For example, what are the altarpieces that have someone holding a flowering staff?

[5] If the painter is not known, we associate with it a special individual `anonymous1`, `anonymous2`, etc. If a painter had two paintings under the same title such as "Coronation of the Virgin" by Lorenzo Monaco in the National Gallery, we enumerate the titles such as `coronationofvirgin1` and `coronationofvirgin2`.

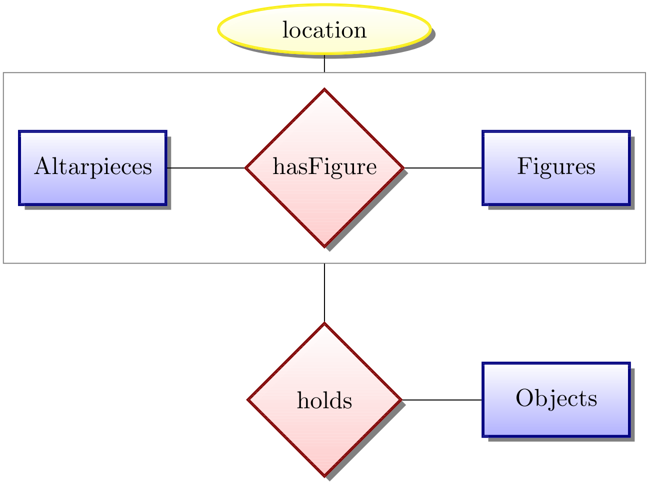

Fig. 3. ER diagram: holds reduced to a binary relation through an aggregation.

(raphael∗marriageofvirgin, joseph),
(corregio∗foursaints, peter),
(roselli ∗ madonnaandsaints, catherine),
(dabrescia ∗ madonnaandchild, catherine)

say, with four individuals:

raphael∗marriageofvirgin∗joseph
corregio∗foursaints∗peter
roselli ∗ madonnaandsaints ∗ catherine,
dabrescia ∗ madonnaandchild ∗ catherine

respectively, and then use a binary relation to represent *holds*:

$\widehat{holds} =$
{(raphael∗marriageofvirgin∗joseph, floweringstaff),
(raphael∗marriageofvirgin∗joseph, ring),
(corregio∗foursaints∗peter, keys),
(corregio∗foursaints∗peter, book),
(roselli∗madonnaandsaints∗catherine, palm),
(roselli∗madonnaandsaints∗catherine, book),
(dabrescia∗madonnaandchild∗catherine, sword),
(dabrescia∗madonnaandchild∗catherine, palm)}

The main simplification arises from the fact that we were able to transform the ternary relation *holds* into a binary relation $\widehat{holds}$, which does not need to be reified: it can be represented directly in OWL through a role.

In addition to saving on the number of individuals (4 instead of 8) and assertions (8 role assertions for the reification instead of 24),[6] we argue that this simplification can be justified in methodological terms as it brings the representation closer to the domain. Indeed, a conceptual model of the whole domain would reveal a richer semantic structure that is not captured in the simple diagram given in Fig. 1. More precisely, a wider conceptual model of the domain of altarpieces as depicted in Fig. 3 shows that the entities *Altarpieces*, *Figures* and *Objects* are involved in more complex relationships. On the one hand, *holds* is actually a binary relationship between *Objects* and the 'aggregation' of a relationship *hasFigure* between *Altarpieces* and *Figures* (the aggregation is depicted by a box surrounding the relationship subdiagram, which is the notation usually adopted in conceptual modelling approaches). On the other hand, *hasFigure* has a 'descriptive attribute' (functional relationship) that returns the location of the figure in the altarpiece – one of right, left, center, top, bottom, heaven or earth.

In summary, the simplification discussed above corresponds, effectively, to reifying *hasFigure* and representing *holds* as a binary relation as depicted in Fig. 3. Our purpose in this paper is, precisely, to investigate how far the reification of *hasFigure* can be taken to represent the aggregation of the relation as understood in conceptual modelling and how this and other constructions (such as descriptive attributes) that have been proposed by the database community 30 years ago can be used for developing simpler and reusable ontologies from conceptual models. To state the obvious, one should not take a blind approach to the representation 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. As in database design, one should build a conceptual model of the domain before starting coding in OWL or any other language, and follow a sound methodology, as outlined in this paper, to generate or reuse code.

3. Aggregation in set theory vs reification in OWL

Aggregation, as defined in [8], 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 Hoare for record structures in programming languages [14]) to be used in the context of relational models. Although a formal definition was not given therein 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.

3.1. Aggregation and reification of binary relations

In this section, we propose a formalisation for the notions of aggregation and reification of binary relations. It may seem strange that we define these notions for binary relations when the motivation for the paper is the representation of non-binary ones. The reasons for doing so are twofold. On the one hand, the case of binary relations is simpler and easier to understand. On the other hand, and more importantly, the method we propose shows that, sometimes, it is convenient to reify binary relations as a means of simplifying the representation of non-binary ones, as illustrated in the previous section. Something similar happens in [15] where, in some cases, a ternary relation is not directly reified but represented indirectly by reifying a binary relation.

We start by defining the concept of aggregation in the context of Set Theory and then that of reification in the context of OWL. We then analyse the extent to which aggregations can be implemented in OWL as reifications.

Definition 1 (*Aggregation of a binary relation*). Let $\Delta_1, \Delta_2 \subseteq \Delta$ be sets and $\rho \subseteq \Delta_1 \times \Delta_2$ be a binary relation. An aggregation of ρ over Δ is a set $\Delta_\rho \subseteq \Delta$ together with two (total) functions σ_1 and σ_2 (called *attribute functions*) from Δ_ρ to Δ_1 and Δ_2, respectively, such that the following conditions hold:

1. For all $r \in \Delta_\rho$, $\langle \sigma_1(r), \sigma_2(r) \rangle \in \rho$ – 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 $\sigma_1(r) = x_1$ and $\sigma_2(r) = x_2$ – i.e., the aggregation covers the whole relation ρ.
3. For all $r_1, r_2 \in \Delta_\rho$, if $\sigma_1(r_1) = \sigma_1(r_2)$ and $\sigma_2(r_1) = \sigma_2(r_2)$ then $r_1 = r_2$ – i.e., there is no 'confusion': every tuple of the relation has a unique representation as an aggregate (see Fig. 4).

[6] There are an additional 8 role assertions for representing the relation itself but these are 'natural' in the sense they do not arise from a reification. Still, this would mean a total of 16 assertions instead of 24.

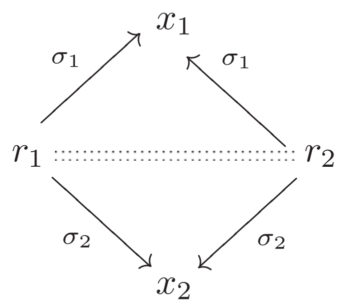

Fig. 4. Unicity of the representation.

Note that the aggregation is the set Δ_ρ together with the attribute functions σ_1 and σ_2, i.e. the whole structure $\langle \Delta_\rho, \sigma_1, \sigma_2 \rangle$.

Throughout the paper, we use the Greek alphabet for metavariables ranging on sets and relations in Set Theory. Capital letters $\Delta, \Gamma \ldots$ are used for sets and lower case letters $\rho, \sigma \ldots$ for relations.

It is easy to see that aggregations are unique up to isomorphism.

Example 1. Consider the following binary relation:

$$\begin{aligned} \mathit{hasPainted} = \{&(\texttt{raphael}, \texttt{marriageofvirgin}),\\ &(\texttt{corregio}, \texttt{foursaints}),\\ &(\texttt{roselli}, \texttt{madonnaandsaints}),\\ &(\texttt{dabrescia}, \texttt{madonnaandchild})\} \end{aligned}$$

An aggregation of *hasPainted* is the set

$$\begin{aligned} &\mathit{Altarpieces} =\\ &\{\texttt{raphael} * \texttt{marriageofvirgin},\\ &\quad \texttt{corregio} * \texttt{foursaints},\\ &\quad \texttt{roselli} * \texttt{madonnaandsaints},\\ &\quad \texttt{dabrescia} * \texttt{madonnaandchild}\} \end{aligned}$$

together with the attribute functions `painterset` and `picturenameset` defined by:

$$\begin{aligned} &\texttt{painter}(\texttt{raphael} * \texttt{marriageofvirgin}) = \texttt{raphael},\\ &\texttt{picturename}(\texttt{raphael} * \texttt{marriageofvirgin}) =\\ &\quad \texttt{marriageofvirgin},\\ &\vdots \end{aligned}$$

Note that we could have used `a1`, `a2`, `a3` and `a4` as an alternative notation for the four altarpieces shown in *Altarpieces* (or any another similar encoding), together with the corresponding attribute functions, because aggregations are unique up to isomorphism.

Example 2. Consider the following binary relation:

$$\begin{aligned} &\mathit{hasFigure} =\\ &\{(\texttt{raphael} * \texttt{marriageofvirgin}, \texttt{joseph}),\\ &(\texttt{corregio} * \texttt{foursaints}, \texttt{peter}),\\ &(\texttt{roselli} * \texttt{madonnaandsaints}, \texttt{catherine}),\\ &(\texttt{dabrescia} * \texttt{madonnaandchild}, \texttt{catherine})\} \end{aligned}$$

An aggregation of *hasFigure* is the set

$$\begin{aligned} &\mathit{FiguresinAltarpieces} =\\ &\{\texttt{raphael} * \texttt{marriageofvirgin} * \texttt{joseph},\\ &\quad \texttt{corregio} * \texttt{foursaints} * \texttt{peter},\\ &\quad \texttt{roselli} * \texttt{madonnaandsaints} * \texttt{catherine},\\ &\quad \texttt{dabrescia} * \texttt{madonnaandchild} * \texttt{catherine}\} \end{aligned}$$

together with the attribute functions *altarpiece* and *figure* defined by:

$$\begin{aligned} &\mathit{altarpiece}(\texttt{raphael} * \texttt{marriageofvirgin} * \texttt{joseph}) =\\ &\quad \texttt{raphael} * \texttt{marriageofvirgin},\\ &\mathit{figure}(\texttt{raphael} * \texttt{marriageofvirgin} * \texttt{joseph}) =\\ &\quad \texttt{joseph},\\ &\vdots \end{aligned}$$

Example 3. In order to represent polyptych altarpieces such as "The Birth of the Virgin" by Lorenzetti, we divide the set *Altarpieces* of altarpieces in two disjoint subsets, the set *OAltarpieces* of one-field altarpieces and the set *MAltarpieces* of polyptych (many-field) altarpieces that can have any number of panels (or fields). We follow the convention of [11] and we enumerate the fields clockwise from left to right, top to bottom. For example, the altarpiece by Lorenzetti has three fields: `field1`, `field2` and `field3`. We associate fields with many-field altarpieces by means of a relation $\mathit{hasField} \subseteq \mathit{MAltarpieces} \times \mathit{Fields}$ which in the example would be:

$$\begin{aligned} &\mathit{hasField} =\\ &\{(\texttt{lorenzetti} * \texttt{birthofvirgin}, \texttt{field1}),\\ &(\texttt{lorenzetti} * \texttt{birthofvirgin}, \texttt{field2}),\\ &(\texttt{lorenzetti} * \texttt{birthofvirgin}, \texttt{field3})\} \end{aligned}$$

The set *FieldsinAltarpieces* defined by

$$\begin{aligned} &\{\texttt{lorenzetti} * \texttt{birthofvirgin} * \texttt{field1},\\ &\texttt{lorenzetti} * \texttt{birthofvirgin} * \texttt{field2},\\ &\texttt{lorenzetti} * \texttt{birthofvirgin} * \texttt{field3}\} \end{aligned}$$

is an aggregation of *hasField* together with the attributes *maltarpiece* and *field* defined in the obvious manner.

The following proposition says that an aggregation of a relation is as expressive as the relation: it stores the same information, which can be retrieved by the attribute functions.

Proposition 1. *Let ρ be a binary relation. Every aggregation Δ_ρ of ρ is isomorphic to ρ in the sense that there exists a unique function $\Psi : \Delta_\rho \to \Delta_1 \times \Delta_2$ such that $\pi_i \circ \Psi = \sigma_i$ $(i = 1, 2)$, where each π_i is the* ith*-projection of the Cartesian product $\Delta_1 \times \Delta_2$.*

It is trivial to prove that Ψ is a bijection. Its inverse defines the encoding of the relation, i.e. it assigns to each pair in the relation ρ a unique element (aggregate) of the set Δ_ρ.

Informally, the reification of a relation ρ in OWL is a class C_ρ representing the tuples of ρ [5,16]. In order to be able to analyse the relationship between reification and aggregation, it is useful to provide a concrete definition of how we use the notion of reification:

Definition 2 (*Reification of a binary relation*). 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 S_1 and S_2 (called *attribute roles*), a role R, two concepts D_1 and D_2, and the following collection $\mathcal{T}_\rho$ of axioms:

(func)	$\top \sqsubseteq \leqslant 1S_1 \sqcap \leqslant 1S_2$
(domain)	$\exists S_1.\top \sqcap \exists S_2. \sqsubseteq C_\rho$
(range)	$\top \sqsubseteq \forall S_1.D_1 \sqcap \forall S_2.D_2$
(totality)	$C_\rho \sqsubseteq \exists S_1.D_1 \sqcap \exists S_2.D_2$
(contains)	$(S_1)^{-1} \circ S_2 \sqsubseteq R$
(unique rep)	C_ρ $\texttt{hasKey}(S_1, S_2)$

Please note that the `hasKey` constructor of OWL-2 [17] is used in the last axiom to ensure that any two named individuals $r$and r' in C_ρ are equal if they satisfy $S_1(r, s_1)$, $S_2(r, s_2)$, $S_1(r', s_1)$ and $S_2(r', s_2)$. The importance of this axiom is discussed after the following example.

For the composition $R \circ S$ of properties R and S, we follow the convention used in OWL where $R \circ S(x, z)$ if $R(x, y)$ and $S(y, z)$. Please, note that this is different from the standard set-theoretical way of writing composition (see, for instance, Proposition 1).

Example 4. Consider the relation *hasPainted* introduced in Example 1. Since we identify altarpieces precisely through the name of the painter and the designation of the picture, the class `Altarpieces` correspond in a natural way to the reification of *hasPainted*. In order to declare the concept `Altarpieces` to be the reification of *hasPainted* in OWL, we need a role `hasPainted`, two attributes roles *Painter* and *PictureName*, the attribute role `inversepainter` inverse of *Painter*, two concepts `Painters` and `PictureNames`, and the axioms of Definition 2. The axioms are written in a user's friendly syntax (OWL Manchester Syntax) below.

```
ObjectProperty: painter
  Characteristics: Functional
  Domain: Altarpieces
  Range: Painters
  InverseOf: inversepainter

ObjectProperty: picturename
  Characteristics: Functional
  Domain: Altarpieces
  Range: PictureNames

ObjectProperty: hasPainted
  Domain: Painters
  Range: PictureNames
  SubPropertyChain: inversepainter ∘ picturename

Class: Painters
  Class: PictureNames
  Class: Altarpieces
  SubClassOf: painter some Painters and
              picturename some PictureNames
  HasKey: (painter,picturename)
```

Fig. 5 shows a diagram illustrating all the components involved in the reification of the relation *hasPainted*. The reification itself is the concept or class `Altarpieces` and the two attribute roles `Painter` and `PictureName`.

We first discuss the importance of the axiom (contains) which is expressed above as a property chain. Using this axiom, we only have to introduce in OWL that the individual `raphael*marriageofvirgin` belongs to the class `Altarpieces` and the values for the projections:

```
Individual: raphael*marriageofvirgin
   Types: Altarpieces
   Facts: painter raphael,
          picturename marriageofvirgin
```

Then, OWL will be able to infer that `raphael` painted `marriageofvirgin`, i.e.

```
Individual: raphael
   Fact: hasPainted marriageofvirgin
```

This inference is denoted by dotted arrow in Fig. 5.

It is worth discussing in more detail the importance of including the last axiom – *unique rep*. In the following examples, we are interested in queries that combine the reasoner with the closed world assumption. For this, we are assuming the notion of certain answer as in [18,19]. Intuitively, the certain answers to a query are the answers which are known so far and are compatible with the ontology. Formally, $\{(c_1, c_2) | \mathcal{O} \models q(c_1, c_2)\}$ where $q(x, y)$ is a formula with only 2 variables and $\mathcal{O}$ is an ontology. If we do not ensure uniqueness of representation, a modeller (or different people working over the same ontology) can perfectly well introduce two individuals representing the same tuple without the reasoner being able to infer that they are equal. In simple terms, this means that, in the absence of the axiom, a query to retrieve the original table may have repeated rows. This implies, in particular, that queries may return incorrect answers. For instance, if we want to query the number of altarpieces that have no signature, we may count the same tuple more than once. Adding the axiom *unique rep* allows us to do a more correct way of closed world reasoning.

For another example, suppose that we want to express two relationships – `paintmedia` and `height` – on the class `Altarpieces`. As depicted in Fig. 6, it is possible that two representations

```
raphael * marriageofvirgin
raphaelmarriageofvirgin
```

of the same tuple of *hasPainted* have been accidentally introduced. The individual `raphael*marriageofvirgin` got connected by the predicate `paintmedia` to `oil` – representing the fact that, Raphael's Marriage of Virgin has been painted in oil – and `raphaelmarriageofvirgin` by the predicate `height` to 170 – representing the fact that, the same painting is 170 cm high.

A query can go awfully wrong if it does not take into account the fact that the tuples can be duplicated. To clarify this point, consider the query "Is there an altarpiece painted in oil and 170 cm high?". The intuitive formulation of the query

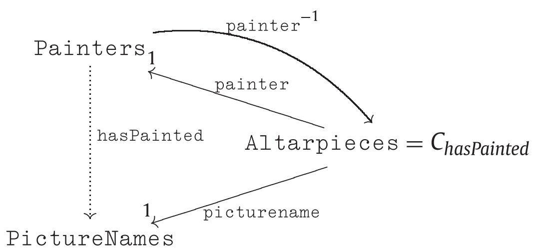

Fig. 5. `Altarpieces` as the reification of *hasPainted*.

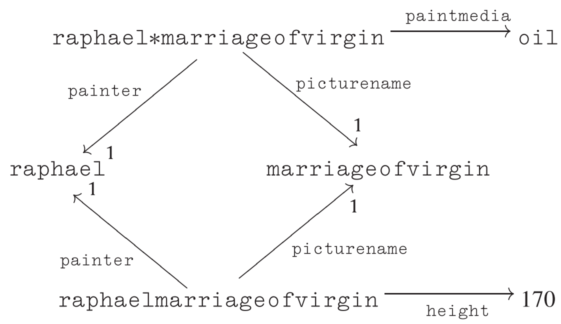

Fig. 6. Two properties connected to two different representations of the same tuple.

$$
\begin{aligned}
\mathtt{q1}(x_1, x_2) \leftarrow\ & \mathtt{paintmedia}(r, \mathtt{oil}) \sqcap \\
& \mathtt{height}(r, 170) \sqcap \\
& \mathtt{Painter}(r, x_1) \sqcap \\
& \mathtt{PictureName}(r, x_2)
\end{aligned}
$$

will incorrectly return no answers because of the duplicated representation. The following alternative formulation would take into account that tuples can be duplicated:

$$
\begin{aligned}
\mathtt{q2}(x_1, x_2) \leftarrow\ & \mathtt{paintmedia}(r_1, \mathtt{oil}) \sqcap \\
& \mathtt{height}(r_2, 170) \sqcap \\
& \mathtt{Painter}(r_1, x_1) \sqcap \\
& \mathtt{PictureName}(r_1, x_2) \sqcap \\
& \mathtt{Painter}(r_2, x_1) \sqcap \\
& \mathtt{PictureName}(r_2, x_2)
\end{aligned}
$$

This query will correctly return one answer which is $x_1 = \mathtt{raphael}$ and $x_2 = \mathtt{marriageofvirgin}$.

However, this formulation is no longer very intuitive, i.e. it needs to anticipate the existence of multiple representations, which is a problem of the representation that does not arise from the domain of discourse. In summary, the inclusion of *unique rep* is essential to ensure that we have a faithful representation of the domain.

Example 5. Consider the relation *hasFigure* defined in Example 2. The corresponding representation in OWL is achieved through the reification of the relation *hasFigure*. We introduce the concept `FiguresinAltarpieces` to represent this reification and add all the corresponding axioms (see Definition 2). These axioms are written in OWL Manchester syntax as follows:

```
ObjectProperty: altarpiece
  Characteristics: Functional
  Domain: FiguresinAltarpieces
  Range: Altarpieces
  InverseOf: inversealtarpieces

ObjectProperty: figure
  Characteristics: Functional
  Domain: FiguresinAltarpieces
  Range: Figures

ObjectProperty: hasFigure
  Domain: Altarpieces
  Range: Figures
  SubPropertyChain: inversealtarpieces ∘ figure

Class: Figures
Class: FiguresinAltarpieces
  SubClassOf: altarpieces some Altarpieces and
              figure some Figures
  HasKey: (altarpiece, figure)
```

Fig. 7 shows a diagram illustrating all the components involved in the reification of *hasFigure*.

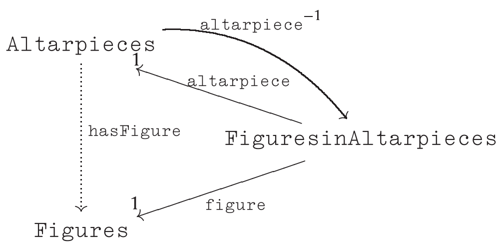

Fig. 7. `FiguresinAltarpieces`: the reification of *hasFigure*.

We follow the convention mentioned in Example 2 and the tuple (`raphael*marriageofvirgin`, `joseph`) is represented by the individual

`raphael * marriageofvirgin * joseph`

We also have to associate values with the attribute roles `altarpiece` and `figure`:

```
Individual: raphael*marriageofvirgin*joseph
    Types: FiguresinAltarpieces
    Facts: altarpiece raphael*marriageofvirgin,
           figure joseph
```

Example 6. Consider the relation *hasField* introduced in Example 3. For the reification of the relation *hasField*, we introduce the concept `FieldsinAltarpieces` and add all the corresponding axioms (see Definition 2). These axioms are written in OWL Manchester syntax as follows:

```
ObjectProperty: maltarpiece
  Characteristics: Functional
  Domain: FieldsinAltarpieces
  Range: MFieldAltarpieces
  InverseOf: inversemaltarpieces

ObjectProperty: field
  Characteristics: Functional
  Domain: FieldsinAltarpieces
  Range: Fields

ObjectProperty: hasField
  Domain: MFieldAltarpieces
  Range: Fields
  SubPropertyChain: inversemaltarpieces ∘ field

Class: Fields
Class: FieldsinAltarpieces
  SubClassOf: maltarpiece
              some MAltarpieces
              and field some Fields
  HasKey: (maltarpiece, field)
```

Fig. 8 shows a diagram illustrating all the components involved in the reification of *hasField*.

We follow the same convention mentioned before and the tuple (`lorenzetti*birthofvirgin`, `field2`) is represented by the individual `lorenzetti*birthofvirgin*field2`. We also have to associate values with the attribute roles `maltarpiece` and `Field`:

```
Individual: lorenzetti*birthofvirgin*field2
    Types: FieldsinAltarpieces
    Facts: maltarpiece lorezetti*birthofvirgin,
           field field2
```

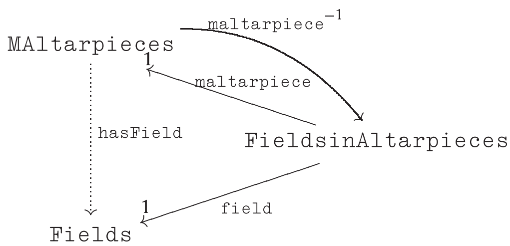

Fig. 8. `FieldsinAltarpieces`: the reification of *hasField*.

We can now define more precisely how a reification relates to the relation. Throughout the remainder of the paper, we use capital letters $C, D \ldots$ for metavariables ranging over concepts or classes and we use $R, S, \ldots$ for roles or properties.

Definition 3 (*Faithful reification*). Let $\rho \subseteq \Delta_1 \times \Delta_2$ be a binary relation. Given an interpretation I, we say that the reification $\langle C_\rho, R, D_1, D_2, S_1, S_2 \rangle$ is *faithful* to ρ in relation to I iff $R^I = \rho$, $D_1^I = \Delta_1$, $D_2^I = \Delta_2$, and $\langle C_\rho^I, S_1^I, S_2^I \rangle$ is an aggregation of ρ.

Unfortunately, the axioms that are part of the reification (Definition 2) are not sufficient to guarantee that the reification is faithful to ρ in relation to every interpretation:

- The first three axioms state that the role names S_1 and S_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 S_1 or S_2 for some element of C_ρ (see [20]).
- The converse of *R-contains*, which would correspond to the second condition of Definition 1, is as follows

  (*R-inclusion*) $R \sqsubseteq (S_1)^{-1} \circ S_2$

  However, this axiom cannot be expressed in OWL in that way because the right-hand side of the inclusion is not a role name (see [3]).
- The axiom *unique rep* is weaker than the third condition of Definition 1 in the sense that the 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 (the so-called "EasyKey constraints") that is valid only for individuals belonging to the Herbrand Universe [17].

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.

3.2. Aggregation and reification of n-ary relations

Definition 1 can be generalised to relations of arbitrary arity and to relations with 'key attributes' as originally introduced in [8]:

Definition 4 (*Aggregation of a relation of arity n*). Let $\Delta_1, \ldots, \Delta_n \subseteq \Delta$ and $\rho \subseteq \Delta_1 \times \cdots \times \Delta_n$ be a relation. Let $i_1, \ldots, i_k \in \{1, \ldots, n\}$. An aggregation of ρ with keys $\{i_1, \ldots, i_k\}$ over a universe Δ is a set $\Delta_\rho \subseteq \Delta$ together with n (total) functions $\sigma_1, \ldots, \sigma_n$ from Δ_ρ to $\Delta_1, \ldots, \Delta_n$, respectively, such that:

1. For all $r \in \Delta_\rho$, we have that $\langle \sigma_1(r), \ldots, \sigma_n(r) \rangle \in \rho$.
2. For all $\langle x_1, \ldots, x_n \rangle \in \rho$, there exists $r \in \Delta_\rho$ such that $\sigma_1(r) = x_1, \ldots, \sigma_n(r) = x_n$.
3. For all $r_1, r_2 \in \Delta_\rho$, if $\sigma_{i_1}(r_1) = \sigma_{i_1}(r_2), \ldots, \sigma_{i_k}(r_1) = \sigma_{i_k}(r_2)$ then $r_1 = r_2$ – i.e., every tuple of the relation is uniquely identified by its key attributes.

The functions $\sigma_{i_1}, \ldots, \sigma_{i_k}$ are called *key attributes functions* and the remaining ones are called *non-key attribute functions*.

Proposition 1 has a trivial generalisation to the n-ary case:

Proposition 2. *Let $\rho \subseteq \Delta_1 \times \cdots \times \Delta_n$ be a relation and $i_1, \ldots, i_k \in \{1, \ldots, n\}$. Then, every aggregation Δ_ρ of ρ with keys $\{i_1, \ldots, i_k\}$ is isomorphic to ρ in the sense that there exists a unique function $\Psi : \Delta_\rho \to \Delta_1 \times \ldots \times \Delta_n$ such that $\pi_i \circ \Psi = \sigma_i (i = 1, \ldots, n)$.*

The existence of an aggregation with given keys cannot always be guaranteed:

Proposition 3. *An aggregation exists for a relation $\rho \subseteq \Delta_1 \times \cdots \times \Delta_n$ with keys $\{i_1, \ldots, i_k\}$ iff*

- *ρ is a partial function from $\Delta_{i_1} \times \ldots \times \Delta_{i_k}$ into $\Delta_{j_1} \times \cdots \Delta_{j_{n-k}}$, where the set $\{j_1, \ldots, j_{n-k}\}$ is the complement of $\{i_1, \ldots i_k\}$;*
- *there is an embedding (injective function) of the domain of ρ in Δ, i.e. the universe is large enough to represent the relation.*

Note that, in the conditions of this proposition, if the universe $i\Delta$ contains the Cartesian product $\Delta_1 \times \cdots \times \Delta_n$, then ρ is an aggregation of itself where the attribute functions are the Cartesian projections π_i. The reason we define the concept of aggregation is that the Cartesian product is not a construct of OWL (or, indeed, DL) and, therefore, one needs to resort to mechanisms like reification to encode relations.

Example 7. We consider the example of *holds* given in Section 2. An aggregation of *holds* is the set

$\Delta_{holds} = \{\mathtt{h1}, \mathtt{h2}, \mathtt{h3}, \mathtt{h4}, \mathtt{h5}, \mathtt{h6}, \mathtt{h7}, \mathtt{h8}\}$

together with the attribute functions *altarpiece*, *figure* and *object* defined by:

$$
\begin{aligned}
&\mathit{altarpiece}(\mathtt{h1}) = \mathtt{raphael} * \mathtt{marriageofvirgin},\\
&\mathit{figure}(\mathtt{h1}) = \mathtt{joseph},\\
&\mathit{object}(\mathtt{h1}) = \mathtt{floweringstaff}\\
&\vdots
\end{aligned}
$$

Example 8. As an example of a relation where only a subset of the attributes are key is *isLocated* the key attributes being *altarpiece* and *figure*. Consider the following definition of *isLocated*:

$$
\begin{aligned}
&\mathit{isLocated} =\\
&\{(\mathtt{raphael} * \mathtt{marriageofvirgin}, \mathtt{joseph}, \mathtt{right}),\\
&(\mathtt{corregio} * \mathtt{foursaints}, \mathtt{peter}, \mathtt{left}),\\
&(\mathtt{roselli} * \mathtt{madonnaandsaints}, \mathtt{catherine}, \mathtt{left}),\\
&(\mathtt{dabrescia} * \mathtt{madonnaandchild}, \mathtt{catherine}, \mathtt{right})\}
\end{aligned}
$$

Note that the third component is functional on the first two ones. The set *FiguresinAltarpieces* defined in Example 2 is an aggregation of *isLocated*. The non-key attribute $\rightarrow$ *locationset* is defined in the obvious manner.

We now generalise Definition 2 to relations of arbitrary arity. The main difference is that, for non-binary relations, we cannot use an atomic role R for representing the relation. For this reason, the axiom *R-contains* has to be dropped.

Definition 5 (*Reification of a relation of arity n*). Let $\Delta_1, \ldots, \Delta_n \subseteq \Delta$ and $\rho \subseteq \Delta_1 \times \cdots \times \Delta_n$ be a relation. Let $i_1, \ldots, i_k \in \{1, \ldots, n\}$. A *reification of ρ in OWL with keys* $\{i_1, \ldots, i_k\}$ is a concept C_ρ together with roles $S_1, \ldots, S_n$ (called *attribute roles*), domains $D_1, \ldots, D_n$, and the following collection $\mathscr{T}_\rho$ of axioms:

(func)	$\top \sqsubseteq \leqslant 1 S_1 \sqcap \ldots \sqcap \leqslant 1 S_n$
(domain)	$\exists S_1.\top \sqcap \ldots \sqcap \exists S_n.\top \sqsubseteq C_\rho$
(range)	$\top \sqsubseteq \forall S_1.D_1 \sqcap \ldots \sqcap \forall S_n.D_n$
(totality)	$C_R \sqsubseteq \exists S_1.D_1 \sqcap \ldots \sqcap \exists S_n.D_n$
(unique rep)	C_ρ `hasKey`$(S_{i_1}, \ldots, S_{i_k})$

We call $S_{i_1}, \ldots, S_{i_k}$ the *key attribute roles* of the reification and the rest are the *non-key attribute roles*.

Example 9. For the reification of *holds*, we introduce the concept `Reifholds`, three roles *Altarpiece*, *Figure* and *Object* and the axioms of Definition 5 (see also Fig. 9). These axioms are written in OWL Manchester Syntax as follows:

```
ObjectProperty: altarpiece  Characteristics:
  Functional
  Domain: Reifholds
  Range: Altarpieces

ObjectProperty: figure
  Characteristics: Functional
  Domain: Reifholds
  Range: Figures

ObjectProperty: object
  Characteristics: Functional
  Domain: Reifholds
  Range: Objects

Class: Objects
Class: Reifholds
  SubClassOf: altarpiece some Altarpieces and
              figure some Figures and
              object some Objects
  HasKey: (altarpiece, figure, object)
```

Example 10. For the reification of the relation *isLocated* from Example 8, we introduce the concept `ReifisLocated`, the key attribute roles `altarpiece`, `Figure`, and the non-key attribute role `Location` (see also Fig. 10). The axioms of Definition 5 are written in OWL Manchester Syntax as follows:

```
ObjectProperty: altarpiece
  Characteristics: Functional
  Domain: ReifisLocated
  Range: Altarpieces

ObjectProperty: figure
  Characteristics: Functional
  Domain: ReifisLocated
  Range: Figures

DataProperty: location
  Characteristics: Functional
  Domain: ReifisLocated
  Range: Locations
```

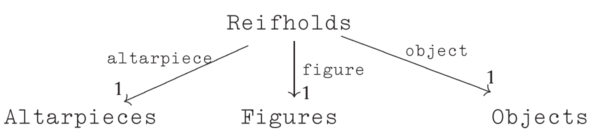

Fig. 9. Representation of *holds* as a relation of arity 3.

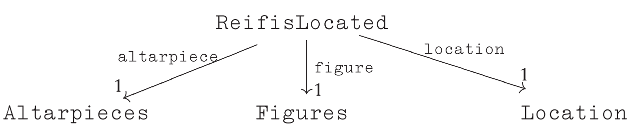

Fig. 10. Reification of *isLocated* as a relation of arity 3.

```
Class: Locations
Class: ReifisLocated
  SubClassOf: altarpiece some Altarpieces and
              figure some Figures and
              location some Locations
  HasKey: (altarpiece,figure)
```

Notice that, in this example, we used the same role names `altarpiece` and `Figure` as in Examples 5 and 9. Strictly speaking, this is an abuse of notation and we should use different role names if the concepts $C_{hasFigure}$, C_{holds} and $C_{isLocated}$ are all different. We will discuss this example again in the next section.

4. Participation dependency

In this section, we put forward a methodological approach aimed at guiding the modeller in the use of reification based on the concepts formalised in the previous section. 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 [12].

The notion of aggregation allows us to reduce the arity of a relation. This reduction can be performed without losing information if the relations satisfy certain dependencies. The notion of inclusion dependency, which is typical in databases [10], is too weak to ensure that arity reduction preserves information. Because of this, we introduce a new notion of dependency called *participation*. We illustrate the approach with some examples that are representative of the situations that we have encountered in the altarpieces project.

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 [12]. Using ER diagrams, the method can be explained in terms of evolving situations such as the one depicted in Fig. 1 to the one depicted in Fig. 3. More specifically, the method consists in identifying a binary relationship – *hasFigure* in the case at hand – such that the ternary relationship – *holds* – can be expressed as a binary relationship between the aggregation of the former and the remaining domain – *Objects* (see Fig. 3).

Following this methodological principle, instead of reifying *holds*, we reify *hasFigure*. Because *hasFigure* is a binary relation, we represent it by a role `hasFigure` and consider the reification of *hasFigure* as in Example 5, which we name `FiguresinAltarpieces`. The relation *holds* is represented as an object property whose domain is `FiguresinAltarpieces` and whose range is `Objects`. This can be expressed in OWL Manchester syntax as follows:

```
ObjectProperty: holds
    Domain: FiguresinAltarpieces
    Range: Objects
```

The result is depicted through the following diagram:

$$\texttt{FiguresinAltarpieces} \xrightarrow{\texttt{holds}} \texttt{Objects}$$

At the level of individuals, we add assertions such as:

```
Individual: raphael*marriageofvirgin*joseph
    Facts: holds floweringstaff
```

The methodological question is, then: What is the property that allows us to reduce the arity of a relation? The answer that we provide to this question is based on the key concept of 'participation'. Intuitively, a relation ρ' participates in another relation ρ if the projection of ρ on some of its components is included in ρ'.

Definition 6 (*Participation in a relation of arity 3*). Let $\Delta_1, \Delta_2, \Delta_3 \subseteq \Delta, \rho' \subseteq \Delta_1 \times \Delta_2$ be a binary relation and $\rho \subseteq \Delta_1 \times \Delta_2 \times \Delta_3$ a ternary relation. We say that ρ' *participates in* ρ if the following condition (called *participation constraint*) is satisfied:

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

Similarly, we can define when $\rho' \subseteq \Delta_2 \times \Delta_3$ or $\subseteq \Delta_1 \times \Delta_3$ participates in ρ. The relation ρ' is called the *participating relation*. We also say that there is a *participation dependency between* ρ' *and* ρ when ρ' participates in ρ.

Notice that our notion of participation differs from the one used in the ER model [12], which refers to the participation of an entity set in a relation, not of a relation in another relation. Furthermore, in the ER model, the participation constraint usually refers to the 'total participation' of an entity Δ_1 in a relation ρ , i.e. for all $x \in \Delta_1$ there exist y, z such that $(x,y,z) \in \rho$.

Example 11. The relation *hasFigure* participates in the relation *holds* because the participation constraint holds:

$$\text{if } (x,y,z) \in \textit{holds} \text{ then } (x,y) \in \textit{hasFigure for all } x,y,z. \tag{1}$$

From the point of view of the database relational model, the notion of participation is a restricted form of inclusion dependency between relations [10]. More precisely, if ρ' participates in ρ then there is an inclusion dependency between ρ' and ρ. However, the converse is not true: not all inclusion dependencies are participation dependencies. The stronger concept of participation dependency is needed to ensure that the arity of a relation can be reduced without losing information as we show in the next proposition.

Proposition 4. *Let ρ' participate in ρ and let $\Delta_{\rho'}$ be an aggregation of ρ' with attributes σ_1 and σ_2. Then, the ternary relation ρ is isomorphic to a binary relation between the aggregation $\Delta_{\rho'}$ and Δ_3, which we call the reduction of ρ by $\Delta_{\rho'}$.*

Proof. The *reduction* of ρ by $\Delta_{\rho'}$ is the relation $\widehat{\rho} \subseteq \Delta_{\rho'} \times \Delta_3$ defined by: $\widehat{\rho} = \{(r,z) | (\sigma_1(r), \sigma_2(r), z) \in \rho\}$

It follows from the fact that ρ' participates in ρ and Proposition 1 that ρ and $\widehat{\rho}$ are isomorphic. □

Definition 7 (*Arity reduction of a relation*). We say that the ternary relation ρ reduces to a binary relation σ if $\sigma = \widehat{\rho}$.

We now show that it is always possible to reduce the arity of a ternary relation.

Proposition 5. *For any ternary relation ρ there exist a binary relation σ such that ρ reduces to σ.*

Proof. Suppose $\rho \subseteq \Delta_1 \times \Delta_2 \times \Delta_3$. Trivially, $\rho' = \Delta_1 \times \Delta_2$ participates in ρ and by Proposition 4 we have that ρ reduces to $\sigma = \widehat{\rho}$ where $\widehat{\rho}$ is the reduction of ρ by $\Delta_{\rho'}$. □

There are more ways of reducing the arity of a relation than the one shown in the proof above. Since $\Delta_2 \times \Delta_3$ and $\Delta_1 \times \Delta_3$ also participate in ρ, we could have aggregated any of these two relations instead of $\Delta_1 \times \Delta_2$. We will show examples where there are more participating relations than those three.

As discussed in Section 3, aggregation can be (partially) implemented in OWL through the mechanism of reification. Taking this forward to participation, if ρ' participates in ρ, we can reify ρ' and represent the reduction $\widehat{\rho}$ as a role R whose domain is $C_{\rho'}$

$$C_{\rho'} \xrightarrow{R} \Delta_3$$

For example, in order to represent the relation *holds* in OWL, we represent its reduction $\widehat{holds}$ using the reification of *hasFigure*.

In summary, 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 $C_{\rho'}$. If ρ' participates in yet another relation, say ρ'', that relation does not need to be reified either and we can reuse instead the reification $C_{\rho'}$ of ρ'.

For example, *hasFigure* participates in many relations other than *holds* – e.g. *wears*. All the corresponding relations can be represented in OWL as object properties whose domain is `FiguresinAltarpieces`. That is, *wears* can be represented as a role `wears` whose domain is `FiguresinAltarpieces` and whose range is `Objects`, which can be written in OWL Manchester syntax as follows:

```
ObjectProperty: wears
   Domain: FiguresinAltarpieces
   Range: Objects
```

The corresponding diagrammatic representation is:

$$\texttt{FiguresinAltarpieces} \xrightarrow{\texttt{wears}} \texttt{Objects}$$

The advantage of reusing reifications is clear. The axioms for expressing that `FiguresinAltarpieces` is the reification of `hasFigure` (see Example 5) have to be introduced only once, thus avoiding the replication that a blind approach to representation would entail. By choosing to reify the participating relation, we do not only reduce the number of reifications but we also reduce the number of codifications for individuals. This is because we are coding only the components that are shared by several tuples as we showed in Section 2. These components are not only shared in one relation but also amongst several relations. For example, in order to express that Catherine is wearing a crown in "The Madonna and Saints" by Roselli, we can reuse the individual `roselli * madonnaandsaints * catherine` which was already introduced to express that Catherine is holding the palm and a book.

```
Individual: roselli*madonnaandsaints*catherine
   Facts: wears crown
```

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 *holds* represented as a property `holds` and not as a class `Reifholds` as in Example 9. Reifications represent properties but they cannot be used in the syntax as properties because they are actually classes. For instance, we cannot use constructors for roles (e.g. composition, quantification or transitive closure) on C_{holds}, which may restrict the ability of the modeller to capture important aspects of the domain. Instead, the representation of *holds* as a property allows us to use the role name `holds` in quantifications or in compositions. Below, we will see an example where `holds` is used in a composition.

Yet, one of the most important aspect of our method is that, by reifying the participating relations, we are reflecting and enforcing the participation constraint within the logic. Fig. 11 illustrates the case for the relation *holds*. The participation constraint shown in Example 11 can be deduced. In particular, OWL will be able to infer the following assertion:

```
Individual: raphael*marriageofvirgin
   Facts: hasFigure joseph
```

using the domain of `holds`, the attribute roles and the axiom (contains) of Example 5.

However, not all relevant participation constraints result in reification and have to be explicitly stated. For example, there is another participation constraint for *holds* that has not been enforced yet:

$$\text{if } (x,y,z) \in \textit{holds} \text{ then } (x,z) \in \textit{hasObject for all } x,y,z. \quad (2)$$

An easy way to enforce this constraint without changing the representation in Fig. 11 is to add the following axiom:

$\mathtt{altarpiece}^{-1} \circ \mathtt{holds} \sqsubseteq \mathtt{hasObject}$

which in OWL Manchester syntax is:

```
ObjectProperty: hasObject
  Domain: Altarpieces
  Range: Objects
  SubPropertyChain: inversealtarpiece o holds
```

Fig. 12 illustrates the case of the two overlapping constraints.

There are other ways of representing the relation *holds* and reflect the overlapping constraints. We prefer the solution already presented because it looks more readable and intuitive.

1. We could have reified *hasObject* and represented the relation *holds* as a role `holds` whose domain is `Figures` and range is the reificaton of *hasObject*. We can see that the pairs

   $(\mathtt{raphael} * \mathtt{marriageofvirgin} * \mathtt{joseph}, \mathtt{book})$

   and

   $(\mathtt{joseph}, \mathtt{raphael} * \mathtt{marriageofvirgin} * \mathtt{book})$

   codify the same triplet, i.e.

   $(\mathtt{raphael} * \mathtt{marriageofvirgin}, \mathtt{joseph}, \mathtt{book})$.

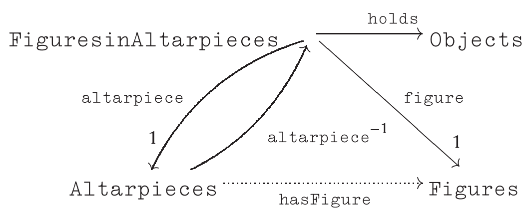

Fig. 11. Capturing the participation constraint between *hasFigure* and *holds*.

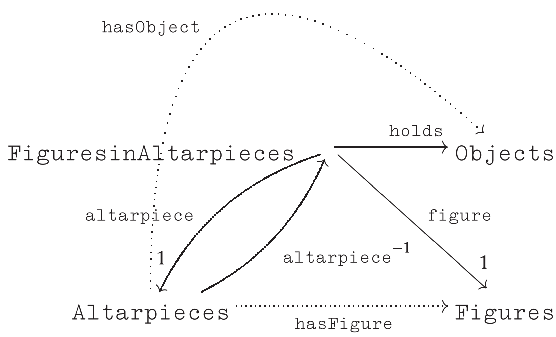

Fig. 12. Capturing the overlapping constraints: *hasFigure* and *hasObject* participate in *holds*.

   There is no mathematical difference between these solutions since both reductions of *holds* are isomorphic by Proposition 4. The diagram of the OWL representation obtained by reifying *hasObject* will look symmetric to Fig. 12.

2. We could also have reified both participating relations *hasFigure* and *hasObject*. This solution has some intuition behind it. The individual `joseph` is representing the figure of Joseph in an abstract way but the individual

   $\mathtt{raphael} * \mathtt{marriageofvirgin} * \mathtt{joseph}$

   is representing the particular figure of Joseph on the altarpiece $\mathtt{raphael} * \mathtt{marriageofvirgin}$. Hence, we could consider the pair

   $(\mathtt{raphael} * \mathtt{marriageofvirgin} * \mathtt{joseph},$
   $\mathtt{raphael} * \mathtt{marriageofvirgin} * \mathtt{book})$

   to represent the triplet. This solution looks as if it is storing redundant information since the altarpiece is coded twice. However, this solution is also mathematically equivalent to the first one and it is possible to use the axioms of the logic to state that both components of the above pair have the same altarpiece.

Example 12. We show an interesting example that involves polyptych altarpieces. Suppose that we want to express the following fact about a triptych altarpiece:

> *St Anne is a figure that appears in the central panel of the altarpiece called "The Birth of the Virgin" by Lorenzetti*

The natural representation of this domain property is in terms of a relation *hasFigureinField* of arity 3

$\textit{hasFigureinField} \subseteq \textit{MAltarpieces} \times \textit{Fields} \times \textit{Figures}$

where

$\textit{Altarpieces} = \textit{OAltarpieces} \cup \textit{MAltarpieces}$

The central panel is identified by `field2`. Then,

$$(\mathtt{lorenzetti} * \mathtt{birthofvirgin}, \mathtt{field2}, \mathtt{anne}) \in \textit{hasFigureinField}$$

We have two participation dependencies:

1. *hasField* participates in *hasFigureinField*, i.e.

   if $(x,y,z) \in \textit{hasFigureinField}$ then $(x,y) \in \textit{hasField}$

for all $x \in \textit{MAltarpieces}$, $y \in \textit{Fields}$ and $z \in \textit{Figures}$.

2. *hasFigure* participates in *hasFigureinField*, i.e.

   if $(x,y,z) \in \textit{hasFigureinField}$ then
   $(x,z) \in \textit{hasFigure}$

   for all $x \in \textit{MAltarpieces}$, $y \in \textit{Fields}$ and $z \in \textit{Figures}$.

The natural and simplest choice for reifying is the first participation relation. Fig. 13 shows the ER diagram obtained by aggregating the first participation relation. Following the same pattern as in the example of *holds*, we represent the ternary relation *hasFigureinField* as an object property whose domain is `FieldsinAltarpieces` and range is `Figures` (see Fig. 14).

```
ObjectProperty: hasFigureinField
   Domain: FieldsinAltarpieces
   Range: Figures
```

The first participation constraint of *hasField* is enforced because the domain of `hasFigureinField` is the reification of *hasField*. To

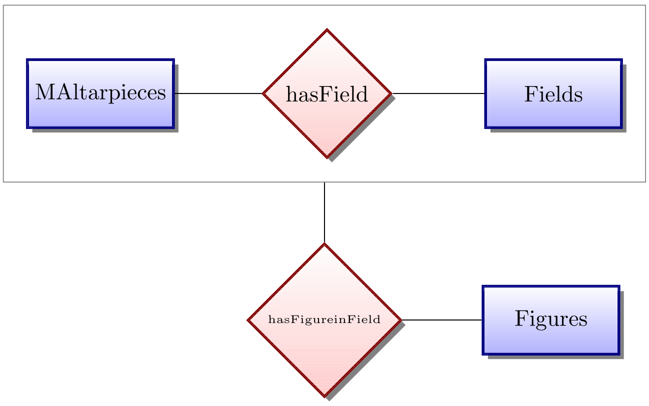

Fig. 13. ER diagram: *hasFigureinField* for Many Field Altarpieces reduced to a binary relation through an aggregation.

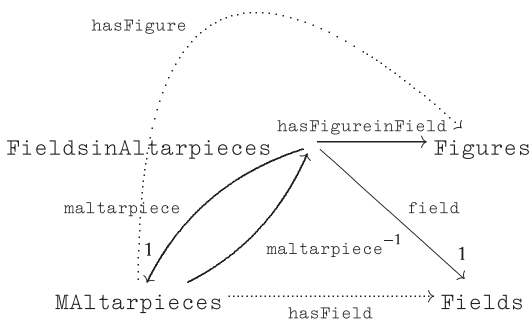

Fig. 14. Capturing overlapping constraints: *hasField* and *hasFigure* participate in *hasFigure*.

enforce the second participation constraint, we follow the same pattern as we did for the overlapping constraints of *holds* and we add the following axiom:

$$\mathtt{maltarpiece}^{-1} \circ \mathtt{hasFigureinField} \sqsubseteq \mathtt{hasFigure}$$

which in OWL Manchester syntax is:

```
ObjectProperty: hasFigure
  SubPropertyChain: inversemaltarpiece ∘
                    hasFigureinField
```

4.2. Descriptive attributes

Another related methodological guideline for the use of reification arises from what in [12] 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 in a ternary relation as an attribute of the aggregation of a binary relation. For example, it would be intuitive to represent *location* in Fig. 15 as a descriptive attribute associated with the relationship *hasFigure*.

Definition 8 (*Descriptive attribute*). Let $\rho \subseteq \Delta_1 \times \Delta_2$ and $\rho' \subseteq \Delta_1 \times \Delta_2 \times \Delta_3$. We say that ρ' is a descriptive attribute of ρ if the following conditions hold:

1. ρ' is a function from $\Delta_1 \times \Delta_2$ to Δ_3.
2. ρ participates in ρ'.

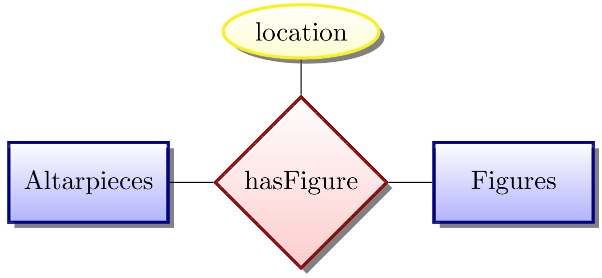

Fig. 15. ER diagram: a descriptive attribute.

It is evident from this definition that the notion of descriptive attribute is a particular case of the notion of participation as introduced in Definition 6. 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 example, *location* is a descriptive attribute of *hasFigure* because the following properties hold in the domain:

1. There is a functional dependency between the location and pair given by the altarpiece and the figure. In other words, the ternary relation *isLocated* is actually a function

   $$\mathit{isLocated} \in \mathit{Altarpieces} \times \mathit{Figures} \to \mathit{Locations}$$

2. There exists a participation dependency between the relations *hasFigure* and *isLocated*. In other words, *hasFigure* participates in *isLocated*, i.e. for all x, y and z, we have that:

   $$\mathit{if}\,(x,y,z) \in \mathit{isLocatedthen}\,(x,y) \in \mathit{hasFigure}.$$

We can represent the descriptive attribute *location* in OWL as follows:

1. We reuse the class `FiguresinAltarpieces` as the reification of *hasFigure* (see Example 5).
2. We define a role `isLocated` representing the descriptive attribute as a function whose domain is

   `FiguresinAltarpieces`.

This is written in OWL Manchester syntax as follows:

```
DateProperty: isLocated
    Characteristics: Functional
    Domain: FiguresinAltarpieces
    Range: Locations
```

The OWL-representation of the descriptive attribute *isLocated* can be depicted as follows:

$$\mathtt{FiguresinAltarpieces} \xrightarrow{\mathtt{isLocated}} \mathtt{Locations}$$

Notice that descriptive attributes are a particular case of non-key attributes (see Definition 4) where we have the additional involvement of a participating relation. At first sight it might look as if reifying *isLocated* as a relation of arity 3 whose key attributes are the first and second components (Example 10) would be the same as first reifying *hasFigure* and later adding `isLocated` as a descriptive attribute. However, there are some important differences between these two processes. Example 10 does not take into account that *hasFigure* is a participating relation. From the point of view of the axioms, this is reflected in the fact that the axiom (contains) was not present in Example 10, whilst this axiom is an important component of the reification of *hasFigure*.

There is also a methodological difference between the two processes. On the one hand, we want to leave the system open to the addition of any number of descriptive attributes (such as *size*). On the other hand, we also want to add the right amount of axioms and have a systematic way for doing so. In order to achieve this, it should be clear which is the participating relation and how the relations interact with each other. That is, descriptive attributes play an important methodological role in the representation of the domain knowledge.

4.3. Relations of arbitrary arity

In this section, we generalise the notion of participation to relations of arbitrary arity and show how the notion of participation can be used to reduce any n-ary relation to a relation of a smaller arity. We also show an example of a relation of arity 4 with several participating relations and how the choice of a participating relation can affect our representation of the relation.

Definition 9 (*Participation in a relation of arity n*). Let $i_1, \ldots, i_k \in \{1, \ldots, n\}$ be all different, $\rho \subseteq \Delta_1 \times \ldots \times \Delta_n$ and $\rho' \subseteq \Delta_{i_1} \times \ldots \times \Delta_{i_k}$. We say that ρ' *participates in* ρ iff the following constraint (also called the *participation constraint*) is satisfied for all $x_1 \in \Delta_1, \ldots, x_n \in \Delta_n$:

$$(x_1, \ldots, x_n) \in \rho \text{ implies } (x_{i_1}, \ldots, x_{i_k}) \in \rho'$$

The relation ρ' is called the *participating relation*. We also say that there is a *participation dependency between* ρ' *and* ρ when ρ' participates in ρ.

The notion of participation allows us to reduce the arity of a relation:

Proposition 6. *If* ρ' *participates in* ρ *then* ρ *is isomorphic to a relation of arity* $n - k + 1$ *whose domains are the aggregation* $\Delta_{\rho'}$ *and the remaining sets* $\Delta_{j_1}, \ldots \Delta_{j_{n-k+1}}$ *where* $\{j_1, \ldots, j_{n-k+1}\}$ *is* $\{1, \ldots n\} - \{i_1, \ldots, i_k\}$.

Proof. For simplicity, and in order to use examples from our case study, we show how the corresponding reduction is performed on a relation $\rho \subseteq \Delta_1 \times \Delta_2 \times \Delta_3 \times \Delta_4$ of arity 4. Suppose that there is a ternary relation $\rho' \subseteq \Delta_1 \times \Delta_2 \times \Delta_3$ participating in ρ, i.e. for all $x_1 \in \Delta_1, \ldots, x_4 \in \Delta_4$,

$$(x_1, x_2, x_3, x_4) \in \rho \text{ implies } (x_1, x_2, x_3) \in \rho'.$$

Let $\Delta_{\rho'}$ be an aggregation of ρ' with attributes σ_1, σ_2 and σ_3. We can reduce the relation ρ to a binary relation $\widehat{\rho}$ between the aggregations $\Delta_{\rho'}$ and Δ_4 as follows:

$$\widehat{\rho} = \{(r, x_4) | (\sigma_1(r), \sigma_2(r), \sigma_3(r), x_4) \in \rho\}$$

As a corollary of Proposition 1, the relations ρ and $\widehat{\rho}$ are isomorphic. □

Definition 10 (*Arity reduction of a relation*). We say that a n-ary relation ρ reduces to a binary relation σ if $\sigma = \widehat{\rho}$.

Similarly to Proposition 7, we can show that it is always possible to reduce the arity of a n-ary relation.

Proposition 7. *For any n-ary relation* ρ *there exist a binary relation* σ *such that* ρ *reduces to* σ.

Proof. Suppose $\rho \subseteq \Delta_1 \times \ldots \times \Delta_n$. Trivially, $\rho' = \Delta_1 \times \ldots \Delta_{n-1}$ participates in ρ and by Proposition 6 we have that ρ reduces to $\sigma = \widehat{\rho}$ where $\widehat{\rho}$ is the reduction of ρ by $\Delta_{\rho'}$. □

Using the above proposition and by successively applying the process of reduction to the participating relations, one can show that there exists a reduction of the relation where only binary relations are aggregated. As for ternary relations, there are more ways of reducing the arity of a relation than the one shown in the proof above.

As an example, suppose that we want to express the following fact about a triptych altarpiece:

> *St Anne is lying in bed in the central panel of the altarpiece called "The Birth of the Virgin" by Lorenzetti*

The natural representation of this domain property is in terms of a relation *liesOn* of arity 4:

$$(\texttt{lorenzetti} * \texttt{birthofvirgin}, \texttt{field2}, \texttt{anne}, \texttt{bed}) \in \mathit{liesOn}$$

The central panel is identified by `field2`. The altarpiece `lorenzetti * birthofvirgin` belongs to the subclass `MAltarpieces` of `Altarpieces` (see Examples 3, 6 and 12).

We have several participation dependencies: the relations *hasField*, *hasFigure*, *hasObject*, *hasFigureinField* and *hasObjectinField* participate in *liesOn*. We will choose to reify *hasFigureinField* and reduce the relation *liesOn* to a binary relation between the aggregation of *hasFigureinField*. and the set *Objects*. The corresponding ER diagram is shown in Fig. 16. The advantage is that we are reusing the reification of the relation *hasFigureinField* of Example 12. We do not only enforce the dependency of *hasFigureinField* but also the one of *hasField*, *hasFigure* and *hasObject*. This is illustrated in Fig. 17 by the fact that we stack one representation on top of the other. The dependency of *hasObjectinField* illustrated in Fig. 18 will be enforced by adding an axiom similar to the one for *hasObject* and *holds* as follows:

$$\texttt{fieldinaltarpiece}^{-1} \circ \texttt{liesOn} \sqsubseteq \texttt{hasObjectinField}$$

The corresponding representation in OWL is achieved through the reification of the relation $\widehat{\mathit{hasFigureinField}}$. We introduce the concept `FiguresinMAltarpieces` to represent this reification and add all the corresponding axioms (see Definition 2). These axioms are written in OWL Manchester syntax as follows:

```
ObjectProperty: fieldinaltarpiece
  Characteristics: Functional
  Domain: FiguresinMAltarpieces
  Range: FieldsinAltarpieces
  InverseOf: inversefieldinaltarpiece
```

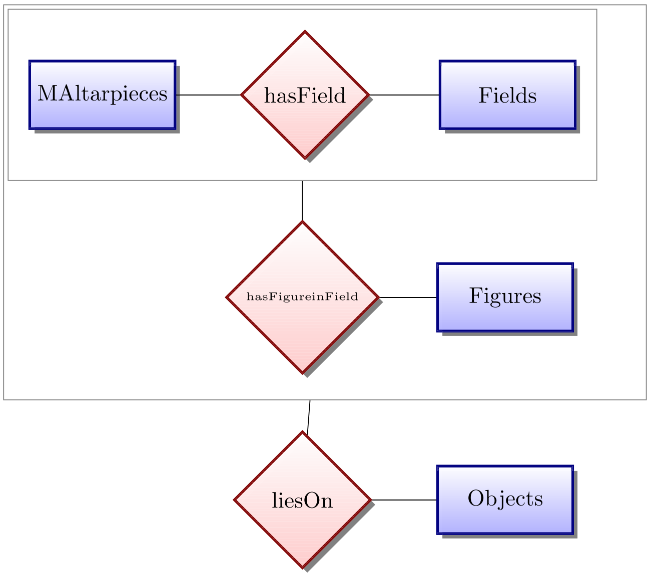

Fig. 16. ER diagram: *liesOn* reduced to a binary relation.

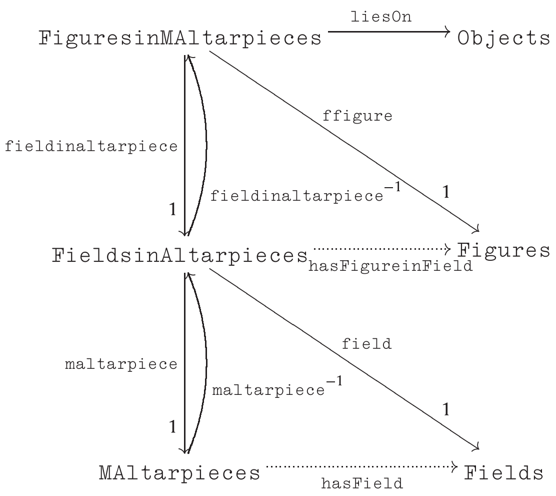

Fig. 17. Representation of *liesOn*.

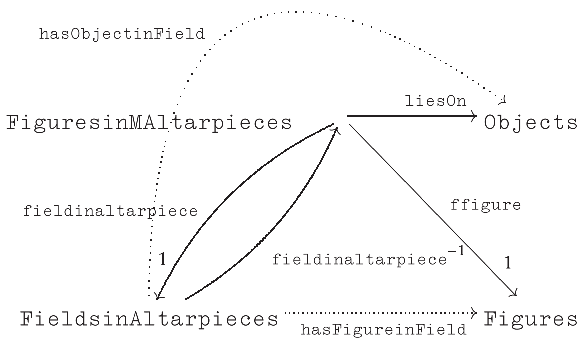

Fig. 18. Capturing overlapping constraints: *hasFigureinField* and *hasObjectinField* participate in *liesOn*.

```
ObjectProperty: ffigure
  Characteristics: Functional
  Domain: FiguresinMAltarpieces
  Range: Figures

ObjectProperty: hasFigureinField
  SubPropertyChain: inversefieldsinaltarpiece ∘
                    ffigure

Class: FiguresinMAltarpieces
  SubClassOf: fieldinaltarpiece
              some FieldinAltarpieces and
              ffigure some Figures
  HasKey: (fieldinaltarpiece, ffigure)
```

Finally, the relation *liesOn* is represented in OWL as a role `liesOn` whose domain is `FiguresinMAltarpieces` and whose range is `Objects`. This is written in OWL Manchester syntax as follows (see also Fig. 17):

```
ObjectProperty: liesOn
   Domain: FiguresinMAltarpieces
   Range: Objects
```

The class `FiguresinMAltarpieces` and its corresponding axioms, can be reused for representing other relations of arity 4 apart from *liesOn*. This is because the relationship *hasFigureinField* participates in many other relations that describe the elements or figures on the field of an altarpiece. For example, consider the sentence *The midservants are washing the newborn Maria in the central panel of the altarpiece "The Birth of the Virgin " by Lorenzetti.* To represent the above property, we need a relation *wash* of arity 4. Because the relation *hasFigureinField* participates in *wash*, we can represent *wash* in OWL similarly to *liesOn* as a role `wash` whose domain is

`FiguresinMAltarpieces`

and whose range is `Figures`.

```
ObjectProperty: wash
   Domain: FiguresinMAltarpieces
   Range: Figures
```

In summary, ontologies obtained by applying our method are easier to extend to include new relations without having to add any reification but reusing existing ones. Moreover, our method leads to enforce the participation constraints.

5. The generalisation construct

In this section, we extend our methodological approach with a construction that is inspired by the mechanism of generalisation introduced in [9] for database design. For simplicity, we consider only binary relations and define generalisation over their ranges. Extending the definition to include domains (not just ranges) and to relations of arbitrary arity is straightforward.

In order to motivate this construct, consider again the relation *hasFigure* introduced in Section 4.1 whose range is *Figures*. Although figures are, understandably, of major interest in altarpieces, there are a number of other elements that play an important role and need to be represented in the ontology – the flowering shaft, tiaras, architectural elements, and so on. For example, we have introduced in Section 4.3 the domain *Objects*, which is used as the range of *holds*. We can then consider a relation *hasObject* similar to *hasFigure* but with *Objects* as range to represent the fact that given objects are depicted in given altarpieces.

However, in certain circumstances, it is more convenient to establish relationships over a more general class *Elements* that consists of *Figures* and *Objects*. For instance, we are interested in representing the fact that

> *The Papal tiara rests on top of the balustrade in the altarpiece Sistine Madonna by Raphael*

For this purpose, it is convenient to define the relation

restson $\subseteq$ *Altarpieces* $\times$ *Elements* $\times$ *Elements*

According to the method that we have been proposing, we should seek to establish a participating relation *hasElement* $\subseteq$ *Altarpieces* $\times$ *Elements* that we would reify. However, given that we already have *hasFigure* and *hasObject*, we should not introduce *hasElement* as an independent relation. Intuitively, *hasElement* is a 'generalisation' of *hasFigure* and *hasObject* and, indeed, we would expect the same to hold between their reifications.

In [9], a generalisation abstraction is introduced as "an adaptation of Hoare's discriminated union structure [14]". This kind of structure, which supports programing language constructs such as Pascal's *variants*, can be explained in terms of the notion of *disjoint union*. In Set Theory, the disjoint union of two sets Δ_1 and Δ_2 is a triple $\langle \Delta_1 \uplus \Delta_2, \iota_1, \iota_2 \rangle$ where each ι_i is an injective function $\iota_i : \Delta_i \to \Delta_1 \uplus \Delta_2$ such that, for any other triple $\langle \Delta', \iota'_1, \iota'_2 \rangle$ with $\iota'_i : \Delta_i \to \Delta'$, there is a unique function $\phi : \Delta_1 \uplus \Delta_2 \to \Delta'$ such that $\phi \circ \iota_i = \iota'_i (i = 1, 2)$. That is, $\Delta_1 \uplus \Delta_2$ is the smallest set that 'contains' both Δ_1 and Δ_2 while distinguishing between the elements that they have in common. The functions ι_i provide the 'tag fields' that, in discriminated unions in the sense of [14], indicate which of the particular constituent sets, Δ_1 or Δ_2, each element of $\Delta_1 \uplus \Delta_2$ originates from.

Naturally, if Δ_1 and Δ_2 are disjoint, their union, together with the corresponding inclusions, is also a disjoint union. In [9], the concept of generalisation applies precisely to disjoint classes to define a superclass. However, for generality, we work with the original definition, which also has the advantage of providing a mathematical structure closer to that of aggregation given in Definition 1.

Definition 11 (*Generalisation*). Let $\rho_1 \subseteq \Gamma \times \Delta_1$ and $\rho_2 \subseteq \Gamma \times \Delta_2$ be binary relations. A generalisation of ρ_1 and ρ_2 is a triple $\langle \rho_3, \iota_1, \iota_2 \rangle$ where:

- $\langle \Delta_1 \uplus \Delta_2, \iota_1, \iota_2 \rangle$ is a disjoint union of Δ_1 and Δ_2
- $\rho_3 \subseteq \Gamma \times (\Delta_1 \uplus \Delta_2)$ is defined by

$$\rho_3 = \{(z, \iota_1(x)) | (z, x) \in \rho_1\} \cup \{(z, \iota_2(y)) | (z, y) \in \rho_2\}$$

We normally use the notation $\rho_1 \uplus \rho_2$ to refer to a generalisation of ρ_1 and ρ_2.

Going back to our example, how can we represent the relation *hasElement* as a generalisation of *hasFigure* and *hasObject*? Consider first the problem of representing *Elements* as a disjoint union of *Figures* and *Objects*. In OWL, the disjoint union of concepts is not available as a primitive construct: it is an abbreviation for a union of two classes with an extra axiom requiring that the classes are disjoint. For example, in the case at hand, we would define:

```
Class: Elements
  DisjointUnionOf: Figures, Objects
```

The system generates an inconsistency whenever the extensions of *Figures* and *Objects* are not disjoint. Unfortunately, OWL does not extend this mechanism to roles (in fact, it does not support the union of roles). Therefore, we cannot express that *hasElement* is the disjoint union of *hasFigure* and *hasObject*.

However, our main interest is not so much *hasElement*, but its aggregation and, ultimately, its reification, as a relation participating in *restson*. Intuitively, given

$$\begin{aligned} &hasElement \subseteq Altarpieces \times Elements, \\ &hasFigure \subseteq Altarpieces \times Figures \\ &hasObject \subseteq Altarpieces \times Objects \end{aligned}$$

where $Altarpieces = \Delta_{hasPainted}$, if $hasElement = hasFigure \uplus hasObject$ then we should also have that $\Delta_{hasElement} = \Delta_{hasFigure} \uplus \Delta_{hasObject}$. This is what we prove next.

Proposition 8. *Let $\rho_1 \subseteq \Gamma \times \Delta_1$ and $\rho_2 \subseteq \Gamma \times \Delta_2$ be binary relations. Let $\langle \Delta_{\rho_1}, \gamma_1, \sigma_1 \rangle$ and $\langle \Delta_{\rho_2}, \gamma_2, \sigma_2 \rangle$ be aggregations of ρ_1 and ρ_2, respectively. Let $\langle \rho_3, \iota_1, \iota_2 \rangle$ be a generalisation of ρ_1 and ρ_2. Finally, let $\langle \Delta_{\rho_1} \uplus \Delta_{\rho_2}, \delta_1, \delta_2 \rangle$ be a disjoint union. Then, $\langle \Delta_{\rho_1} \uplus \Delta_{\rho_2}, \gamma, \sigma \rangle$ define an aggregation of ρ_3 where*

- γ is the unique function $\Delta_{\rho_1} \uplus \Delta_{\rho_2} \to \Gamma$ s.t. $\gamma \circ \delta_i = \gamma_i$ $(i = 1, 2)$.
- σ is the unique function $\Delta_{\rho_1} \uplus \Delta_{\rho_2} \to \Delta_1 \uplus \Delta_2$ s.t. $\sigma \circ \delta_i = \iota_i \circ \sigma_i (i = 1, 2)$.

The above conditions are depicted in Fig. 19.

Proof. The existence of the functions σ and γ results from the universal properties of $\langle \Delta_{\rho_1} \uplus \Delta_{\rho_2}, \delta_1, \delta_2 \rangle$ as a disjoint union. The three conditions of Definition 1 are also easy to prove. □

That is to say, the disjoint union of the aggregations of two relations is an aggregation of the generalisation of the relations.

In OWL, taking the concept `FiguresinAltarpieces` for the reification of *hasFigure* as defined in Example 5, and similarly `ObjectsinAltarpieces` for the reification of *hasObject*, we would introduce a concept

`ElementsinAltarpieces`

for the reification of *hasElement* as follows:

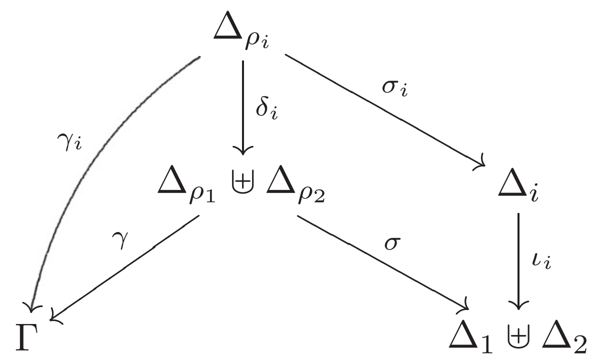

Fig. 19. Disjoint union of aggregations.

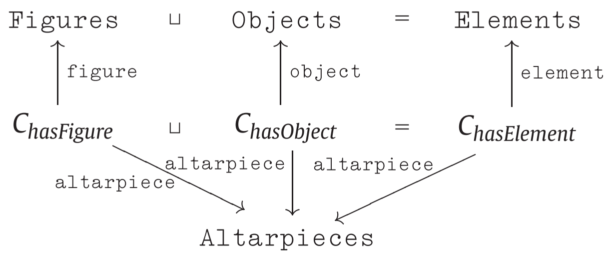

Fig. 20. Generalisation.

```
ObjectProperty: altarpiece
  Characteristics: Functional
  Domain: ElementsinAltarpieces
  Range: Altarpieces
  InverseOf: inversealtarpieces

ObjectProperty: element
  Characteristics: Functional
  Domain: ElementsinAltarpieces
  Range: Elements

ObjectProperty: hasElement
  Domain: Altarpieces
  Range: Elements
  SubPropertyChain:inversealtarpieces ∘ element

Class: ElementsinAltarpieces
  SubClassOf: altarpieces some Altarpieces and
              element some Elements
  HasKey: (altarpiece, element)

Class: ElementswithAltarpieces
  DisjointUnionOf: FiguresinAltarpieces,
                   ObjectsinAltarpieces
```

Notice that, relative to Definition 2, we add that the reification of *hasElement* is the disjoint union of the reifications of *hasFigure* and *hasObject*. A diagram connecting the reifications of the three relations *hasElement*, *hasFigure* and *hasObject* is shown in Fig. 20.

Note that the attribute `altarpiece` is shared by the three classes. We have to declare that the domain of the attribute `altarpiece` is the biggest class which is the reification of *hasElement*.[7]

7 Actually, we should also remove it from the axioms in the reification of *hasFigure* (see Section 4.3) otherwise OWL takes the intersection.

Finally, we investigate how generalisation can work together with the notion of participation and discuss the representation of *restson*.

Definition 12 (*Restriction*). Let $\langle \Delta_1 \uplus \Delta_2, \iota_1, \iota_2 \rangle$ be a disjoint union of Δ_1 and Δ_2, and $\rho \subseteq \Gamma \times (\Delta_1 \uplus \Delta_2) \times \Theta$. We define the restrictions $\rho|\iota_1 \subseteq \Gamma \times \Delta_1 \times \Theta$ and $\rho|\iota_2 \subseteq \Gamma \times \Delta_2 \times \Theta$ as follows:

$$\rho|\iota_1 = \{(x, y_1, z) | (x, \iota_1(y_1), z) \in \rho \wedge y_1 \in \Delta_1\}$$
$$\rho|_{\iota_2} = \{(x, y_2, z) | (x, \iota_2(y_2), z) \in \rho \wedge y_2 \in \Delta_2\}$$

That is, a restriction extracts from a relation involving a disjoint union those triples that involve only the elements of one of the sets. Notice that, in the case in which the sets are disjoint, the functions ι_1 and ι_2 are inclusions, which leads to a simpler formulation of the restrictions. As already explained, this is case that interests us in OWL.

It is easy to see that if a relation participates in another relation ρ, it also participates in any relation that contains ρ. The following proposition is a refinement of this observation for the case of generalisations:

Proposition 9. *Let $\rho_1 \subseteq \Gamma \times \Delta_1$, $\rho_2 \subseteq \Gamma \times \Delta_2$ and $\langle \rho_3, \in c_1, \in c_2 \rangle$ be a generalisation of ρ_1 and ρ_2. Let $\rho \subseteq \Gamma \times (\Delta_1 \uplus \Delta_2) \times \Theta$. The relation ρ_3 participates in ρ iff ρ_1 participates in $\rho|\iota_1$ and ρ_2 participates in $\rho|\iota_2$.*

That is, the participation of a generalisation in another relation can be reduced to the participation of the components of the generalisation in the corresponding restrictions. For example, *hasElement* participates in *restson* iff *hasFigure* and *hasObject* participate in the corresponding restrictions of *restson*.

Following the methodology that we have introduced in Section 4, we can represent the relation *restson* in OWL as a role `restson` whose range is `Elements` and whose domain is `ElementsinAltarpieces`$(= C_{hasElement})$.

$$\texttt{ElementsinAltarpieces} \xrightarrow{\texttt{restson}} \texttt{Elements}$$

```
ObjectProperty: restson
   Domain: ElementsinAltarpieces
   Range: Elements
```

The generalisation *hasElement* is the right conceptualisation because most of the relations used for describing altarpieces are between elements. Using this generalisation, the ontology can be easily extended to include new properties with the same characteristics in the sense that the representation of those properties does not require any further reification. More precisely, we can re-use the reification $C_{hasElement}$ and represent any new relation involving elements of an altarpiece as a role R whose domain is `ElementsinAltarpieces`$(= C_{hasElement})$ and whose range is `Elements`.

$$\texttt{ElementsinAltarpieces} \xrightarrow{R} \texttt{Elements}$$

The reification $C_{hasElement}$ can be used to represent even more relations, indeed any relation in which *hasElement* participates. For example, if we take the inscription of a book on a certain altarpiece to be represented by a string, we can define a relation *hasInscription* in which *hasElement* participates but whose range is the set of strings.

$$\texttt{ElementsinAltarpieces} \xrightarrow{\texttt{hasInscription}} \texttt{String}$$

```
DataProperty: hasInscription
   Domain: ElementsinAltarpieces
   Range: String
```

6. Related work and concluding remarks

In this paper, we proposed a methodological approach for Ontology Engineering aimed at guiding the use of reification as a way of representing *n*-ary relations. Our method simplifies ontologies in the sense that it not only reduces the number of codifications but, more importantly, rationalises the choice of which relations to reify based on dependencies between relations that can be derived from a conceptual analysis of the application domain. This approach promotes reuse through the sharing of reifications of relations that participate in several other relations.

In a nutshell, we advocate that:

- Domain experts should start by building a conceptual model in which they can identify relationships between relations and descriptive attributes.
- Participation dependencies should be identified in those models with a view to reifying the participating relation (as in Sections 4.1 and 4.2).
- A common domain that generalises the domains of several participating relations should also be identified. This would induce a common participating relation generalising all of them that should be reified (as in Section 5).

In order to justify our approach, we provided mathematical definitions of the aggregation and generalisation constructs as used for database design [12,8,9], which we used to analyse the extent to which they can be implemented in languages such as OWL. A number of results were proved that attest to the soundness of the methodological guidelines that we put forward.

Theoretically, our method can always be applied (see Propositions 5 and 7). Since there are many participating relations (at least n for a relation of arity n) it is the task of the ontologist to make the right choice.

From the point of view of the database relational model, we have defined a new notion of integrity constraint called participation constraint. Our method can handle this new notion of integrity constraint. The notion of participation is a restricted form of inclusion dependency between relations [10]. The stronger concept of participation dependency is needed to ensure that the arity of a relation can be reduced without losing information.

An ontology with the examples presented in this paper can be found in [21]. The ontology editor used to write this ontology was Protege 4.1_beta. It is interesting to see how the reasoner (either Hermit or Pellet) can deduce the assertions on the participation relation through the participation constraint.

The use of conceptual modelling primitives in the context of ontologies is not new. For instance, [22] and [23] 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 [24]. However, the author represents aggregations using union of classes, which does not correspond in any way to their original meaning [8]. Our use of aggregation (based on cartesian products) adheres to its use in databases and explores its methodological advantages for conceptual modelling [12].

Other proposals can be found in the literature 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 reifies the whole relation and, in the remaining cases, reifies $B \times C$ 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 adds depth and mathematical rigour to the study of these patterns in the sense that it guides the application of reification by the identification of relations that participate in other relations.

Extensions of description logics with n-ary relations or with aggregations can also be found in the literature [5,16,25]. However, the Web Ontology Language (OWL 2), which is based on the Description Logic of [3], does not provide this capability. OWL 2 provides the possibility of defining n-ary datatype predicates F, albeit in a restricted way [26]. 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 $\lambda(x_1 \ldots x_n).\mathtt{comp}(p,q)$ where $\mathtt{comp} \in \{\leqslant, =, \geqslant, <, >, \neq\}$ 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.

The feedback received from using the method in the construction of the Ontology of Altarpieces 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. Our plans for future work include developing tools for helping ontologist follow the proposed methodology and assist them in the representation of relations of arity n.

References

[1] W3C, OWL 2 Overview, <http://www.w3.org/TR/owl2-overview/>.
[2] I. Horrocks, P.F. Patel-Schneider, D.L. McGuinness, C.A. Welty, OWL: a Description Logic Based Ontology Language for the Semantic Web, in: Baader et al. [27].
[3] I. Horrocks, O. Kutz, U. Sattler, 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, pp. 57–67.
[4] P. Hitzler, M. Krötzsch, S. Rudolph, Foundations of Semantic Web Technologies, Chapman & Hall/CRC, 2009.
[5] D. Calvanese, G.D. Giacomo, Expressive descriptions logics, in: Baader et al. [27], pp. 193–236.
[6] A. Borgida, R.J. Brachman, Conceptual modeling with description logics, in: Baader et al. [27].
[7] P. Severi, J. Fiadeiro, D. Ekserdjian, Guiding reification through aggregation, in: V. Haarslev, D. Toman, G. Weddell (Eds.), Proceedings of the 23rd International Workshop on Description Logics, 2010.
[8] J. Smith, D. Smith, Database abstractions: aggregation, Communications of the ACM 20 (6) (1977) 405–413.
[9] J. Smith, D. Smith, Database abstractions: aggregation and generalization, ACM Transactions of Databases 2 (2) (1977) 105–133.
[10] S. Abiteboul, R. Hull, V. Vianu, Foundations of Databases, Addison-Wesley, 1995.
[11] A. Chasel, La Pala d'Altare nel Rinascimento, Garzanti Editore, 1993.
[12] R. Ramakrishna, J. Gehrke, Database Management Systems, third ed., Mc Graw Hill, 2003.
[13] P. Severi, D. Ekserdjian, Ontology of Altarpieces, Tech. Rep., University of Leicester, 2009.
[14] T. Hoare, Notes on data structuring, APIC Studies in Data Processing (8) (1972) 83–174.
[15] N. Noy, A. Rector, Defining n-ary relations on the semantic web, <http://www.w3.org/TR/swbp-n-aryRelations/>.
[16] G.D. Giacomo, M. Lenzerini, Description Logics with Inverse Roles, Functional Restrictions, and N-ary Relations, in: JELIA '94, Springer, 1994, pp. 332–346.
[17] B. Parsia, U. Sattler, T. Schneider, Easy keys for OWL, in: OWLED, 2008.
[18] D. Calvanese, M. Zakharyaschev, Course on answering queries in description logics: theory and applications to data management, in: 22nd European Summer School in Logic Language and Information (ESSLLI 2010), 2010.
[19] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, A. Poggi, M. Rodriguez-Muro, R. Rosati, Ontologies and databases: The DL-lite approach, in: S. Tessaris, E. Franconi (Eds.), Semantic Technologies for Informations Systems – Fifth Int. Reasoning Web Summer School (RW 2009), Lecture Notes in Computer Science, Vol. 5689, Springer, 2009, pp. 255–356.
[20] B. Motik, I. Horrocks, U. Sattler, Bridging the gap between OWL and relational databases, Journal of Web Semantics 7 (2) (2009) 74–89.
[21] P. Severi, Ontology for guiding representations of relations of arity n, <http://www.cs.le.ac.uk/people/ps56/AggregationMethod.owl>, April 2011.
[22] U. Sattler, D. Calvanese, R. Molitor, Relation with other formalisms, in: Baader et al. [27], pp. 149–192.
[23] A. Borgida, M. Lenzerini, R. Rosati, Description logic for databases, in: Baader et al. [27], pp. 149–192.
[24] C. Veres, Aggregation in ontologies: practical implementations in OWL, in: ICWE 2005, LNCS, Springer, 2005.
[25] G.D. Giacomo, M. Lenzerini, What's in an aggregate: foundations for description logics with tuples and sets, in: IJCAI, vol. 1, 1995, pp. 801–807.
[26] B. Parsia, U. Stattler, OWL 2: Data range extension: linear equations, <http://www.w3.org/TR/2009/NOTE-owl2-dr-linear-20091027/>.
[27] F. Baader, D. Calvanese, D.L. McGuinness, D. Nardi, P.F. Patel-Schneider (Eds.), The Description Logic Handbook: Theory, Implementation, and Applications, Cambridge University Press, 2003.