



# School of Computer Science

Faculty of Science and Agriculture,  
University of KwaZulu-Natal

School of Computer Science, University of KwaZulu-Natal, Durban, South Africa  
Tel: (+27) 031 260 7136, Fax: (+27) 031 260 7001, Email: [enquiry@cs.ukzn.ac.za](mailto:enquiry@cs.ukzn.ac.za)

*UKZN School of Computer Science Technical Report:*

## Enhancing Identification Mechanisms in UML Class Diagrams with Meaningful Keys (extended version)

C. Maria Keet

<b>Affiliation</b>	School of Computer Science University of KwaZulu-Natal South Africa
<b>Corresponding author</b>	<a href="mailto:keet@ukzn.ac.za">keet@ukzn.ac.za</a>
<b>Keywords</b>	UML Class Diagram, Keys, Identification, Ontology, Defined Class, Conceptual Data Modeling
<b>Number</b>	SoCS11-1
<b>Date</b>	August 8, 2011
<b>URL</b>	<a href="http://www.cs.ukzn.ac.za/">http://www.cs.ukzn.ac.za/</a>

© School of Computer Science, University of KwaZulu-Natal. This work may not be copied or reproduced in whole or part for any commercial purpose. Permission to copy in whole or part without payment of fee is granted for non-profit educational and research purposes provided that all such whole or partial copies include the following: a notice that such copying is by permission of the School of Computer Science, University of KwaZulu-Natal, South Africa; an acknowledgement of the authors and individual contributors to the work; all applicable portions of this copyright notice. Copying, reproducing, or republishing for any other purpose shall require a license with payment of fee to the School of Computer Science, University of KwaZulu-Natal.

# Enhancing Identification Mechanisms in UML Class Diagrams with Meaningful Keys (extended version)

C. Maria Keet  
School of Computer Science  
University of KwaZulu-Natal  
Durban, South Africa  
keet@ukzn.ac.za

## ABSTRACT

The task of declaring good keys is more relevant for ER and ORM than UML, because the former two require a modeller to do so in the methodology or CASE tool, whereas UML uses internal, system-generated, identifiers, with a little-known underspecified option for user-defined identifiers. This raises questions, such as which semantic identification mechanisms should be included and how to incorporate it in UML to foster consistent usage. The distinct implicit assumptions and explicit formalisations of extant identification schemes are elucidated by considering both ontology and (E)ER and ORM. Whereas ontology seeks identity, conceptual data modelling uses strong or weak identification of entities. To increase the ontological foundations of UML, the more detailed insights lead to a proposal for two language enhancements for UML, being formally defined *simple* and *compound identifiers* and the notion of *defined class*, which also have a corresponding extension of UML's metamodel.

## Categories and Subject Descriptors

I.2.4 [Knowledge Representation Formalisms and Methods]: Representation languages; H.1.1.m [Information Systems]: Models and Principles—*Miscellaneous*; H.2 [Database Management]: Data Models

## Keywords

Identity, Identification, Key, Defined Class, Quality of models and their languages

## 1. INTRODUCTION

In conceptual data modelling literature, identification and identity are often used interchangeably and despite the long history on the topic, there is a remarkable lack of agreement about what they are, how identification mechanisms—keys, reference schemes, identifiers—are dealt with in graphical languages such as UML and EER, how to formalise it, and how modelling tools implement them. Ascertaining when a

particular key is a good one has been investigated before, and the reader is referred to [17, 26] for exemplary and lucid examples of good and bad keys. However, these and other works on identity and identification in conceptual data modelling do not address the logical and ontological underpinnings of identity and identification with keys, i.e., whether the identification mechanism itself is a good one or not. In addition, one needs to decide whether the *procedure* to find and represent identity, or at least good keys, should be (i) be a step in the modelling methodology; (ii) also be enforced in the CASE tool; or (iii) be part of the metamodel and/or in the conceptual modelling language itself, be it graphically, in its logic-based counterpart, or with a conceptual model validator, and whether that should be the same for all conceptual data modelling languages. A uniform, or at least a structured and unambiguous approach, can reduce or even avoid inconsistencies in a conceptual data model and achieve interoperability through less resource-consuming information integration. The present lack of consistency in handling identity and identification impedes this due to its unsystematic and differing approaches to it.

Considering the UML v2.3 specification [23] in particular, it contains many occurrences of the terms “identity”, “identification”, and “identifier”, but does not provide detail on how exactly the identification system is implemented, other than using the *internal identifier* mechanism where the identifiers are generated and assigned by the system. This poses challenges for devising subclass hierarchies—a subclass with a semantically incompatible identity is incorrect with respect to the information, but remains erroneously in the model without further machinery to detect it—and for application integration, because it requires a manual re-assessment of the information and each data point to elucidate which objects are the same. Concerning the latter, the recent data integration problems of the disparate software systems for the Johannesburg area integrated services delivery information management system may serve as an example of the pressing need to resolve this; at present, there are no tools that negotiate automatically which objects across software systems represent the same entity in reality, which otherwise could have alleviated the problems and have speed up the integration process. Concerning the former, take, e.g., the case where one may have modelled that a **Social** entity (e.g., SABC) is a subclass of **Group of people**: instances of the former can change its members whilst being the same object, but the latter cannot, hence, they have different identity criteria, which thus will cause problems in the OO system if implemented as such nevertheless. Or take a typical problem

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

A shorter version of this paper is published at SAICSIT '11, October 3-5, Cape Town, South Africa

Copyright 2011 ACM X-XXXXX-XX-X/XX/XX ...\$10.00.

with `Location` that subsumes `Country` in some UML diagram: a particular location is a fixed region that can be captured by its shape and coordinates, which clearly is not the case for a country through time—instead, identification of a `Country` revolves around it being a political administrative entity that is located *at* some geographical region (more examples can be found in [14]). This problem has been addressed for ontologies with the OntoClean method [14], but there is no usable counterpart in conceptual data modelling regarding the detection of such incompatible identity criteria, because there is no unambiguous way to deal with identity criteria.

Ontology-driven information systems [10] and conceptual data modelling are gaining momentum [1, 11, 15, 18], where it is demonstrated that reusing notions from Ontology leads to better quality conceptual data models. Here we take a similar approach aiming at addressing the issue of identity and identification in UML Class Diagrams so that the subject domain semantics can be represented more comprehensively and thereby foster the development of better quality UML Class Diagrams that can generate both better software systems and that can be more easily aligned or integrated with other information systems. We do this by, first, assessing the notions of identity and identification from ontology and knowledge representation, and, second, examining two other conceptual data modelling languages that already have more identification features in their languages (EER and ORM2), comprising step-wise ‘simplifications’ from the practically difficult notion of identity from philosophy to the identification mechanisms for conceptual data modelling. These more detailed insights will be used to propose two language enhancements for UML that are intended for use in UML Class Diagrams, being formally defined *simple* and *compound identifiers* and the notion of *defined class*, which also have a corresponding extension of UML’s metamodel.

The remainder of the paper is organised as follows. We assess ontological aspects of identity in Section 2 and proceed to identification mechanisms in conceptual data modelling in Section 3. Extensions and guidelines are proposed in Section 4, and we close in Section 5.

## 2. IDENTITY AND IDENTIFICATION IN ONTOLOGY

We consider philosophical and artificial intelligence aspects of identity and identification so as to lay a foundation from ‘theoretical ideal’ toward applicability, and to clarify terminology. The first paragraph provides an informal summary and introduces some terminology and the second paragraph addresses logical and language aspects.

### 2.1 Different identities

Ultimately, one wants to be able to identify objects in reality and to be able to say if objects *a* and *b* are identical. Within an information system, this is reduced to determining if two *descriptions* (e.g., tuples) *a* and *b* are the same, where, if this is indeed the case, *a* and *b* are assumed to refer to one object in reality, and if not, then each one refers to a different object. At the type-level in an ontology, the intension has to be defined in such a way so as to not only distinguish one type from another but also that at any given time, the individuals in the extension can be distinguished, hence, at least identified as distinct members. In Ontology, we then

have, at least, a unary property  $\phi$ , which loosely matches what is called class in UML and the Web Ontology Language (OWL), entity type in ER and Object-Role Modeling (ORM), and concept in Description Logics (DL). Strictly speaking, though, a *class* is a set that in ontology terminology may correspond to a universal<sup>1</sup> or a concept<sup>2</sup> in its intension. Although conceptual modelling for most information systems aims to represent a piece of reality, i.e., represent universals, this need not be the case (e.g., a conceptual model about deities of different religions in the stone age). Thus, when we develop a conceptual data model, we deal at least with concepts, certainly with classes, and, depending on the subject domain, also with universals. In all cases, however, one uses the simpler notion of naming a ‘collection of properties’, be such entities referred to as a class in UML or entity type in ORM and EER, or concept in DL.

Concerning identity, philosophers have proposed several ways to identify objects and what identifying means. We summarize only the three most relevant ones for the current scope. Numerical identity, in the “classical” philosophical sense, is a single relation an object has to itself and nothing else; e.g., two material objects are the same if they occupy the same spatial region [21]. This definition is circular and has been subject to much debate, extensions, and less restricted notions because it is deemed nigh on unapplicable [21, 8]. More relaxed notions are qualitative and relative identity. In short:

- *Numerical identity*: objects not only share all properties but is assumed to be *exactly one* single characteristic; e.g., sets are the same if they have the same members;
- *Qualitative identity*: objects have *some properties* (including attributes) in common so that they can be more or less qualitatively identical, such as that all *Lepidoptera* (moths, butterflies) have *scaled wings* and *exactly four life phases* in their normal full life;
- *Relative identity*: two objects can be the same kind of thing *X* but they are not the same *Y*, i.e., they share *a property*, a ‘kind of thing’ such as the sortal<sup>3</sup> *Apple* that is not a mere attribute like *Colour*.

Relating this to conceptual data modelling, then ascertaining qualitative or relative identity aids modelling a taxonomy, whereas numerical identity for objects is reduced to comparing their computerised representation within a closed subject or application domain. Thus, in traditional conceptual modelling we focus on being able to uniquely identify the instances within the information system only. If, however, we

<sup>1</sup>There are several definitions for *universal*; for instance, that it is a combination of properties. One can refine this imprecise description, e.g., “Universals are a class of mind independent entities, usually contrasted with individuals, postulated to ground and *explain relations of qualitative identity* and *resemblance among individuals*. Individuals are said to be similar in virtue of sharing universals.” [19] (emphasis added).

<sup>2</sup>The term *concept* in Ontology generally refers to mind-dependent entities [24].

<sup>3</sup>A sortal is defined by philosophers in different, though closely related, ways. It gives a criterion for counting the items of that kind, it answers the question ‘what is it?’ for things of that kind, and/or is said to specify the essence of things of that kind [9]. E.g., one cannot sort things and identify, say, greenish objects and ascertain what they are based on their green-ness (being green is a property of something else), but one can ‘sort’ (identify) fruit in a fruit basket into apples, oranges, and other fruit.

also avail over automated reasoners to check satisfiability of conceptual data models (among other reasoning services), we also check equivalence of the concepts at the type-level (e.g., [4]). Combining this with qualitative and relative identity, we then arrive at a point where some specific collection of properties ('attributes') are described for universal (/concept/class/etc.)  $X$  that enables us to say that *if* the values of the identifying properties are the same *then* they denote the same object, and vice versa. This may be a daunting task for real life objects, but is doable in information systems to some extent.

In addition to these three notions of identity, there are also synchronic and diachronic identity:

- *synchronic identity* enables one to distinguish between objects at a single point in time; e.g., Mary from John,
- *diachronic identity* is to establish that two entities are the same at two different points in time; e.g., Mary as child and as adult.

The former is commonly used in applications, the latter is useful in particular for temporal conceptual modelling and information systems.

To make a further step toward accommodating conceptual modelling, we can play with the *criterion of identity* to require either only necessary or only sufficient conditions [21], which Guarino and Welty [13] call "weak identity". For instance, a *necessary condition* for Hominid is that each such an animal has a spinal column, but it is not a sufficient condition because there are vertebrates that are not hominids. A *sufficient condition* to identify persons in a university database could be their social security number, i.e., if the object has a SSN then it will be classified as a person, although it is not necessary for an object (in reality) to be an instance of **Person** only if it has a social security number and it does not say what are the necessary conditions for a person to be a person. The SSN is, in fact, an *artificial identification* mechanism but not identity in the original meaning. Observe also the usage of the term 'necessary' property  $\phi$ : necessary in the ontological sense with an open world assumption means indeed *all such individuals in the world* must necessarily (modal operator  $\Box$ ) have or be a  $\phi$ , whereas necessary in the conceptual modelling sense for a software application and normally using a closed world assumption amounts to a mandatory constraint that *at least those objects that will be represented in the application* must have (existential quantification  $\exists$ ) or be a  $\phi$ ; it may be that the two coincide in reality, but this need not be the case.

Thus, we arrive at two distinct ways that can be used in the modelling to make sure and ascertain that objects and their concepts are unique: seek a combination of properties that are necessary for the concept and its instances, or invent or find an identification scheme that suffices.

## 2.2 Representing it

We introduce basic definitions from philosophy and AI first, and subsequently assess what remains of these notions in ontology engineering, the Semantic Web in particular, and conceptual modelling languages.

The "classical view" of identity can be represented such that  $\forall x.R(x, x)$  and  $\forall xy.R(x, y) \rightarrow (\phi(x) \rightarrow \phi(y))$  hold where  $x$  and  $y$  are extensionally equivalent,  $R$  is some arbitrary binary relationship and  $\phi$  an unary property [21]. The predicate can be verbalised as a "x and y being the same with respect to R" and in case of numerical identity, there is only one such  $R$ ,

e.g., material objects ( $x$  and  $y$ ) and occupying the same place ( $R$ ). With identity being a relation, the afore-mentioned criterion of identity is commonly formalised as follows (attributed to Lowe [21]):

*Definition 1. (Criterion of identity)*  $\forall xy.\phi(x) \wedge \phi(y) \rightarrow (x = y \leftrightarrow R(x, y))$

That is, if  $x$  and  $y$  instantiate the same property  $\phi$  then  $x = y$  if and only if they relate through  $R$ . Only a few examples of criteria for identity have been proposed in the literature, most notably the one about material objects and that sets are the same if they have the same members [6, 21]. In general, however, a single criterion of identity is deemed too strong and practically not applicable. A simpler, more realistic, approach can be taken by separating the double implication into a so-called necessary and a sufficient condition [21], which, loosely, say that "if  $x$  and  $y$  instantiate the same property then if  $x$  and  $y$  are identical then they relate through  $R$ " and "if  $x$  and  $y$  instantiate the same property then if they relate through  $R$  then they are identical":

*Definition 2. (Necessary condition)*  $\forall xy.\phi(x) \wedge \phi(y) \rightarrow (x = y \rightarrow R(x, y))$

*Definition 3. (Sufficient condition)*  $\forall xy.\phi(x) \wedge \phi(y) \rightarrow (R(x, y) \rightarrow x = y)$

These definitions, while weakening the criterion of identity, are not satisfactory, for ideally one should have a notion of identity criterion (IC) that takes into account both synchronic and diachronic identity so as to be as general as possible, one would need an existence predicate for the entities ( $E(x, t)$ ), and one should bind it to a sortal that is a rigid property<sup>4</sup> ( $\phi$ ) [13]. This led Guarino and Welty to introduce their notion of weak identity [12, 13] (which are used for OntoClean [14]). We combine the various components of [12] into one explicit definition for *weak identity*:

*Definition 4. (Weak identity)* Let  $\phi$  be a rigid property,  $\Gamma$  the identity criterion and such that  $\neg\forall xyt't'.\Gamma(x, y, t, t') \leftrightarrow x = y$ , then  $\Gamma$  is a weak identity criterion iff either  $\Gamma$  is a necessary IC carried by  $\phi$  when:

$$E(x, t) \wedge \phi(x, t) \wedge E(y, t') \wedge \phi(y, t') \wedge x = y \rightarrow \Gamma(x, y, t, t') \quad (1)$$

$$\neg\forall xy.E(x, t) \wedge \phi(x, t) \wedge E(y, t') \wedge \phi(y, t') \rightarrow \Gamma(x, y, t, t') \quad (2)$$

or  $\Gamma$  is a sufficient IC carried by  $\phi$  when:

$$E(x, t) \wedge \phi(x, t) \wedge E(y, t') \wedge \phi(y, t') \wedge \Gamma(x, y, t, t') \rightarrow x = y \quad (3)$$

$$\exists xyt't'.\Gamma(x, y, t, t') \quad (4)$$

where  $t$  and  $t'$  are points in time, so that for synchronic weak identity  $t = t'$  and for diachronic weak identity  $t < t'$ , and  $E$  denotes the existence of the objects.

Regarding a non-rigid property<sup>5</sup>, it carries an IC  $\Gamma$  iff it is subsumed by a rigid property carrying  $\Gamma$  [12]. For instance,

<sup>4</sup>A rigid property is a property that is essential to *all* its instances ( $\forall x\phi(x) \rightarrow \Box\phi(x)$ ), e.g., **Person**, **Apple**.

<sup>5</sup>A non-rigid property is a property that is not essential to some of its instances. Of higher relevance in practice is an *anti-rigid* property, which is not essential to *all* its instances ( $\forall x\phi(x) \rightarrow \neg\Box\phi(x)$ ) [12], e.g., being a **Student** (for all students it holds that at some point in their existence they were not a student); cf. footnote 3.

the non-rigid *Butterfly* carries identity inherited from its rigid parent class *Papilionoidea*. This attempt, however, is also not free of problems, which are discussed, but not solved, in [6] who doubt if the matter on identity and sortals can be fixed at all. While philosophically still under investigation, the notions of necessary or sufficient conditions are already useful and are being used in knowledge representation.

Let us take a step toward implementations by considering ontologies in the sense of engineering artifacts, and those formalised in a Description Logics (DL) language in particular, such as OWL ontologies for the Semantic Web. In DLs, we have so-called *primitive* (denoted with  $\sqsubseteq$ ) and *defined* (denoted with  $\equiv$ ) concepts [2], where the latter is to mean that that concept has *all* necessary *and* sufficient properties declared—hence, not necessarily one single elusive *R* as in Definition 1—and the former has no or only some properties defined that together are not sufficient to identify the instances. For instance, one could define *Omnivore* as an animal that eats (parts of) plants and animals:

$$\text{Omnivore} \equiv \text{Animal} \sqcap \exists \text{eats} . (\text{Animal} \sqcup \text{Plant}) \sqcap \exists \text{eats} . (\exists \text{partOf} . \text{Plant}) \sqcap \exists \text{eats} . (\exists \text{partOf} . \text{Animal}) \sqcap \forall \text{eats} . (\text{Animal} \sqcup \text{Plant} \sqcup \exists \text{partOf} . \text{Plant} \sqcup \exists \text{partOf} . \text{Animal})$$

On the other hand, we have only several necessary conditions for *Plier*, hence, it is a primitive concept:

$$\text{Plier} \sqsubseteq \text{AssistiveDevice} \sqcap = \text{hasPivotAngle} . \text{Angle} \sqcap \exists \text{hasFunction} . (\text{Bending} \sqcup \text{Cutting} \sqcup \text{Holding})$$

Thus, in principle, the aim is to represent qualitative identity, not identification. The OWL 2 W3C recommendation [20] allows both primitive and defined concepts through *SubclassOf*(C CE) and *EquivalentClasses*(C CE), respectively, where C is the primitive/defined class and CE the class expression. Noteworthy is that within the setting of OWL, one can also choose to use a *user-defined artificial identification scheme* by means of a single attribute using OWL’s *DataProperty*. For instance, one can define the data property *hasRegNr* with as range the data type *nonNegativeInteger*, and declare it functional and inverse functional and mandatory so that each plier has exactly one registration number and for each registration number there is exactly one plier.

Conceptual modelling languages do not have a feature to declare a class to be a defined class like in OWL or to represent a criterion of identity, although for each concept/class/entity type it is possible to describe one or more attributes (mostly quality properties) so as to have a set of ‘necessary’ (mandatory) conditions. (E)ER and ORM also allow or even require one to define sufficient conditions, better known as *keys*. There are, however, differences between how EER, UML, and ORM implement this, both in the language and in the modelling methodology; we elaborate on such details in Section 3. One could reuse the notion of defined concept and add it to conceptual modelling languages to make explicit the distinction between primitive and formally defined concepts, or assume that all declared properties/attributes constitute the necessary and sufficient conditions, or deem it unnecessary for information systems modelling to identify objects in this way and assume enforced user-specified identification suffices; we shall return to this issue in Section 4.3.

Compared to explicitly defined concepts and necessary conditions in ontology, using an identification scheme in conceptual data modelling can occur automatically in the background (UML tools), be enforced upon the user by ORM tools through the notion of a manually specified *reference*

*scheme* for each entity type, or embedded in the modelling methodology to assign keys (EER). Thus, identification is the norm rather than the exception for conceptual models. A natural next question then is how to devise a good identification system for representing a sufficient condition (in the sense of Definition 3) in a conceptual model. This has been well-studied for implementations with a symbol notation-based naming scheme [26], which also describes requirements that a good identification scheme should satisfy—singular reference, singular naming, rigid reference, rigid naming and the monotonic designation—that comply roughly with the philosophical notions underlying identity, which are more precisely represented with Definition 4. A gap lies between philosophy and implementations, i.e., the solid foundations of identity and identification in conceptual data models, which we are working toward closing.

### 3. IDENTIFICATION IN EER, ORM, AND UML

Mandatory and sufficient conditions are incorporated in conceptual models in different ways. We analyse, make more precise, and discuss them in the remainder of this section, considering simple keys and complex keys. This, in turn, enables one to refine the identification mechanism in UML and harmonise them with other conceptual data modelling and knowledge representation languages, which will be discussed in Section 4.

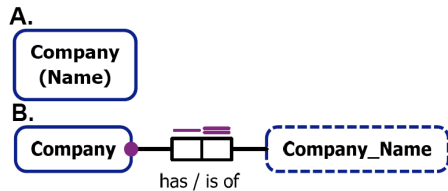
#### 3.1 Simple reference schemes

Object-Role Modelling (ORM) is most explicit in its use of an identification mechanism, which is also enforced through diagram validation in the two well-known ORM CASE tools VisioModeler 3.1 and NORMA. Each entity type (denoted with a solid roundtangle) has a so-called *reference scheme*, which in its simplest form—then dubbed *reference mode*—is a binary relationship from that population to the values of some value type (denoted with a dashed roundtangle), which was originally just a string [16] but has been extended to other data types, with a *mandatory* participation by the entity type and a *1:1 uniqueness* constraint; Fig. 1-A and B depict the compact and expanded graphical representation, respectively. The double line indicates that that reference mode is chosen as primary, for one also can include other reference schemes. Let us formalise this as follows:

*Definition 5. (ORM reference mode (REFM)) An ORM reference mode for an object type is a identification mechanism that provides a sufficient condition for identity such that: let C be the object type, V a value type, I relating C and V, i.e.,  $\forall x, y (I(x, y) \rightarrow C(x) \wedge V(y))$ , and constrained to  $\forall x (C(x) \rightarrow \exists y I(x, y))$ ,  $\forall x, y, z (I(x, y) \wedge I(x, z) \rightarrow y = z)$ , and  $\forall x, y, z (I(x, y) \wedge I(z, y) \rightarrow x = z)$ .*

The single-attribute primary key, *simple key*, in ER and EER has essentially the same definition as ORM’s REFM but without the mandatory constraint [7]:

*Definition 6. (ER/EER simple key (SKEY)) An ER/EER simple key for an entity type is an identification mechanism that provides a sufficient condition for identity such that: let C be the entity type, V a data type, I relating C and V, i.e.,  $\forall x, y (I(x, y) \rightarrow C(x) \wedge V(y))$ , and constrained to  $\forall x, y, z (I(x, y) \wedge I(z, y) \rightarrow x = z)$  and  $\forall x, y, z (I(x, y) \wedge I(x, z) \rightarrow y = z)$ .*



**Figure 1: A: Compact representation of ORM’s reference mode, with the entity type *Company* identified by its *Name*; B: expanded representation of the reference mode, drawn with the NORMA tool.**

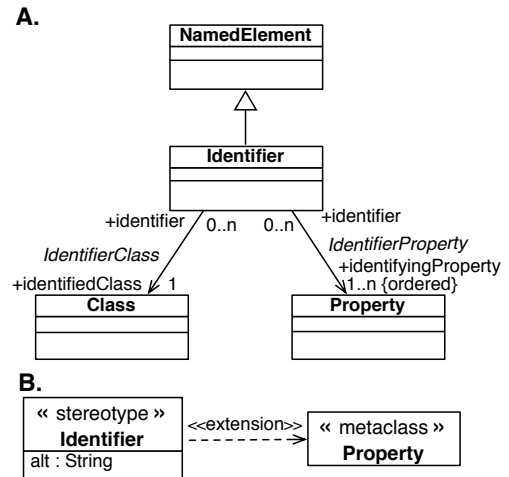
Noteworthy is that in DL-inspired definitions for ER (e.g., [5]) and for the *Haskey* feature in OWL 2 [20], the semantics is a mandatory 1:n constraint for both (OWL 2 also requires named instances for *Haskey* and does not adhere to the Unique Name Assumption (UNA), which leads to different deductions compared to keys in the conceptual modelling and database setting that do use the UNA (see [20] for details)). Hence, we arrive at the following definition for such an ER/EER ‘key’, which we call *object uniqueness* to distinguish it from keys in ER and ORM:

*Definition 7. (ER/EER object uniqueness (OBJU)) ER/EER object uniqueness for an entity type is a weak identification mechanism that provides a sufficient condition for identity such that: let  $C$  be the entity type,  $V$  a data type,  $I$  relating  $C$  and  $V$ , i.e.,  $\forall x, y(I(x, y) \rightarrow C(x) \wedge V(y))$ , and constrained to  $\forall x(C(x) \rightarrow \exists y I(x, y))$  and  $\forall x, y, z(I(x, y) \wedge I(z, y) \rightarrow x = z)$ .*

It is worth elaborating on this difference. OBJU meets the requirement ‘given an attribute value  $v$ , retrieve a unique object  $o$ ’, i.e., for each value in the population of  $V$ , there is at most 1 object in  $C$ , but it does not constrain how the objects in  $C$  participate in  $I$ . However, for a reliable identification scheme, being ‘given two objects  $o_1$  and  $o_2$ , ascertain if they are the same or different’, one needs at least a 1:1 constraint, not 1:n (and mandatory, if unidentified objects are not allowed), because with the 1:n, we can have two values,  $v_1$  and  $v_2$ , pointing to  $o$ , but with that ‘key’ alone one cannot ascertain whether both values refer to the same object. Put differently, the key should be able to be used for counting the distinct objects, but if we do that with the values of  $I$  in OBJU and its 1:n, we may count more objects than there actually are. For instance, say, Vodacom would ‘identify’ a customer by her phone number. With OBJU, we obtain that for each phone number there is one customer; however, it may well be that a customer has two or more working Vodacom SIM cards, yet obviously, there is just that one customer (a phone number ID may be suitable to count the active SIM cards, instead of customers). With OBJU, we cannot detect this, but with an appropriate SKEY/REFM, we can—and find out that phone numbers cannot be used as identification of customers but, say, a customer’s valid SA ID number can such that one does not count objects more than once. The reasons why OBJU sufficed for the DL scope at the time are the implicit assumption of *internal identifiers* with a 1:1 constraint that take care of the real identification of the objects and the assumption that one has good conceptual models that are, in relational model terminology, at least in 4NF. Wieringa and de Jonge [26]

list several issues with internal identifiers, of which the lack of (subject domain) semantics for identification is, in the current context, the most problematic. Primarily, it may make modelling easier, but conducting a conceptual analysis on how one does identify objects will not only bring afore more of the implicit semantics but also add to correctness of the data thanks to decreasing the chance of redundancy and increasing interoperability.

We mentioned in the introduction that UML uses the *internal identifier* mechanism where the identifiers are assigned by the system. Accordingly, one neither can specify manually if that should occur under the constraints of REFM or SKEY (or even OBJU). A proposal to insert a user-defined identification was made by Halpin and Morgan [17] through appending the chosen preferred attribute with “{P}”. If this were to be added to UML proper, then such an attribute would not be primary—the internal identifier is—but an *alternative* identifier that bears some subject domain semantics compared to the internal identifier, i.e., be slightly closer to the notion of identity in ontology. The UML Ontology Definition Meta-model (ODM) v1.0 [22] proposes the notion of alternative identifiers, depicted diagrammatically in Fig. 2-A, and has an optional UML package where *Property* is extended with *Identifier* and where *alt:string* is the “name of instance of alternative identifier if there are more than one identifiers for a given class” (Fig. 2-B). A problem with this proposal is that it does not specify multiplicity constraints on the identifier and it does not distinguish between simple, multi-attribute, and external keys.



**Figure 2: A: Suggested extension for identifiers by the OMG [22] (Fig B-1, p291); B: its UML profile [22] (Fig B-2, p292).**

### 3.2 Complex keys in ORM and EER

ORM and EER have additional features to describe more advanced identification than just REFM and SKEY. These include  $n$ -ary keys with  $n > 1$ , weak entity types, and external uniqueness, which are at least within the scope of sufficient conditions, although one cannot exclude *a priori* that some of them may actually represent both the necessary and sufficient conditions for identity.

### 3.2.1 N-ary keys

SKEY has been extended to keys that span more than one attribute in EER, i.e., for an entity type with  $n$  attributes,  $n \geq 2$ , then the key spans at most  $n$  attributes. This can be formalised as follows:

*Definition 8. (Multi-attribute key (MKEY)) An ER/EER multi-attribute key for an entity type is a identification mechanism that provides at least a sufficient condition for identity such that: let  $C$  be the entity type,  $V_1, \dots, V_n$  data types,  $I$  relating  $C$  and  $V_1, \dots, V_n$ , i.e.,  $\forall x, y_1 \dots y_n (I(x, y_1, \dots, y_n) \rightarrow C(x) \wedge V_1(y_1) \wedge \dots \wedge V_n(y_n))$ , and constrained to  $\forall x (C(x) \rightarrow \exists y_1, \dots, y_n I(x, y_1, \dots, y_n))$  and for each  $V_i$ , with  $2 \leq i \leq n$ , denoted together with  $\bar{y}$  that participate in the key (and with  $\underline{y}$  those attributes that do not),  $\forall x_1, x_2, \bar{y}, \underline{y} (I(x_1, \bar{y}, \underline{y}) \wedge I(x_2, \bar{y}, \underline{y}) \rightarrow x_1 = x_2)$  and  $\forall x, \bar{y}, \underline{y}, \bar{z}, \underline{z} (I(x, \bar{y}, \underline{y}) \wedge I(x, \bar{z}, \underline{z}) \rightarrow \bar{y} = \bar{z})$ .*

Recollecting the discussion with SKEY and OBJU, MKEY also enforces a mandatory 1:1 so as to follow ontological principles more closely. With a minor variation in the definition ( $1 \leq i \leq n$ ), one can generalise to an ‘EER key’ that subsumes both MKEY and SKEY. ORM’s analogue of MKEY is referred to as an internal uniqueness constraint that spans  $> 1$  ORM roles where the other types are not constrained to “ $V_1, \dots, V_n$ ” but may also be object types.

These type of keys are also referred to as *natural keys* or *semantic identifiers* for they seek to use real attributes of the objects to identify them, compared to artificial keys like a student matriculation number. Put differently, if we would not define the keys, we have a list of mandatory (possibly also necessary) conditions that contribute to a description of the identity of the objects, in the same fashion as the Plier example for OWL.

### 3.2.2 Weak entity types and external uniqueness

The ontological status of ER/EER’s weak entity types and ORM’s external uniqueness with a compound reference scheme is less clear. In EER, one speaks of the ‘owner’ and ‘identifying relationship’ of a weak entity type, which corresponds roughly to UML’s qualified association. However, there need not be one ‘owner’, nor that the weak entity types should be modelled away for implementation convenience [3] (with the cost of introducing an attribute of an attribute, and no equivalence has been proven). While weak entity types require more than the simple identification schemes REFM and SKEY, as they rely to a greater extent on other information represented in the conceptual model, it does not imply that therefore they have to be considered dependent entities<sup>6</sup>. Let us take ORM’s formalisation of standard external uniqueness with a compound reference scheme [16] and place this in a more comprehensive and precise definition that also respects ORM CASE tool implementations:

*Definition 9. (Compound reference scheme (CREFS)) A compound reference scheme is an identification mechanism*

<sup>6</sup>The argument is long and omitted here. We acknowledge there exist narrower views of weak entity type than taken here; e.g., where there is intrusion of system behaviour (if an instance of the ‘owner’ is deleted, then so must its related object that instantiates the related weak entity type) [25], or only “moment universals” or so-called “modes” are equated with weak entity type, which inhere in their bearer, such a particular disease symptom inheres only in that patient [15].

that provides a sufficient condition for identity, where an object type  $C$  participates in  $m$  roles ( $m \geq 2$ ) in  $m$  separate binary relationships  $I_1, \dots, I_m$ , the range of each  $I_i$  is either  $C_i$  or  $V_i$ , and for each  $i$  (with  $1 \leq i \leq m$ ),  $\forall x (C(x) \rightarrow \exists y I_i(x, y))$  and  $\forall x, y, z (I_i(x, y) \wedge I_i(x, z) \rightarrow y = z)$ , and  $\forall x_1, x_2, y_1 \dots y_m (I_1(x_1, y_1) \wedge \dots \wedge I_m(x_1, y_m) \wedge I_1(x_2, y_1) \wedge \dots \wedge I_m(x_2, y_m) \rightarrow x_1 = x_2)$ .

Regarding ER’s weak entity types [7, 25], this is the same as CREFS, including the mandatory participation. Thus, to obtain the desired behaviour for ‘weak entity type’  $C$ , there is no need for characterising an owner, qualified association, or identifying relationship: the only thing that has changed compared to REFM is the *amount* of relationships involved, but the principle remains the same. Put differently, REFM is a special case of CREFS.

## 4. EXTENSIONS TO UML

To put the preceding analysis to use, we first address inclusion of a more precise and comprehensive way of identification in UML Class Diagrams, using ER keys and ORM’s reference schemes as a basis, and subsequently examine the feasibility of adding DL’s and OWL’s notion of ‘defined concept’ from ontologies into UML to assert that, with respect to the subject domain, a particular combination of properties fully define the class, hence, that the properties determine the class membership and vice versa. Section 4.3 contains reflections on the choices made.

### 4.1 Extending and formalising UML’s identifiers

Due to the lack of a formal specification of UML’s identifier, anyone can interpret the text in UML’s ODM [22] (pp. 290-293) to one’s liking, which we aim to do here in the light of the identification schemes in (E)ER and ORM of the previous section, and the ontological guidance from Section 2. UML ODM’s described intention is to capture all three types of (E)ER keys (simple and multi-attribute key, and weak entity type) in the one alternate key extension as depicted in Fig. 2-A. Although ER does not have mandatory participation in the keys as introduced by Chen [7], because there unidentified objects were allowed, this does not hold for UML, because it is subsumed by `NamedEntity` and uses the internal identifiers; hence, ORM’s REFM and CREFS are applicable. This entails that, first, the order of the properties of the identifier do not matter, hence `{ordered}` can be omitted. Second, we have to add that the property has a multiplicity of exactly one, which can be done by setting the `Multiplicity` option of `Property` to 1; this enforces the default value, yet it also prevents modifications, so that a common meaning of the identifier is ensured. Because of this change, which clearly does not hold for all properties, a new subclass is introduced, `IdentifyingProperty`, which must participate in at least one `Identifier`. These changes are reflected in Fig. 3.

Thus, with this extension, one can represent natural keys that are meaningful with respect to the subject domain, just like in ER/EER and ORM/ORM2, and they amount to a set of sufficient conditions for identity in the ontological sense.

What it glosses over, however, is ORM’s attribute-free approach (there are just predicates), whereas UML makes a distinction between attribute of a class and association between classes. UML’s “`Property`” can refer to association, association end, or attribute [23], hence, it is conflating iden-

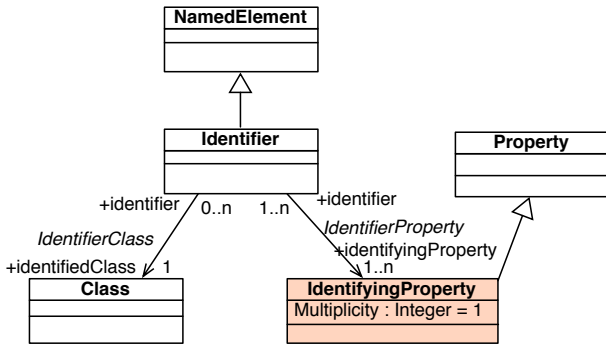


Figure 3: Extension (shaded) to the UML identifier for a single and multi-attribute key, basic extension.

tifiers ‘inside’ the class and those ‘outside’ the class; the latter are needed in a specification for a ‘weak class’ in analogy with ER’s weak entity type. In addition, the ODM restricts **Property** to association end or attribute of the class or its superclass [22] (p292) for identifiers, so that one can distinguish between an external identifier that involves at least one identifying association end and possibly one or more attributes, yet keep the option to also have only identification within the class (i.e., an identifier composed of only attributes). These refinements are depicted in Fig. 4 (note: **ExternalIdentifier**’s associations have only the multiplicity drawn to avoid cluttering the diagram, but the same adornments apply as with **IdentifierProperty**).

Thus, compared to Fig. 2, we have captured more precisely UML’s description and intention of inclusion of keys with CREFS as a basic formal foundation and Fig. 4 as extension to the ODM. Note that this can be represented easily also in the  $\mathcal{DLR}_{ifd}$  DL language (an EXPTIME-complete fragment of First Order Predicate Logic) that Berardi et al. used to formalise a large fragment of UML Class Diagrams [4], principally thanks to the **id** construct that was introduced specifically to handle complex keys, together with the inverse functional and mandatory (existential quantification) constraints. OWL and OWL 2, on the other hand, neither have the **id** construct nor  $n$ -ary relations with  $n > 2$  [20], hence, one can represent the REFM identification only.

## 4.2 Defined classes in UML

From the representation of natural keys, it is a small step to also include the notion of defined class, in analogy to defined concept in DL and OWL, which, loosely, amounts to identifying—that is, providing the mandatory and sufficient conditions—by a set of one or more properties. From an ontological viewpoint, those properties do not include attributes—philosophers are not concerned with data types in applications—but in praxis, both (in OWL terminology) data properties and object properties can be used to declare a defined class (most DL languages do not include data types, hence assume that (in DL terminology) defined concepts are defined using one or more DL role). Then, with an eye on using ontological principles to stimulate more detailed conceptual modelling by representing the subject domain semantics more precisely, one also may gain benefits for automated reasoning over formal conceptual data models.

How can one can augment UML with more aspects of

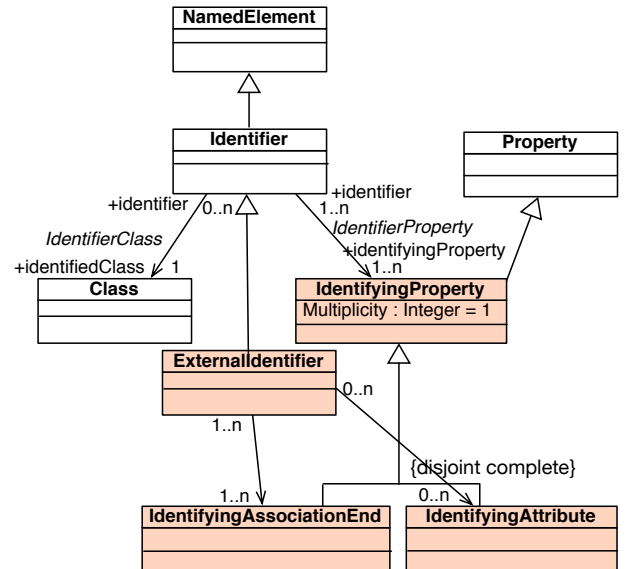
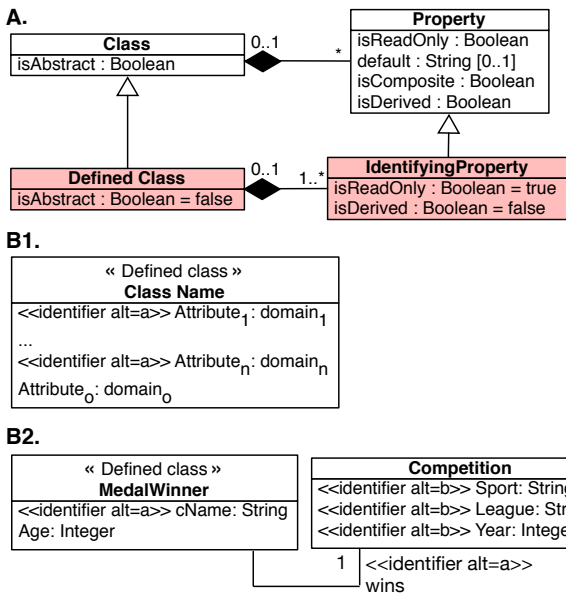


Figure 4: A more precise extension to the UML identifier (cf. Fig. 3) (only multiplicity is drawn for **ExternalIdentifier**); see text for explanation.

the notion of identity, i.e., to have a way to declare that certain properties amount to the combination of at least mandatory (and possibly necessary) and sufficient conditions in the ontological sense? For UML, we either could have a look-ahead in the spirit of the ODM [22] so that we only would have to provide a formal definition, or add it in some way to UML’s metamodel, or exploit OCL. The former is trivial from a logic perspective, provided one adopts a formal foundation of UML class diagrams (so that a metamodel would not be necessary anymore other than for human communication): defining **Defined Class** is straightforward with, depending on one’s preferred logic and notation, symbols such as “ $\leftrightarrow$ ” or “ $\equiv$ ” with their semantics, so that with the to-be-defined class on the left-hand side, the right-hand side of the definition can have any property, be it an attribute or another class. Regarding UML’s OCL, this might work, but it does not pose any restrictions on how to use it consistently and there is no conveniently usable graphical element for it.

Concerning the second option, Fig. 5-A demonstrates a possible extension to the UML metamodel by subtyping **Class** and **Property**, which is the configuration with the least amount of objections and restrictions. It limits any class stereotyped as «**Defined class**» to be composed of attributes and/or association (ends), i.e., roughly corresponding to OWL’s **DataProperty** and other properties (**ObjectProperty**). A concession for UML with respect to OWL is that **IdentifyingProperty**’s attribute **isReadOnly** (inherited from **Property**) is set to **true**, which guarantees it is not changeable. More importantly, its subclasses thus cannot override this property with some conflicting attribute declaration once defined, i.e., one only can add or refine properties. Also, **isDerived** is set to **false** so that if it were derived then one should use the original classes/attributes for identification instead. This has been added because one’s identity should not be dependent on implicit knowledge in the conceptual data model. In addition, if we would have allowed **isDerived** to



**Figure 5: A: proposed extension (shaded classes) to UML’s metamodel that introduces *Defined class* and refines *IdentifyingProperty*, see text for explanation. B: Examples of defined classes with  $n$  mandatory and sufficient conditions and one other attribute, and one in the spirit of OMG’s Olympics example [22].**

be also true, then consistency checking becomes more complicated. Last, the `isAbstract` attribute of `Class` is by default set to `false` already because abstract classes are not assumed to be instantiated directly anyway—hence do not face the issue of identity or identification in the information system—but its subclasses denoting some piece of reality or other subject domain directly, are.

Fig. 5-B demonstrates an example with `MedalWinner`’s properties that make up the identifier, and the identifier inside of `Competition` consisting of three attributes, i.e., a CREFS and a MKEY, respectively. The “`<<identifier alt=a>>`” follows through the initial proposal by [22], where the “a” is chosen string to denote which properties belong together to make up a key, as shown in Fig. 2-B. Thus, this mechanism also trivially caters for ER’s and ORM’s alternate keys.

### 4.3 Discussion

Reflecting on the gradually weakened notions of identity, identification, and keys, and the proposed extension for UML, there are four aspects to consider.

With UML’s metamodel, `IdentifyingProperty` inherits the attributes `isReadOnly` and `isDerived` from `Property` and `Defined class` inherits `isAbstract`, so they *have to* be set to some value to ensure consistency in meaning. Such constraints have not been put on OWL’s notion of defined class because there is no option to do so and, moreover, it does not make a distinction between ‘regular’ and ‘identifying’ properties. Likewise, ontologists have not considered such attributes of properties that are used for identity. From an ontological viewpoint, one may want to question the merits of the introduction of ‘identifying property’ as some kind of special property and ‘defined class’ as some special kind of universal or concept.

However, this is inverting what UML commits to now: `IdentifyingProperty` stands for ‘property used in identification’ and the constraints put on it concern *how to use it uniformly* in a conceptual data modelling language (UML Class Diagrams), but it definitely does *not* make an ontological commitment as to *the possibly different nature of* (what are) properties of identity. That is, the proposed extensions are focussed on practical usability in conceptual data modelling, informed by ontology. The proposals for conceptual data modelling with UML Class Diagrams are approximations to the qualitative, relative, and synchronic identity, and to the notion of OWL’s and DL’s defined concept described in Section 2.

Second, although UML’s ease of system-generated identifiers relieves the burden of detailed conceptual analysis by the modeller, it is exactly making implicit subject domain semantics explicit that is crucial in the analysis stage; or: less analysis during the modelling stage stores up more problems down the road in terms of software bugs and interoperability. Therefore, even if one would want to keep internal identifiers, then at least the option to describe a user-defined alternate key should be encouraged in the tools and recommended to use. As a start, this could be done with the proposed refinements incorporating multi-attribute and external keys in UML in Fig. 4. With the latter, we obtain a compromise-approximation with the elaborate SKEY, REFM, and MKEY of ER/EER and ORM, i.e., the user-defined natural keys or *semantic identifiers*. One can then expand on it with the notion of defined classes, which is already possible with the logical counterpart of the languages, and now also more precisely as proposed in Section 4.2 and illustrated in Fig. 5. The adornments of the graphical language is in line with existing practices in UML, i.e. the stereotyping with “`<<Defined class>>`”, “`<<identifier>>`”, and “`<<identifier alt=a>>`”.

Third, for both UML’s identifying properties (alternative identifiers) and ER/EER keys, it is important to have a *mandatory 1:1* relationship with the key property/ies to prevent ontologically and reality flawed use of the identifier with respect to both the weaker OBJU and to prevent occurrences of unidentified objects, which, in contrast to the Semantic Web setting, is normally not allowed in information systems.

Fourth, in contrast to (E)ER’s weak entity types, ORM’s CREFS is conceptually, logically, and diagrammatically cleaner and therefore it should be easier to understand than the terminology and icons of weak entity types, in turn, possibly simplifying modelling them as well. Following through on this line, Halpin and Morgan [17] have shown there is a correspondence between external uniqueness (here formalised as CREFS) and UML’s qualified associations. Their finer details, and possibly further refinements as to their exact meaning of either one, may be possible; this is left to future work.

To put it to good use, modelling methodologies have yet to be developed to extract good attributes and associations from the subject domain, and additional rules have to be devised for cleaning up and checking ontological classification of a taxonomy in UML Class Diagrams, alike `OntoClean` for ontologies. Also, mapping assertions between ER/EER, ORM and UML diagrams are yet to be implemented to facilitate information integration.

Having ontologically grounded, more precise, ways of representing (a weak form of) identity and identification in UML is the first step. To put it to good use, modelling methodologies have yet to be extended to handle the step of extracting good attributes and associations from the subject domain

to function as keys. Additional rules have to be devised for cleaning up and checking ontological classification of a taxonomy in UML Class Diagrams, alike OntoClean does for ontologies. Also, mapping assertions between constructs in ER, EER, ORM and UML conceptual data models are yet to be implemented to facilitate content negotiation and information integration.

## 5. CONCLUSIONS

Taking into account the philosophical notion of identity and its weaker versions of just necessary or just sufficient conditions for identity, we showed that the common ontology languages aim at weak identity with mandatory, necessary and/or sufficient conditions whereas common conceptual data modelling languages resort to identification mechanisms to represent principally only mandatory or sufficient conditions for identification with respect to the application domain. UML uses internal identifiers that do not bear any meaning, whereas ER, EER, and ORM aim at natural keys that do emphasize the properties of the entity type, although each one uses a slightly different mechanism; this was formalised unambiguously. To increase the amount of ontological foundations of UML, we have proposed two language enhancements for UML, being formally defined *simple* and *compound identifiers* and *defined classes*, which also have a corresponding extension of UML's metamodel.

### Acknowledgments.

The author thanks Diego Calvanese for fruitful discussion on a earlier draft.

## 6. REFERENCES

- [1] A. Artale, N. Guarino, and C. M. Keet. Formalising temporal constraints on part-whole relations. In G. Brewka and J. Lang, editors, *Proc. of KR'08*, pages 673–683. AAAI Press, 2008. Sydney, Australia, September 16-19, 2008.
- [2] F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logics Handbook*. Cambridge University Press, 2003.
- [3] M. Balaban and P. Shoval. Resolving the “weak status” of weak entity types in entity relationship schemas. In J. Akoka et al., editors, *Proc. of ER'99*, volume 1728 of *LNC3*, pages 363–383. Springer, 1999.
- [4] D. Berardi, D. Calvanese, and G. De Giacomo. Reasoning on UML class diagrams. *Artificial Intelligence*, 168(1-2):70–118, 2005.
- [5] D. Calvanese, G. De Giacomo, and M. Lenzerini. Identification constraints and functional dependencies in description logics. In *Proc. of IJCAI'01*, pages 155–160, 2001.
- [6] M. Carrara and P. Giaretta. Identity criteria and sortal concepts. In *Proc. of FOIS '01*, pages 234–243, New York, NY, USA, 2001. ACM.
- [7] P. P. Chen. The entity-relationship model—toward a unified view of data. *ACM Transactions on Database Systems*, 1(1):9–36, 1976.
- [8] H. Deutsch. Relative identity. In E. N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Winter 2008 edition, 2008. <http://plato.stanford.edu/archives/win2008/entries/relative-identity/>.
- [9] R. E. Grandy. Sortals. In E. N. Zalta, editor, *Stanford Encyclopedia of Philosophy*. Stanford, fall 2008 edition, 2007. <http://plato.stanford.edu/archives/fall2008/entries/sortals/>.
- [10] N. Guarino. Formal ontology and information systems. In *Proc. of FOIS'98*. Amsterdam: IOS Press, 1998.
- [11] N. Guarino and G. Guizzardi. In the defense of ontological foundations for conceptual modeling. *Scandinavian Journal of Information Systems*, 18(1):(debate forum, 9p), 2006.
- [12] N. Guarino and C. Welty. A formal ontology of properties. In R. Dieng, editor, *Proc. of EKAW'00*, LNC3. Springer Verlag, 2000.
- [13] N. Guarino and C. Welty. Identity, unity, and individuality: towards a formal toolkit for ontological analysis. In W. Horn, editor, *Proc. of ECAI'00*. IOS Press, Amsterdam, 2000.
- [14] N. Guarino and C. Welty. An overview of OntoClean. In S. Staab and R. Studer, editors, *Handbook on ontologies*, pages 151–159. Springer Verlag, 2004.
- [15] G. Guizzardi. *Ontological Foundations for Structural Conceptual Models*. Phd thesis, University of Twente, The Netherlands. Telematica Instituut Fundamental Research Series No. 15, 2005.
- [16] T. Halpin. *A logical analysis of information systems: static aspects of the data-oriented perspective*. Phd thesis, University of Queensland, Australia, 1989.
- [17] T. Halpin and T. Morgan. *Information modeling and relational databases*. Morgan Kaufmann, 2nd edition, 2008.
- [18] C. M. Keet and A. Artale. Representing and reasoning over a taxonomy of part-whole relations. *Applied Ontology*, 3(1-2):91–110, 2008.
- [19] M. C. MacLeod and E. M. Rubenstein. Universals. In *The Internet Encyclopedia of Philosophy*. 2005. <http://www.iep.utm.edu/u/universa.htm>.
- [20] B. Motik, P. F. Patel-Schneider, and B. Parsia. OWL 2 web ontology language structural specification and functional-style syntax. W3c recommendation, W3C, 27 Oct. 2009. <http://www.w3.org/TR/owl2-syntax/>.
- [21] H. Noonan. Identity. In E. N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Fall 2008 edition, 2008. <http://plato.stanford.edu/archives/fall2008/entries/identity/>.
- [22] Object Management Group. Ontology definition metamodel v1.0. Technical Report formal/2009-05-01, Object Management Group, 2009. <http://www.omg.org/spec/ODM/1.0>.
- [23] Object Management Group. Superstructure specification. Standard 2.3, Object Management Group, May 2010. <http://www.omg.org/spec/UML/2.3/>.
- [24] B. Smith. Beyond concepts, or: Ontology as reality representation. In A. Varzi and L. Vieu, editors, *Proc. of FOIS'04*, pages 73–84. Amsterdam: IOS Press, 2004.
- [25] I.-Y. Song and P. P. Chen. Entity relationship model. In L. Liu and M. T. Özsu, editors, *Encyclopedia of Database Systems*, volume 1, pages 1003–1009. Springer, 2009.
- [26] R. Wieringa and W. de Jonge. Object identifiers, keys, and surrogates—object identifiers revisited. *Theory and Practice of Object Systems*, 1(2):101–114, 1995.