# Essential, mandatory, and shared parts in conceptual data models*

Alessandro Artale and C. Maria Keet

KRDB Research Centre, Faculty of Computer Science, Free University of Bozen-Bolzano, Italy
tel: +39 0471 0161{50,27} – fax: +39 0471 016009 – {artale,keet}@inf.unibz.it

**Abstract.** This chapter focuses on formally representing life cycle semantics of part-whole relations in conceptual data models by utilizing the temporal modality. We approach this by resorting to the temporal conceptual data modeling language $\mathcal{ER}_{VT}$ and extend it with the novel notion of status relations. This enables a precise axiomatization of the constraints for essential parts and wholes compared to mandatory parts and wholes, as well as introduction of temporally suspended part-whole relations. To facilitate usage in the conceptual stage, a set of closed questions and decision diagram are proposed. The long-term objectives are to ascertain which type of shareability and which lifetime aspects are possible for part-whole relations, investigate the formal semantics for sharability, and how to model these kind of differences in conceptual data models.

## 1 INTRODUCTION

Modeling part-whole relations and aggregations has been investigated and experimented with from various perspectives and this has resulted in advances and better problem identification to a greater or lesser extent, depending on the conceptual modeling language (Artale *et al.*, 1996a; Barbier *et al.*, 2003; Bittner & Donnelly, 2005; Borgo & Masolo, 2007; Gerstl & Pribbenow, 1995; Guizzardi, 2005; Keet, 2006b; Keet & Artale, 2008; Motschnig-Pitrik & Kaasbøll, 1999; Odell, 1998; Sattler, 1995). Nowadays, part-whole relations receive great attention both in conceptual modeling community (e.g. the Unified Modeling Language, UML, the Extended Entity Relationship, EER, and the Object-Role Modeling, ORM) as well as in the semantic web community (e.g. the Description Logic based language OWL).

Several issues, such as transitivity and types of part-whole relations, are being addressed successfully with converging approaches from an ontological, logical, and/or linguistic perspectives (Borgo & Masolo, 2007; Keet & Artale, 2008; Varzi, 2004; Vieu & Aurnague, 2005). On the other hand, other topics, such as horizontal relations among parts and *life cycle semantics* of parts and wholes, remain still an open research area with alternative and complimentary approaches (Bittner & Donnelly, 2007; Guizzardi, 2005; Motschnig-Pitrik & Kaasbøll, 1999). For instance, how to model differences between an Information System for, say, a computer spare parts inventory compared to one for transplant organs? Indeed, organs are at the time before transplantation not on the shelf as are independently existing computer spare parts, but these organs are part of another whole and can only be part of another whole *sequentially*. For a university events database, one may wish to model that a seminar can be part of both a seminar

* This is a preprint version of the book chapter. The final version, in "Innovations in Information Systems modeling: Methods and Best Practices" has minor changes in the text due to changes in layout (this version is typeset with LaTeX whereas IGI Global required an MS Word document).

series and a course, *concurrently*. Another long-standing issue is how to represent essential versus mandatory parts and wholes (Artale *et al.*, 1996a). The solution proposed in Guizzardi (2005) as an extension to UML class diagrams is not easily transferable to other modelling/representation languages.

In this chapter we study representation problems related to the notion of *sharability* between parts and wholes. In particular, we are interested in representing that parts (i) cannot be shared by more than one whole; (ii) cannot exist without being part of the whole; (iii) can swap wholes in different ways. Clearly, these rich variations in shareability of parts cannot be represented in any of the common, industry-grade, UML class diagram, EER, or ORM CASE tools. In order to reach such a goal, we take a first step by aiming to answer these main questions:

- Which type of shareability and which lifetime aspects are possible?
- What is the formal semantics for sharability?
- How to model these kind of differences in a conceptual data model?

To address these questions, we merge and extend advances in representing part-whole relations as in UML class diagrams with formal conceptual data modeling for temporal databases (temporal EER) and ORM's usability features. The various shareability constraints are reworded into a set of modeling guidelines in the form of closed questions and a decision diagram to enable easy navigation to the appropriate sharability case so as to facilitate its eventual integration in generic modeling methodologies.

Concerning the formalization of the sharability notion and the relationships between the lifespans of the involved entities, we use the temporally extended Description Logic $\mathcal{DLR}_{\mathcal{US}}$ (Artale *et al.*, 2002). Indeed, while DLs have been proved useful in capturing the semantics of various conceptual data models and to provide a way to apply automatic reasoning services over them (Artale *et al.*, 2007a; Berardi *et al.*, 2005; Calvanese *et al.*, 1998b, 1999; Franconi & Ng, 2000; Keet, 2007), temporal DLs have been applied to the same extent for temporal conceptual models (Artale & Franconi, 1999; Artale *et al.*, 2003, 2002, 2007b). The formalization we present here is based on the original notion of *status relations* that captures the evolution of a relation during its life cycle. Furthermore, a set of $\mathcal{DLR}_{\mathcal{US}}$ axioms are provided and proved to be correct with respect to the semantic we provide for each particular sharability relation.

The remainder of the chapter is organised as follows. We start with some background in section 2, where we review the state of the art of representing part-whole relations in the four main conceptual modeling languages, being UML class diagrams, EER, ORM, and Description Logic languages. The problems regarding shareability of parts are summarised in section 2.4 and the basic preliminaries of the temporal $\mathcal{DLR}_{\mathcal{US}}$ and $\mathcal{ER}_{VT}$ languages are given in section 3. Our main contribution is temporalising part-whole relations to give a clear and unambiguous foundation to notions such as essential part and concurrent/sequentially shared part (section 4). Modeling guidelines will be presented in section 5 so as to transform the theory into usable material. Last, we look ahead by highlighting some future trends (section 6) and close with conclusions (section 7).

## 2 PART-WHOLE RELATIONS IN CONCEPTUAL MODELING LANGUAGES

Part-whole relations have been investigated from different starting points and with different aims. At one end of the spectrum we have philosophy with a sub-discipline called

*mereology* and its sub-variants mereotopology and mereogeometry (Borgo & Masolo, 2007; Simons, 1987; Varzi, 2004, 2006a) that focus on the nature of the *part_of* relation and its properties, such as transitivity of *part_of*, reflexivity, antisymmetry, and parts and places, or take as point of departure natural language (mereonomy). From a mathematical perspective, there is interest in the relation between set theory and mereology (e.g. Pontow & Schubert (2003) and references therein). At the other end of the spectrum we have application-oriented engineering solutions, such as aggregation functions in databases and data warehouses. Investigation into and use of representation of part-whole relations in conceptual data modeling languages lies somewhere in the middle: on the one hand, there is the need to model the application domain as accurately as possible to achieve a good quality application, yet, on the other hand, there is also the need to achieve usability and usefulness to indeed have a working information system. In this section, we take a closer look at four such conceptual modeling languages and how they fare regarding the part-whole relation: UML class diagrams (in section 2.1), EER and ORM (section 2.2), and Description Logics (section 2.3). It must be noted, however, that we primarily focus on notions such as exclusive and shareable parts among wholes and this overview is as such not a comprehensive introduction to all aspects of part-whole relations[1].

### 2.1 Aggregation in UML Class Diagrams

Part-whole relations in UML class diagrams are represented with *aggregation* associations. We first consider the UML specification of 2005 (OMG, 2005) and subsequently look at several significant extensions and formalizations that seek to address its shortcomings in particular regarding exclusiveness, sharability of parts among wholes, and thereby thus also the life cycle semantics of parts and wholes.

**UML specification.** UML (OMG, 2005) offers aggregation in two modes for UML class diagrams: composite and shared aggregation. Composite aggregation, denoted with a filled diamond on the whole-side of the association (see Figure 1), is defined as

> a strong form of aggregation that requires a part instance be included in *at most one composite at a time*. If a composite is deleted, all of its parts are normally deleted with it. Note that a part can (where allowed) be removed from a composite before the composite is deleted, and thus not be deleted as part of the composite. Compositions define transitive *asymmetric* relationships – their links form a directed, acyclic graph. (OMG, 2005) (emphases added)

The composite object is responsible for the existence and storage of the parts (OMG, 2005), which means an implicit *ontological commitment at the conceptual level*: the parts are existentially dependent on the whole (which, in turn, implies mandatoryness), and not that when a whole is destroyed its parts can exist independently. However, the "at a time" suggests that at another instant of time the same part could be part of another whole; may the part switch wholes instantaneously? In addition, the description for composite aggregation neither says that the whole is, or should be, existentially dependent on its part(s) nor that it has the part mandatorily. The difference between existential dependence and mandatory parts and wholes is refined well by Guizzardi (2005), as we will see in the next paragraph. There are three issues to take into account. First, to represent the difference between, say, a heart that must be part of a vertebrate animal but can be changed as long as the animal has a heart, whereas a brain cannot be transplanted and thus is deemed essential to the animal. Second, and a weaker constraint than essential/mandatory, is that

we can have that a part indeed must be part of a whole, but either the part $p$ (or whole $w$) can continue to exist as long as it is part of (has as part) some $w$ ($p$). Third, it is not clear if $w$ has as part $p$ *only*. More general, let $A$ be the whole with parts $B$, $C$, and $D$ in a UML class diagram as in Figure 1-A, then each part is associated to the whole through a separate binary composite aggregation, as if $A$ is a whole where its instance $a$ is made up of a collection of instances of type $B$, and/or made up of a collection of instances of type $C$ and/or $D$, making $A$ a different type of entity for each way of aggregating its parts, which cannot be the intention of the representation because that does not have a correspondence with the portion of the real world it is supposed to represent. In addition, the description of composite aggregation says it is an "asymmetric" relationship, which is in mereological theories (Varzi, 2004) always attributed to *proper parthood*[2]. Thus, what needs to be represented (at least), is that instances of $B$, $C$, and $D$ *together* make up the instance of the whole entity $A$, as in Figure 1-B, and prevent a modeler to create something like Figure 1-A. This difference is not mentioned in the UML specification, and one is left to assume it is a "semantic variation point" (OMG, 2005) which of the readings should be used. Of course, the Object Constraint Language (OCL) aids disambiguation, but is optional and one can only 'hope' for appropriate use as opposed to forcing the modeler to make such modeling decisions in the modeling stage.

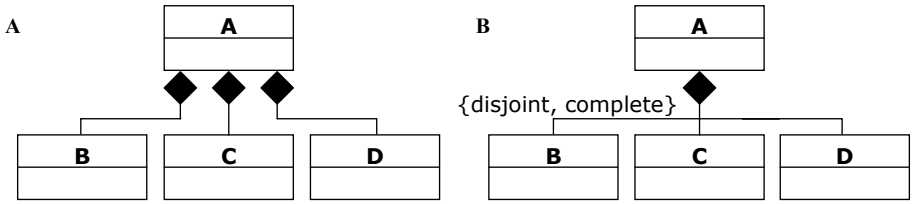


**Fig. 1.** A: Ontologically ambiguous UML composite aggregation as separate binary associations; B: the composite $A$ is composed of parts $B$, $C$, and $D$.

Shared aggregation is denoted with an open diamond on the whole-side of the aggregation association, which has it that "precise semantics ... varies by application area and modeler" (OMG, 2005), and presumably can be used for any of the types of part-whole relations and shareability described in (Barbier *et al.*, 2003; Guizzardi, 2005; Johansson, 2004; Keet, 2006b; Keet & Artale, 2008; Motschnig-Pitrik & Kaasbøll, 1999; Odell, 1998; Winston *et al.*, 1987). The main difference with composite aggregation is that shared aggregation poses no constraint on multiplicity with respect to the whole it is part of. Thus, the part may be *directly shared* by more than one whole—be it *at the same time* or *sequentially*—where those wholes are instances of either the same class or different classes. Thus, this raises at least four possibilities, but neither one can be represented in standard UML class diagrams in a way so as to be able to distinguish between them. Let $p$ be a part of type $P$ ($p \in P$) and $w_i$ stand for wholes such that $w_1, w_2 \in W$, $w_a \in W'$, $\neg(W = W')$, and $t_1, t_2$ are points in time such that $t_1 < t_2$, then one can have that

1. $p$ is part of $w_1$ at time $t_1$ and of $w_2$ at time $t_2$; e.g., a heart is transplanted from—was structural part of—one human $w_1$ into another $w_2$, or a car engine that used to be part of a FerrariCar $w_1$ used in Formula 1 racing and is put in another FerrariCar $w_2$, but the heart (car engine) cannot be part of both humans (cars) at the same time.
2. $p$ is part of $w_1$ and $w_2$ at time $t_1$; e.g., an ethics course is part of multiple BSc curricula.
3. $p$ is part of $w_1$ at time $t_1$ and of $w_a$ at time $t_2$; e.g., phosphorylation (a phosphor atom part $p$ is exchanged between two molecules $w_1$ and $w_a$ of different type) or South Tyrol

used to be part of (located in) the Austro-Hungarian Empire $w_1$ and is now part of the Republic of Italy $w_a$.

4. $p$ is part of $w_1$ and $w_a$ at time $t_1$; e.g., a seminar is part of both the Language Colloquia and of the Knowledge Representation course, or a cello player is member of a chamber ensemble $w_1$ and member of the Royal Philharmonic Orchestra $w_a$.

The examples for the four cases are fairly straightforward for transplant databases, car mechanics information systems, university databases, geographic information systems, and employment information systems, yet cannot be represented unambiguously with UML—nor with commonly used other conceptual data modeling languages, as we will see in the next sections—other than using shared aggregation for all of them.

A range of scenarios of life cycles of participating objects are possible, each with different behavior in the system, which thus ought to have their implementation-independent counterpart in the conceptual model. However, overall, the ambiguous specification and modeling freedom in UML does not enable making implicit semantics explicit in the conceptual model, and rather fosters creation of unintended models. This has been observed by several researchers, who have proposed a range of extensions to UML class diagrams.

**Formalizations of aggregation in UML class diagrams.** UML does not have a formal semantics, which demands from the researchers who propose extensions to also give the formal semantics. A near-complete formalization is proposed by Berardi *et al.* (2005), who developed a First Order Logic (FOL) as well as Description Logic (DL) encoding of UML with $\mathcal{DLR}_{ifd}$; that is, for each UML model, there is an equi-satisfiable $\mathcal{DLR}_{ifd}$ knowledge base. However, UML's aggregation has not been addressed other than 'shared aggregation' (no formalisation is provided to account for additional constraints and composite aggregation). In Berardi et al's (2005) formalisation of shared aggregation in UML class diagrams, we have $\forall x, y(G(x, y) \rightarrow C_1(x) \wedge C_2(y))$, where $G$ is a binary predicate for the aggregation and $C_1$ and $C_2$ are concepts. That is, a straightforward binary relation as if it were a mere UML association relation. The avoidance to map UML's intuitive part-whole relation is partially due to the ambiguous semantics of aggregation in UML and partially because adequately representing parthood in DL has its own issues (see below). In contradistinction, others have been more precise on the aggregation, but they omitted a formalization of UML class diagram language. For instance, Barbier *et al.* (2003) represent several constraints on the part-whole relation using OCL, which is, however, not immediately transferrable to other conceptual modeling languages. They formulate the various add-ons as a `context` in OCL and add a meta-model fragment for whole-part relations where the attribute `aggregation` is removed from the `AssociationEnd` meta-class, `Whole` added as a subclass of `Relationship` and has two disjoint subclasses `Aggregation` and `Composite`. Motschnig-Pitrik & Kaasbøll (1999) and Guizzardi (2005) present a First Order Logic formalization and corresponding adornments for the graphical notation in UML class diagrams using new icons and labels, which are effectively *in addition* to the axiomatizations of mereological theories and perceived to be necessary for conceptual modeling of mereological and meronymic relations. Guizzardi adds, among others, the notion of essential part $EP$, which he defines as (Guizzardi (2005): p165):

> **Definition 5.11 (essential part)**: An individual $x$ is an essential part of another individual $y$ iff, $y$ is existentially dependent on $x$ and $x$ is, necessarily, a part of $y$: $EP(x, y) =_{def} ed(y, x) \wedge \Box(x \leq y)$. This is equivalent to stating that $EP(x, y) =_{def} \Box(\epsilon(y) \rightarrow \epsilon(x)) \wedge \Box(x \leq y)$, which is, in turn, equivalent to $EP(x, y) =_{def} \Box(\epsilon(y) \rightarrow \epsilon(x) \wedge (x \leq y))$. We adopt here the mereological continuism defended by (Simons,

1987), which states that the part-whole relation should only be considered to hold among existents, i.e., $\forall x, y(x \leq y) \rightarrow \epsilon(x) \wedge \epsilon(y)$. As a consequence, we can have this definition in its final simplification

(47). $EP(x, y) =_{def} \Box(\epsilon(y) \rightarrow (x \leq y))$

where $\epsilon$ denotes existence, $\leq$ a partial order, and $\Box$ necessity. The weaker version is mandatory parthood $MP$, which is defined as (Guizzardi (2005): p167):

**Definition 5.13 (mandatory part)**: An individual $x$ is a mandatory part of another individual $y$ iff, $y$ is generically dependent of an universal $U$ that $x$ instantiates, and $y$ has, necessarily, as a part an instance of $U$:

(49). $MP(U, y) =_{def} \Box(\epsilon(y) \rightarrow (\exists U, x)(x < y))$.

Observe that in this setting, essential parts are also *immutable*—"stability in identity and number" (Barbier *et al.*, 2003) of the part—and *inseparable* ("$IP(x, y) =_{def} \Box(\epsilon(x) \rightarrow (x \leq y))$" in (Guizzardi, 2005)). There are finer-grained details between essential parts (or wholes) and immutable parts (wholes) that are caused by the kind of classes that participate in the part-whole relation (Guizzardi, 2005, 2007; Artale *et al.*, 2008); that is, the former has participating classes that are rigid, whereas for immutable parts (wholes) the class is not rigid (indeed, anti-rigid). The notion of a class' metaproperty cencerning rigidity—i.e., being rigid, non-rigid, semi-rigid, or anti-rigid—is important for designing good subtype hierarchies (Guarino & Welty, 2000) and its use with UML and ORM2 is actively being investigated regarding how to incorporate it and to what extent (Guizzardi, 2005; Halpin, 2007).

An example of adorning UML class diagrams is depicted in Figure 2 that demonstrates the proposed representation for the *sub_quantity_of* relation with an additional symbol, OCL constraint, and stereotypes. Motschnig-Pitrik and Kaasbøll, on the other hand,

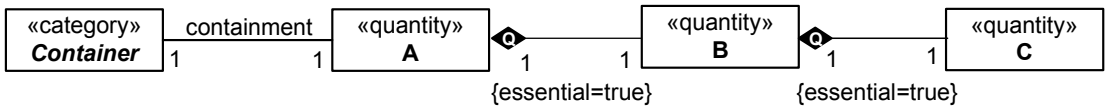


**Fig. 2.** Part-whole relations among quantities. Essential parts are indicated with `essential = true`, which implies a composite aggregation (filled diamond), which is of the type "Q" for quantities. The stereotypes ("≪≫") add further constraints to the permitted types of classes. (Source: Guizzardi (2005))

focus on *gradations* of exclusiveness between part and whole. This corresponds partially to Guizzardi's mandatoryness and (in)separability of the part from the whole, as can be observed from one of the definitions, such as total exclusiveness (Motschnig-Pitrik & Kaasbøll (1999): p785):

**Total exclusiveness**. A part-of reference is *totally exclusive* if there exists exactly one immediate part-of link from a part-type $P$ to a whole-type $W$ and, for each instance $p$ of $P$, there exists at most one instance $w$ of $W$ such that $p$ part-of $w$. More formally, let:

$p_k$ instance-of $P, k \in [1..n]$,
$w_i$ instance-of $W, i \in [1..n']$, $w_{i'}$ instance-of $W, i' \in [1..n]$,
$wx_j$ instance-of $WX, j \in [1..n'']$
then
$P$ totally-exclusive part-of $W \Leftrightarrow \forall WX$

$((P \text{ part-of } W \wedge P \text{ part-of } WX) \Rightarrow (W = WX \vee W \text{ part-of } WX)) \wedge$
$((p_k \text{ part-of } w_i \wedge p_k \text{ part-of } w_{i'}) \Rightarrow (i = i')) \wedge$
$((p_k \text{ part-of } w_i \wedge p_k \text{ part-of } wx_j) \Rightarrow (w_i = wx_j \vee w_i \text{ part-of } wx_j))$

One can add further gradations in *sharability*. Motschnig-Pitrik & Kaasbøll (1999) distinguish between "*degree of sharing* of parts among whole objects" and "*degree of dependence* between some part object and some whole object(s)" where the former acts out as static constraints and the latter concerns the life-cycle of objects. To summarize and comment on Motschnig-Pitrik & Kaasbøll (1999), there are six cases that each get their own modeling construct in a UML class diagram.

- Total exclusiveness: there exists exactly one immediate part-of relation from $P$ to $W$ (thus $P$ *cannot* have another part-of relation to a $W'$), there is *at most one* instance $w$ s.t. $p$ part-of $w$. Thus, with this constraint, the $w$ can also exist without having as part $p$, hence neither essential nor mandatory participation from that side.
- Arbitrary sharing: "A part-of link from $P$ to $W$ is shared if there may exist further shared or intraclass exclusive (see below) links from $P$ to whole-types $WX$, $WY$, etc., and if, for each instance $p$ of $P$, there may exist more than one instance of $W : w_1, w_2, ...$ such that $p$ part-of $w_1$, $p$ part-of $w_2$, etc.". Thus, with this combination where $p$ can be part of $w_i \in W^j$ where $0 \leq i \leq n$ and $1 \leq j \leq n$; i.e., $p$ can be part of zero or more $w$s that are instances of one or more $W$s, and thereby subsumes the next four options.
- Interclass exclusiveness: there exists exactly one type-level part-of relation from $P$ to $W$ and for each $p_1, ...p_m \in P$ there may exist $w_1, ...w_n \in W$ such that we have $p_1$ part-of $w_k$, ... , $p_1$ part-of $w_n$, ... $p_n$ part-of $w_1$, ... , $p_h$ part-of $w_l$ (with $h \leq m$, $k \leq n$, and $l \leq n$). Or, simply a 0:n relation between $p_i$ and $w_i$, where $w_i \in W$ and $0 \leq i \leq n$.
- Intraclass exclusiveness: as for interclass, but then $\geq 1$ part-of relations to $\geq 1$ different types of wholes.
- Selective exclusiveness: $\geq 1$ part-of relations to different types of wholes, but only one of them may be instantiated. It is unclear if this means *at a time* or possibly *ever* during the life time of the part (an XOR constraint at the type level).
- Selectively intraclass exclusive: as for selective exclusiveness, but then also that all part-of relations have a max cardinality of 1.

In addition, the dependence/independence axis concerns "A *dependent* part-of relationship between a part-type $P$ and a whole-type $W$ is one in which the existence of each part-object $p_i$ of type $P$ depends on the existence of one and the same whole-object $w_i$ of type $W$ throughout the lifetime of the part-object", which is also called "lifetime-dependence", or in Guizzardi's terminology *essential* whole to the part, but which implies only a minimum cardinality of one on the $W$-side by Motschnig-Pitrik & Kaasbøll (1999).

Despite the problems with the UML class diagram specification as well as the limited extensions, the issues have been investigated to a greater extent than within other conceptual modeling languages. The next two sections focus on EER, ORM and DL languages.

### 2.2 Part-Whole Relations in (E)ER and ORM

It may be clear from the previous section that part-whole relations in UML class diagrams can have poorly defined semantics, but what about other conceptual modeling languages? Entity-Relationship (ER) does not have a separate constructor for the part-whole relations, despite the occasional (Shanks *et al.*, 2004) request. Neither does Object-Role Modelling (ORM) have a separate constructor for parthood relation. Are they better off than UML? What, if any, can already be represented from part-whole relations with ER or ORM? Here, we summarise ORM's difference with UML based on (Keet, 2006b).

Recollect that the UML specification inserts design and implementation considerations for composite aggregation, so that a part is *existentially dependent* on the whole, and not that when the whole is destroyed, the parts, explicitly, can have their own life or, explicitly, become part of another whole. Here there is a difference between UML and

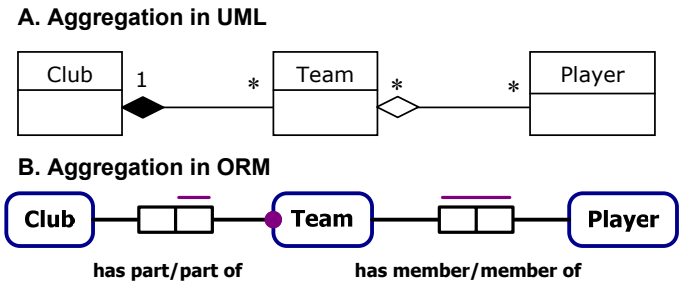


**Fig. 3.** Graphical representation of "aggregation" in UML and ORM. (Source: adapted from Halpin (1999) with part-whole relations as proposed in Keet (2006b))

ORM intended semantics: with composite aggregation in the UML specification, part $p$ *cannot* exist without that whole $w$, but ORM semantics of the suggested mapping (Halpin, 2001) says that 'if there is a relation between part $p$ and whole $w$, then $p$ must participate exactly once'. Put differently, $p$ may indeed become part of some other whole $w'$ after $w$ ceases to exist as a whole, as long as there is *some* whole it is part of, but not necessarily *the same* whole. Hence, in contrast with UML, in ORM there is no implicit existential dependency of the part on the whole (see also Figure 3-B).

Compared to more and less comprehensive formalizations and extensions for aggregation in UML (Barbier *et al.*, 2003; Guizzardi, 2005; Motschnig-Pitrik & Kaasbøll, 1999; Berardi *et al.*, 2005), for ORM, richer representations of the semantics are possible already even without dressing up the ORM diagram with icons and labels. For instance, Motschnig-Pitrik & Kaasbøll (1999)'s new "selectively intraclass exclusive" constraint is an XOR constraint (dotted circle with cross) over the Part-roles of two or more fact types to different types of Whole in an ORM diagram. Suggestions to model several aspects of the part-whole without extending ORM were presented in (Keet, 2006b), which also includes several guidelines to ease selecting the appropriate part-whole relation and its mandatory and uniqueness constraints.

## 2.3 Description Logics

Description Logic (DL) languages are more often used as knowledge representation languages than as conceptual data modeling languages and if they are used for conceptual modeling, they are used in the background hidden from the modelers and domain experts because the formalisms are deemed not easily accessible. Thus far, the combination of DL languages and conceptual data modeling languages is primarily limited to the well-studied $\mathcal{DLR}$ family of DL languages (Artale *et al.*, 2006, 2007a; Berardi *et al.*, 2005; Calvanese *et al.*, 1998b, 1999; Calvanese & De Giacomo, 2003; Fillottrani *et al.*, 2006; Franconi & Ng, 2000; Keet, 2007), which provide not only a formal foundation for the mostly graphically-oriented conceptual data modeling languages, but also offer prospects of automated reasoning over conceptual data models to derive implicit relations, constraints, and inconsistencies and thereby contribute to better quality conceptual data

models. Therefore, efforts in representing parthood in DL languages will be briefly summarised.

Research on part-whole relations for DLs date back to the early '90s, but thus far none of the DL languages that are being investigated and implemented (DL-Lite, $\mathcal{DLR}$, OWL, and $\mathcal{EL}$ families) have the part-whole relation as a first-class citizen. However, as neither UML nor ER nor ORM, implement part-whole relations properly, it might not matter that most DL languages do not have a comprehensive treatment of part-whole relations, at present. It is, however, being investigated.

We first address two early attempts. Artale *et al.* (1996a) experimented with adding a *has_part* relation as $\succeq$ with the transitive closure of a parthood relation (1). One can define, e.g., `Car` as having wheels that in turn have tires (2), such that it follows that cars have as part tires ($\texttt{Car} \sqsubseteq \exists \succeq.\texttt{Tire}$).

$$\succeq \doteq (\texttt{primitive-part})* \tag{1}$$

$$\texttt{Car} \doteq \exists \succeq .(\texttt{Wheel} \sqcap \exists \succeq .\texttt{Tire}) \tag{2}$$

However, adding transitive closure makes languages of even low expressivity, such as $\mathcal{ALC}$, already ExpTime-complete. Alternatively, one can define direct parthood $\prec_d$ (Sattler, 1995), but this should verify the immediate inferior, which makes the language undecidable (Artale *et al.*, 1996b), which is even less desirable for operational information systems. Schulz *et al.* (2000) have developed an elaborate workaround (with $\mathcal{ALC}$) so as to be able to simulate transitivity of parthood relations by remodeling the part-of relation as *is_a* hierarchies using so-called SEP triplets. The three core items are the **S**tructure-concept node that subsumes one (anatomical) entity, called **E**-node, and the parts of that entity (the **P**-node). An *is_a* hierarchy is then built up by relating the P-node of a whole concept $D$ to the S-node of the part $C$, where in turn the P-node of $C$ is linked to the S-node of $C$'s part. More formally, the definition of the whole $D$ is (3), by which one can derive its anatomical proper part (*a-pp*) $C$ as (4). Obviously, if this were to be used, this would require an intuitive user interface.

$$D_P \doteq D_S \sqcap \neg D_E \sqcap \exists \textit{a-pp}.D_E \tag{3}$$

$$C_E \sqsubseteq \exists \textit{a-pp}.D_E \tag{4}$$

Around the same time, Sattler (2000) showed that with some extensions to $\mathcal{ALC}$, it is possible to include more aspects of the parthood relation. These are: transitive roles (that is, permit $R_+ \subseteq R$), inverse roles to have both part-of and has-part, role hierarchies to include subtypes of the parthood relation, and number restrictions to model the amount of parts that go in the whole. This brings us to the language called $\mathcal{SHIQ}$, which is a predecessor of the OWL-DL ontology language. In fact, the base language for the even more expressive OWL 2, $\mathcal{SROIQ}$, has constructors for all but one of the relational properties: antisymmetry, required for mereological part-of (Varzi, 2004), is not possible yet (Horrocks *et al.*, 2006).

The latest—and most comprehensive—attempt to represent parthood relations in a DL language is put forward by Bittner & Donnelly (2005), who approach the problem starting from a FOL characterisation and subsequently limit its comprehensiveness and complexity to fit it into a DL language, although it is unclear if their $\mathcal{L}^{\sim Id \sqcup}$ is decidable. In their theory, called DL-PCC, several constraints and definitions cannot be represented. These are: impossibility to state that *component_of* (`CP`), *proper_part_of* (`PP`) and *contained_in* (`CT`) are irreflexive and asymmetric, and it is missing a discreteness axiom for `CP` or `CT` or

a density axiom for PP (see Bittner & Donnelly (2005) for details and discussion). They include transitivity of the characterised parthood relations, but thereby do not have the option to state also that, e.g., a *directly_contained_in* relation is *in*transitive (the same problem as mentioned above for $\mathcal{ALC}$).

Artale *et al.* (1996a,b) have placed the requirement for adequately representing the part-whole relation in a wider context, where some outstanding issues of 12 years ago are still in need of a solution. For instance, (non)distributivity of part-whole relations[3], 'horizontal' relations between the parts, and disjoint covering over the parts. The latter is an issue with DL but not for database models, because DL languages adhere to the open world assumption whereas databases do not. For instance, if we have in a DL language a type-level (TBox-) statement where C has two parts D and E:

$\texttt{C} \sqsubseteq \exists\ \texttt{has\_part.D} \sqcap \exists\ \texttt{has\_part.E}$

then it may be that instances of C have more parts than only instances of D and E because the composite C is not fully defined. No DL language deals with an additional axiom that states that C is *composed* of—the mereological sum of—D and E *only*; what we can state is that C is *defined* by having D and E as parts ($\texttt{C} \doteq \exists\ \texttt{has\_part.D} \sqcap \exists\ \texttt{has\_part.E}$). In contradistinction, conceptual models and databases do adhere to a closed world assumption, thereby making instances of C uniquely composed of at least one instance of D and at least one instance of E. The status of differentiating between e.g. essential and mandatory part (see section 2.1) is unclear, unless we use a temporal DL such as $\mathcal{DLR}_{\mathcal{US}}$ (see below).

### 2.4 Problems and Requirements for Modeling Shared and Composite Parts and Wholes

Summarizing the problems for adequately modeling the different ways that parts can be part of a whole in the main commonly used conceptual modeling languages—with or without extensions—, we have (1) the absence of adequately distinguishing between mandatory and essential parts and wholes and, vice versa, existence of $p$ and/or $w$ independently for some time (e.g., the relation is temporarily "suspended" or $p$ is "scheduled" to become part of $w$), (2) lack of clarity how to represent that part $p$ that can be part of more than one whole s.t. either $w_i \in W$ or ($w_j \in W$ and $w_k \in W'$) and (3) if these shared parts can be shared concurrently and/or sequentially among the wholes $p$ being part of $w_i$ and $w_j$. These issues can be reformulated in a set of requirements for conceptual data modeling languages if they want to be expressive enough to enable full shareability semantics of parts and wholes.

**Requirements for modeling shareability of parts.** Based on the literature review, we can formulate the following requirements for modeling shareability and, implicitly, life cycle semantics in conceptual modeling languages[4].

1. Arbitrarily shareable with no particular constraints;
2. Existentially dependent/essential part or whole (mutually, or not), and (im)possibility of independent existence of the part from the whole;
3. Being able to differentiate between mandatory and essential parts and wholes;
4. Change in whole; that is, a part $p$ was part of $w_1$ (where $w_1 \in W$) and can become part of another whole $w_2$, be it that there is a (negligible) time that $p$ exists independently before (after) being part of and be it that $w_2 \in W$ or $w_2 \in W'$;
5. Change in part; that is, the inverse of requirement nr.4 where the whole loses and gains a part during its lifetime;

6. Disjoint covering of parts being part of a whole.

Notions such as essential parts, change, and before and after indicate *temporality* of either the part-whole relation, or the participating parts and wholes, or both; hence, the more general requirement for *temporal conceptual data modeling* in order to address the above-mentioned requirements. For the current restricted scope, we may not need a full-fledged temporal knowledge representation language to adequately address the shareability of part and wholes. Before going into those details in section 3, we summarize the literature dealing with temporal parts.

**Temporalizing part-whole relations.** The most straightforward, yet also limited, way to temporalize part-whole relations is to turn a part-of predicate from a binary into a ternary relation, such that we have $p$ part of $w$ at time $t$: $part\_of(p, w, t)$. To the best of our knowledge, almost all extant temporalizations of parthood take this approach (Bittner & Donnelly, 2007; Masolo *et al.*, 2003; Smith *et al.*, 2005) but do not go further to take advantage of a temporal knowledge representation language[5]. An exception is Barbier *et al.* (2003), who created an oclUndefined observer function to "assert that all parts of result do not exist before (@pre) the execution" of the creation of the whole instance $w$, which is intended for representing life time dependencies in UML class diagrams. They also tried representation of immutability, but this remained an open problem due to the lack of a full-fledged implementation of a temporal UML. Furthermore, Barbier *et al.* (2003); Opdahl *et al.* (2006) listed nine principle life cycle cases. We extend this here to 18 cases (see Figure 4) mainly since they represent two distinct perspectives: (i) fixed the lifespan of a whole, we are interested in characterizing the lifespan of its part (Figure 4-A) and, vice versa, (ii) fixed the lifespan of the part, we are interested in the temporal relations with the lifespan of its whole (Figure 4-B). We shall see in section 4.3 that these two views require distinct constraints, too. A curious feature of UML is the Boolean readOnly metaproperty that was initially proposed for attributes as a Changeability sort with constant symbols "frozen" and "changeable" (Álvarez & Alemán, 2000). OMG (2005) now constrains it such that "[i]f a navigable property is marked as readOnly, then it cannot be updated once it has been assigned an initial value" (OMG, 2005), where a property is "structural feature", such as attribute and association end but it is also suggested for representing rigid classes (Halpin, 2007). Unfortunately, its "semantics is undefined" (pp 241, 249, 251, 254, 280) for various cases, such as what should happen with a readOnly association end when a link is destroyed. Intuitively, it captures a property that holds globally during its entire existence, hence, is a candidate to use for representing at least some aspects of lifecycle semantics. Albert *et al.* (2003) provide an additional interpretation where objects instantiating readOnly classes can only participate in links created during creation of the object, but no links to that object can be added afterward, and subsequently use this for constraining the part-class in a composite aggregation. This seems too restrictive, however, because if we assume Brain to be such a composite part of Patient, then it would surely be possible in a hospital information system that brain $o_1$ of patient $o_2$ may have to be linked to, say, some instance $o_3 \in$ BrainScan and at a later point in time participate in a new aggregation association to $o_4 \in$ BrainTumor. The prospects for usage of readOnly may be more interesting if it is (1) applied either to the composite aggregation association or to the association ends because for the part-whole relation one has to consider also the temporal behavior of the relation and (2) its semantics would be defined precisely and have an effect in the modeling as opposed to in the software code only.

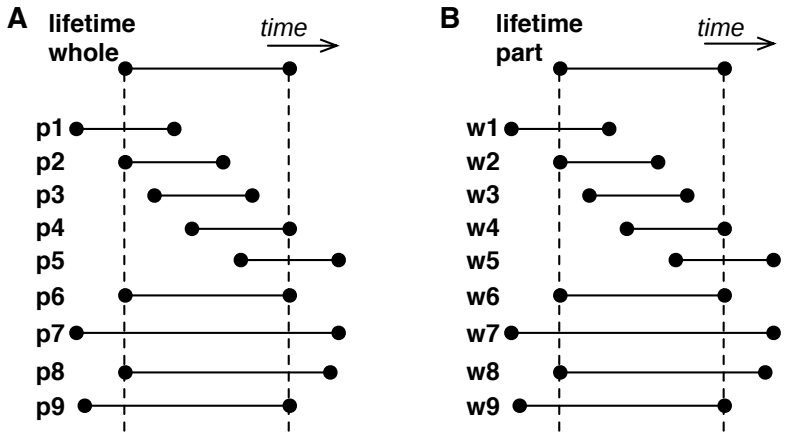


**Fig. 4.** Possible lifespans of the part with respect to the whole it is part of (A) and similarly for the whole's lifespan w.r.t its part (B).

Regarding the ternary temporal part-whole relation, we have, for instance, Bittner and Donnelly's "temporal mereology" (2007), which was developed to deal with "portions of stuff", i.e., how to deal with subquantities (portions) of amounts of matter, such as gold, and mixtures, such as lemonade, in time (see also Figure 9 for types of part-whole relations). Limiting oneself to only ternary part-whole relations runs into rather complicated formalizations, whereas well-defined temporal logics—and those applied to temporal conceptual data modeling in particular—can hide at least some of the details, which enhances understandability and (re)usability of a conceptual model by modeler and domain expert alike. In order to arrive at this point, we will introduce such a formal temporal conceptual modeling language, which enables one to model essential and shareable parts in a precise and clear way. Moreover, in addition to the modeling enhancements for the conceptual models themselves (in section 4), we will add modeling guidelines to facilitate easy navigation and choosing the appropriate part-whole relation and object types (section 5).

## 3 TEMPORAL DATA MODELS

In order to capture the range of possibilities of sharebility, composite, and essential parts and represent them in a convenient way in a conceptual modeling language, we introduce here representation languages able to capture time varying information. As outlined in the previous section, we need temporal constructs in a conceptual modeling language. Temporal EER have been developed (e.g., Artale *et al.* (2003, 2006, 2007b) and references therein) and a subset has been implemented in MADS (Parent *et al.*, 2006), but one may contend that, ideally, one should have a way to have the approach general enough so as to be transferrable to ORM and UML class diagrams, too. Artale et al's temporal EER, called $\mathcal{ER}_{VT}$, has a correspondence with the temporal Description Logic $\mathcal{DLR}_{\mathcal{US}}$ (Artale *et al.*, 2002), which, in turn, gives a model-theoretic semantics to $\mathcal{ER}_{VT}$—any corresponding icon in the graphical diagrams can then be considered 'syntactic sugar' with a precise meaning. In addition, with an UML/EER/ORM to DL transformation we can then provide the sought-after genericity of the approach[6].

$$
\begin{aligned}
C &\rightarrow \top \mid \bot \mid CN \mid \neg C \mid C_1 \sqcap C_2 \mid \exists^{\leqslant k}[U_j]R \mid \\
&\quad \Diamond^+ C \mid \Diamond^- C \mid \Box^+ C \mid \Box^- C \mid \bigoplus C \mid \bigominus C \mid C_1 \,\mathcal{U}\, C_2 \mid C_1 \,\mathcal{S}\, C_2 \\
R &\rightarrow \top_n \mid RN \mid \neg R \mid R_1 \sqcap R_2 \mid U_i/n : C \mid \\
&\quad \Diamond^+ R \mid \Diamond^- R \mid \Box^+ R \mid \Box^- R \mid \bigoplus R \mid \bigominus R \mid R_1 \,\mathcal{U}\, R_2 \mid R_1 \,\mathcal{S}\, R_2
\end{aligned}
$$

$$
\begin{aligned}
\top^{\mathcal{I}(t)} &= \Delta^{\mathcal{I}} \\
\bot^{\mathcal{I}(t)} &= \emptyset \\
CN^{\mathcal{I}(t)} &\subseteq \top^{\mathcal{I}(t)} \\
(\neg C)^{\mathcal{I}(t)} &= \top^{\mathcal{I}(t)} \setminus C^{\mathcal{I}(t)} \\
(C_1 \sqcap C_2)^{\mathcal{I}(t)} &= C_1^{\mathcal{I}(t)} \cap C_2^{\mathcal{I}(t)} \\
(\exists^{\leqslant k}[U_j]R)^{\mathcal{I}(t)} &= \{\, d \in \top^{\mathcal{I}(t)} \mid \sharp\{\langle d_1, \ldots, d_n\rangle \in R^{\mathcal{I}(t)} \mid d_j = d\} \leqslant k\} \\
(C_1 \,\mathcal{U}\, C_2)^{\mathcal{I}(t)} &= \{\, d \in \top^{\mathcal{I}(t)} \mid \exists v > t.(d \in C_2^{\mathcal{I}(v)} \wedge \forall w \in (t, v).d \in C_1^{\mathcal{I}(w)})\} \\
(C_1 \,\mathcal{S}\, C_2)^{\mathcal{I}(t)} &= \{\, d \in \top^{\mathcal{I}(t)} \mid \exists v < t.(d \in C_2^{\mathcal{I}(v)} \wedge \forall w \in (v, t).d \in C_1^{\mathcal{I}(w)})\} \\
(\top_n)^{\mathcal{I}(t)} &\subseteq (\Delta^{\mathcal{I}})^n \\
RN^{\mathcal{I}(t)} &\subseteq (\top_n)^{\mathcal{I}(t)} \\
(\neg R)^{\mathcal{I}(t)} &= (\top_n)^{\mathcal{I}(t)} \setminus R^{\mathcal{I}(t)} \\
(R_1 \sqcap R_2)^{\mathcal{I}(t)} &= R_1^{\mathcal{I}(t)} \cap R_2^{\mathcal{I}(t)} \\
(U_i/n : C)^{\mathcal{I}(t)} &= \{\, \langle d_1, \ldots, d_n\rangle \in (\top_n)^{\mathcal{I}(t)} \mid d_i \in C^{\mathcal{I}(t)}\} \\
(R_1 \,\mathcal{U}\, R_2)^{\mathcal{I}(t)} &= \{\, \langle d_1, \ldots, d_n\rangle \in (\top_n)^{\mathcal{I}(t)} \mid \\
&\qquad \exists v > t.(\langle d_1, \ldots, d_n\rangle \in R_2^{\mathcal{I}(v)} \wedge \forall w \in (t, v).\, \langle d_1, \ldots, d_n\rangle \in R_1^{\mathcal{I}(w)})\} \\
(R_1 \,\mathcal{S}\, R_2)^{\mathcal{I}(t)} &= \{\, \langle d_1, \ldots, d_n\rangle \in (\top_n)^{\mathcal{I}(t)} \mid \\
&\qquad \exists v < t.(\langle d_1, \ldots, d_n\rangle \in R_2^{\mathcal{I}(v)} \wedge \forall w \in (v, t).\, \langle d_1, \ldots, d_n\rangle \in R_1^{\mathcal{I}(w)})\} \\
(\Diamond^+ R)^{\mathcal{I}(t)} &= \{\langle d_1, \ldots, d_n\rangle \in (\top_n)^{\mathcal{I}(t)} \mid \exists v > t.\, \langle d_1, \ldots, d_n\rangle \in R^{\mathcal{I}(v)}\} \\
(\bigoplus R)^{\mathcal{I}(t)} &= \{\langle d_1, \ldots, d_n\rangle \in (\top_n)^{\mathcal{I}(t)} \mid \langle d_1, \ldots, d_n\rangle \in R^{\mathcal{I}(t+1)}\} \\
(\Diamond^- R)^{\mathcal{I}(t)} &= \{\langle d_1, \ldots, d_n\rangle \in (\top_n)^{\mathcal{I}(t)} \mid \exists v < t.\, \langle d_1, \ldots, d_n\rangle \in R^{\mathcal{I}(v)}\} \\
(\bigominus R)^{\mathcal{I}(t)} &= \{\langle d_1, \ldots, d_n\rangle \in (\top_n)^{\mathcal{I}(t)} \mid \langle d_1, \ldots, d_n\rangle \in R^{\mathcal{I}(t-1)}\}
\end{aligned}
$$

**Fig. 5.** Syntax and semantics of $\mathcal{DLR}_{\mathcal{US}}$.

This section presents the formal background in modeling temporal varying information. Such a formalization will be used in the following sections when the basic conceptual data model is extended to capture particular properties of part-whole relations.

### 3.1 The Temporal Description Logic $\mathcal{DLR}_{\mathcal{US}}$

The temporal description logic $\mathcal{DLR}_{\mathcal{US}}$ (Artale *et al.*, 2002) combines the propositional temporal logic with the *Since* and *Until* operators and the (non-temporal) description logic $\mathcal{DLR}$ (Calvanese *et al.*, 1998a; Baader *et al.*, 2003) that serves as common foundational language for various conceptual data modeling languages (Calvanese *et al.*, 1998b, 1999). $\mathcal{DLR}_{\mathcal{US}}$ can be regarded as an expressive fragment of the first-order temporal logic $L^{\{\textbf{since, until}\}}$ (Chomicki & Toman, 1998; Hodgkinson *et al.*, 2000).

The basic syntactical types of $\mathcal{DLR}_{\mathcal{US}}$ are *classes* (also known as *entity types* or *object types*) and $n$-ary *relations* (*associations*) of arity $\geq 2$. Starting from a set of *atomic classes* (denoted by $CN$), a set of *atomic relations* (denoted by $RN$), and a set of *role symbols* (denoted by $U$, comparable to an ORM-role or component of a UML association) we can define inductively (complex) class and relation expressions (see upper part of Figure 5), where the binary constructors $(\sqcap, \sqcup, \mathcal{U}, \mathcal{S})$ are applied to relations of the same arity, $i$, $j$, $k$, $n$ are natural numbers, $i \leq n$, and $j$ does not exceed the arity of $R$. Observe that for both class and relation expressions all the Boolean constructors are available. The selection expression $U_i/n : C$ denotes an $n$-ary relation whose $i$-th argument ($i \leq n$), named $U_i$, is of type $C$. (In ORM terminology, $U_i/n : C$ refers to the role $U_i$ played by $C$ in the fact type.) If it is clear from the context, we omit $n$ and simply write $(U_i : C)$. The projection expression $\exists^{\leqslant k}[U_j]R$ is a generalisation with cardinalities of the projection

operator over argument $U_j$ of relation $R$; the plain classical projection is $\exists^{\geq 1}[U_j]R$. It is also possible to use the pure argument position version of the language by replacing role symbols $U_i$ with their corresponding position numbers $i$.

The model-theoretic semantics of $\mathcal{DLR}_{\mathcal{US}}$ assumes a flow of time $\mathcal{T} = \langle \mathcal{T}_p, < \rangle$, where $\mathcal{T}_p$ is a set of time points (also called chronons) and $<$ a binary precedence relation on $\mathcal{T}_p$, which is assumed to be isomorphic to $\langle \mathbb{Z}, < \rangle$. The language of $\mathcal{DLR}_{\mathcal{US}}$ is interpreted in *temporal models* over $\mathcal{T}$, which are triples of the form $\mathcal{I} \doteq \langle \mathcal{T}, \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}(t)} \rangle$, where $\Delta^{\mathcal{I}}$ is non-empty set of objects (the *domain* of $\mathcal{I}$) and $\cdot^{\mathcal{I}(t)}$ an *interpretation function* such that, for every $t \in \mathcal{T}$ ($t \in \mathcal{T}$ will be used as a shortcut for $t \in \mathcal{T}_p$), every class $C$, and every $n$-ary relation $R$, we have $C^{\mathcal{I}(t)} \subseteq \Delta^{\mathcal{I}}$ and $R^{\mathcal{I}(t)} \subseteq (\Delta^{\mathcal{I}})^n$. The semantics of class and relation expressions is defined in the lower part of Figure 5, where $(u, v) = \{w \in \mathcal{T} \mid u < w < v\}$.

We will use the following equivalent abbreviations: $C_1 \sqcup C_2 \equiv \neg(\neg C_1 \sqcap \neg C_2)$; $C_1 \rightarrow C_2 \equiv \neg C_1 \sqcup C_2$; $\exists[U]R \equiv \exists^{\geq 1}[U]R$; $\forall[U]R \equiv \neg\exists[U]\neg R$; $R_1 \sqcup R_2 \equiv \neg(\neg R_1 \sqcap \neg R_2)$. Furthermore, the operators $\Diamond^*$ (at some moment) and its dual $\Box^*$ (at all moments) can be defined for both classes and relations as $\Diamond^* C \equiv C \sqcup \Diamond^+ C \sqcup \Diamond^- C$ and $\Box^* C \equiv C \sqcap \Box^+ C \sqcap \Box^- C$, respectively.

A *knowledge base* is a finite set $\Sigma$ of $\mathcal{DLR}_{\mathcal{US}}$ axioms of the form $C_1 \sqsubseteq C_2$ and $R_1 \sqsubseteq R_2$, and with $R_1$ and $R_2$ being relations of the same arity. An interpretation $\mathcal{I}$ satisfies $C_1 \sqsubseteq C_2$ ($R_1 \sqsubseteq R_2$) if and only if the interpretation of $C_1$ ($R_1$) is included in the interpretation of $C_2$ ($R_2$) at all time, i.e. $C_1^{\mathcal{I}(t)} \subseteq C_2^{\mathcal{I}(t)}$ ($R_1^{\mathcal{I}(t)} \subseteq R_2^{\mathcal{I}(t)}$), for all $t \in \mathcal{T}$. Various *reasoning services* can be defined in $\mathcal{DLR}_{\mathcal{US}}$, such as satisfiability, logical implication and class (relation) subsumption (see Baader *et al.* (2003) for details). While $\mathcal{DLR}$ knowledge bases are fully able to capture atemporal EER schemas (Berardi *et al.*, 2005; Calvanese *et al.*, 1998a,b)—i.e. given an EER schema there is an equi-satisfiable $\mathcal{DLR}$ knowledge base—in the following Sections we show how $\mathcal{DLR}_{\mathcal{US}}$ knowledge bases can capture temporal EER schemas with both timestamping and evolution constraints.

### 3.2 The Temporal Conceptual Model $\mathcal{ER}_{VT}$

In this Section, the temporal EER model $\mathcal{ER}_{VT}$—which will be the basis to present our proposal—is briefly introduced (see (Artale & Franconi, 1999; Artale *et al.*, 2003) for full details). $\mathcal{ER}_{VT}$ supports timestamping for classes, attributes, and relationships. $\mathcal{ER}_{VT}$ is equipped with both a textual and a graphical syntax along with a model-theoretic semantics as a temporal extension of the EER semantics (Calvanese *et al.*, 1999). The formal foundations of $\mathcal{ER}_{VT}$ allowed also to prove a correct encoding of $\mathcal{ER}_{VT}$ schemas as knowledge base in $\mathcal{DLR}_{\mathcal{US}}$ (Artale *et al.*, 2002, 2003).

**Definition 1 ($\mathcal{ER}_{VT}$ Conceptual Data Model).** *An $\mathcal{ER}_{VT}$ conceptual data model is a tuple: $\Sigma = (\mathcal{L}, \text{REL}, \text{ATT}, \text{CARD}, \text{ISA}, \text{DISJ}, \text{COVER}, \text{S}, \text{T}, \text{KEY})$, such that: $\mathcal{L}$ is a finite alphabet partitioned into the sets: $\mathcal{C}$ (*class *symbols), $\mathcal{A}$ (*attribute *symbols), $\mathcal{R}$ (*relationship *symbols), $\mathcal{U}$ (*role *symbols), and $\mathcal{D}$ (*domain *symbols) and*

1. *The set $\mathcal{C}$ of class symbols is partitioned into a set $\mathcal{C}^S$ of* Snapshot classes *(marked with an $\mathsf{S}$), a set $\mathcal{C}^M$ of* Mixed classes *(unmarked classes), and a set $\mathcal{C}^T$ of* Temporary classes *(marked with a $\mathsf{T}$). A similar partition applies to the set $\mathcal{R}$.*
2. ATT *is a function that maps a class symbol in $\mathcal{C}$ to an $\mathcal{A}$-labeled tuple over $\mathcal{D}$, $\text{ATT}(C) = \langle A_1 : D_1, \ldots, A_h : D_h \rangle$.*
3. REL *is a function that maps a relationship symbol in $\mathcal{R}$ to an $\mathcal{U}$-labeled tuple over $\mathcal{C}$, $\text{REL}(R) = \langle U_1 : C_1, \ldots, U_k : C_k \rangle$, and $k$ is the* arity *of $R$.*

4. CARD *is a function* $\mathcal{C} \times \mathcal{R} \times \mathcal{U} \mapsto \mathbb{N} \times (\mathbb{N} \cup \{\infty\})$ *denoting cardinality constraints. We denote with* $\text{CMIN}(C, R, U)$ *and* $\text{CMAX}(C, R, U)$ *the first and second component of* CARD.
5. ISA *is a binary relationship* $\text{ISA} \subseteq (\mathcal{C} \times \mathcal{C}) \cup (\mathcal{R} \times \mathcal{R})$. ISA *between relationships is restricted to relationships with the same arity.* ISA *is visualized with a directed arrow.*
6. DISJ, COVER *are binary relations over* $(2^{\mathcal{C}} \times \mathcal{C}) \times (2^{\mathcal{R}} \times \mathcal{R})$, *describing disjointness and covering partitions, respectively, over a group of* ISA *that share the same superclass/super-relation.* DISJ *is visualized with a circled "*$\mathfrak{d}$*" and* COVER *with a double directed arrow.*
7. S, T *are binary relations over* $\mathcal{C} \times \mathcal{A}$ *containing, respectively, the snapshot and temporary attributes of a class;*
8. KEY *is a function,* $\text{KEY} : \mathcal{C} \rightarrow \mathcal{A}$, *that maps a class symbol in* $\mathcal{C}$ *to its key attribute. Keys are visualized as underlined attributes.*

The model-theoretic semantics associated with the $\mathcal{ER}_{VT}$ modelling language adopts the snapshot representation of temporal conceptual data models (Chomicki & Toman, 1998)[7].

**Definition 2 ($\mathcal{ER}_{VT}$ Semantics).** *Let* $\Sigma$ *be an* $\mathcal{ER}_{VT}$ *schema. A* temporal database state *for the schema* $\Sigma$ *is a tuple* $\mathcal{B} = (\mathcal{T}, \Delta^{\mathcal{B}} \cup \Delta^{\mathcal{B}}_D, \cdot^{\mathcal{B}(t)})$, *such that:* $\Delta^{\mathcal{B}}$ *is a nonempty set of abstract objects disjoint from* $\Delta^{\mathcal{B}}_D$; $\Delta^{\mathcal{B}}_D = \bigcup_{D_i \in \mathcal{D}} \Delta^{\mathcal{B}}_{D_i}$ *is the set of basic domain values used in the schema* $\Sigma$; *and* $\cdot^{\mathcal{B}(t)}$ *is a function that for each* $t \in \mathcal{T}$ *maps:*

- *Every basic domain symbol* $D_i$ *into a set* $D_i^{\mathcal{B}(t)} = \Delta^{\mathcal{B}}_{D_i}$.
- *Every class* $C$ *to a set* $C^{\mathcal{B}(t)} \subseteq \Delta^{\mathcal{B}}$ *—thus* objects *are instances of classes.*
- *Every relationship* $R$ *to a set* $R^{\mathcal{B}(t)}$ *of* $\mathcal{U}$*-labeled tuples over* $\Delta^{\mathcal{B}}$ *—i.e. let* $R$ *be an n-ary relationship connecting the classes* $C_1, \ldots, C_n$, $\text{REL}(R) = \langle U_1 : C_1, \ldots, U_n : C_n \rangle$, *then,* $r \in R^{\mathcal{B}(t)} \rightarrow (r = \langle U_1 : o_1, \ldots, U_n : o_n \rangle \wedge \forall i \in \{1, \ldots, n\}.o_i \in C_i^{\mathcal{B}(t)})$. *We adopt the convention:* $\langle U_1 : o_1, \ldots, U_n : o_n \rangle \equiv \langle o_1, \ldots, o_n \rangle$, *when* $\mathcal{U}$*-labels are clear from the context.*
- *Every attribute* $A$ *to a set* $A^{\mathcal{B}(t)} \subseteq \Delta^{\mathcal{B}} \times \Delta^{\mathcal{B}}_D$, *such that, for each* $C \in \mathcal{C}$, *if* $\text{ATT}(C) = \langle A_1 : D_1, \ldots, A_h : D_h \rangle$, *then,* $o \in C^{\mathcal{B}(t)} \rightarrow (\forall i \in \{1, \ldots, h\}, \exists a_i . \langle o, a_i \rangle \in A_i^{\mathcal{B}(t)} \wedge \forall a_i . \langle o, a_i \rangle \in A_i^{\mathcal{B}(t)} \rightarrow a_i \in \Delta^{\mathcal{B}}_{D_i})$.

$\mathcal{B}$ *is said a* legal temporal database state *if it satisfies all of the constraints expressed in the schema, i.e. for each* $t \in \mathcal{T}$*:*

- *For each* $C_1, C_2 \in \mathcal{C}$, *if* $C_1$ ISA $C_2$, *then,* $C_1^{\mathcal{B}(t)} \subseteq C_2^{\mathcal{B}(t)}$.
- *For each* $R_1, R_2 \in \mathcal{R}$, *if* $R_1$ ISA $R_2$, *then,* $R_1^{\mathcal{B}(t)} \subseteq R_2^{\mathcal{B}(t)}$.
- *For each cardinality constraint* $\text{CARD}(C, R, U)$, *then:* $o \in C^{\mathcal{B}(t)} \rightarrow \text{CMIN}(C, R, U) \leq \#\{r \in R^{\mathcal{B}(t)} \mid r[U] = o\} \leq \text{CMAX}(C, R, U)$.
- *For* $C, C_1, \ldots, C_n \in \mathcal{C}$, *if* $\{C_1, \ldots, C_n\}$ DISJ $C$, *then,* $\forall i \in \{1, \ldots, n\}.C_i$ ISA $C \wedge \forall j \in \{1, \ldots, n\}$, $j \neq i . C_i^{\mathcal{B}(t)} \cap C_j^{\mathcal{B}(t)} = \emptyset$. *(Similar for* $\{R_1, \ldots, R_n\}$ DISJ $R$*)*
- *For* $C, C_1, \ldots, C_n \in \mathcal{C}$, *if* $\{C_1, \ldots, C_n\}$ COVER $C$, *then,* $\forall i \in \{1, \ldots, n\}.C_i$ ISA $C \wedge C^{\mathcal{B}(t)} = \bigcup_{i=1}^{n} C_i^{\mathcal{B}(t)}$. *(Similar for* $\{R_1, \ldots, R_n\}$ COVER $R$*)*
- *For each snapshot class* $C \in \mathcal{C}^S$, *then,* $o \in C^{\mathcal{B}(t)} \rightarrow \forall t' \in \mathcal{T} . o \in C^{\mathcal{B}(t')}$.
- *For each temporary class* $C \in \mathcal{C}^T$, *then,* $o \in C^{\mathcal{B}(t)} \rightarrow \exists t' \neq t . o \notin C^{\mathcal{B}(t')}$.
- *For each snapshot relationship* $R \in \mathcal{R}^S$, *then,* $r \in R^{\mathcal{B}(t)} \rightarrow \forall t' \in \mathcal{T} . r \in R^{\mathcal{B}(t')}$.
- *For each temporary relationship* $R \in \mathcal{R}^T$, *then,* $r \in R^{\mathcal{B}(t)} \rightarrow \exists t' \neq t . r \notin R^{\mathcal{B}(t')}$.

- *For each class* $C \in \mathcal{C}$*, if* $\text{ATT}(C) = \langle A_1 : D_1, \ldots, A_h : D_h \rangle$*, and* $\langle C, A_i \rangle \in \text{S}$*, then,* $(o \in C^{\mathcal{B}(t)} \wedge \langle o, a_i \rangle \in A_i^{\mathcal{B}(t)}) \rightarrow \forall t' \in \mathcal{T}.\langle o, a_i \rangle \in A_i^{\mathcal{B}(t')}$.
- *For each class* $C \in \mathcal{C}$*, if* $\text{ATT}(C) = \langle A_1 : D_1, \ldots, A_h : D_h \rangle$*, and* $\langle C, A_i \rangle \in \text{T}$*, then,* $(o \in C^{\mathcal{B}(t)} \wedge \langle o, a_i \rangle \in A_i^{\mathcal{B}(t)}) \rightarrow \exists t' \neq t.\langle o, a_i \rangle \notin A_i^{\mathcal{B}(t')}$.
- *For each* $C \in \mathcal{C}, A \in \mathcal{A}$ *such that* $\text{KEY}(C) = A$*, then,* $A$ *is a snapshot attribute–i.e.* $\langle C, A_i \rangle \in \text{S}$*— and* $\forall a \in \Delta_D^{\mathcal{B}}.\#\{o \in C^{\mathcal{B}(t)} \mid \langle o, a \rangle \in A^{\mathcal{B}(t)}\} \leq 1$.

Given such a set-theoretic semantics for the temporal EER (or, for that matter, UML class diagrams or ORM), some relevant modelling notions such as satisfiability, subsumption, and derivation of new constraints by means of logical implication have been defined rigorously (Artale *et al.*, 2007b).

**Definition 3 (Reasoning Services).** *Let* $\Sigma$ *be a schema,* $C \in \mathcal{C}$ *a class, and* $R \in \mathcal{R}$ *a relationship. The following modelling notions can be defined:*

1. $C$ *(*$R$*) is* satisfiable *if there exists a legal temporal database state* $\mathcal{B}$ *for* $\Sigma$ *such that* $C^{\mathcal{B}(t)} \neq \emptyset$ *(*$R^{\mathcal{B}(t)} \neq \emptyset$*), for some* $t \in \mathcal{T}$*;*
2. $\Sigma$ *is* satisfiable *if there exists a legal temporal database state* $\mathcal{B}$ *for* $\Sigma$ *(*$\mathcal{B}$ *is also said a* model *for* $\Sigma$*);*
3. $C_1$ *(*$R_1$*) is* subsumed *by* $C_2$ *(*$R_2$*) in* $\Sigma$ *if every legal temporal database state for* $\Sigma$ *is also a legal temporal database state for* $C_1$ ISA $C_2$ *(*$R_1$ ISA $R_2$*);*
4. *A schema* $\Sigma'$ *is* logically implied *by a schema* $\Sigma$ *over the same signature if every legal temporal database state for* $\Sigma$ *is also a legal temporal database state for* $\Sigma'$*.*

### 3.3 Mapping $\mathcal{ER}_{VT}$ into $\mathcal{DLR}_{\mathcal{US}}$

We briefly summarize how $\mathcal{DLR}_{\mathcal{US}}$ is able to capture temporal schemas expressed in $\mathcal{ER}_{VT}$—see (Berardi *et al.*, 2005; Artale *et al.*, 2006) for more details.

**Definition 4 (Mapping $\mathcal{ER}_{VT}$ into $\mathcal{DLR}_{\mathcal{US}}$).** *Let* $\Sigma = (\mathcal{L}, \text{REL}, \text{ATT}, \text{CARD}, \text{ISA}, \text{DISJ}, \text{COVER}, \text{S}, \text{T}, \text{KEY})$ *be an* $\mathcal{ER}_{VT}$ *schema. The* $\mathcal{DLR}_{\mathcal{US}}$ *knowledge base,* $\mathcal{K}$*, mapping* $\Sigma$ *is as follows.*

- *For each* $A \in \mathcal{A}$*, then,* $A \sqsubseteq \texttt{From}:\top \sqcap \texttt{To}:\top \in \mathcal{K}$*;*
- *If* $C_1$ ISA $C_2 \in \Sigma$*, then,* $C_1 \sqsubseteq C_2 \in \mathcal{K}$*;*
- *If* $R_1$ ISA $R_2 \in \Sigma$*, then,* $R_1 \sqsubseteq R_2 \in \mathcal{K}$*;*
- *If* $\text{REL}(R) = \langle U_1 : C_1, \ldots, U_k : C_k \rangle \in \Sigma$*, then* $R \sqsubseteq U_1 : C_1 \sqcap \ldots \sqcap U_k : C_k \in \mathcal{K}$*;*
- *If* $\text{ATT}(C) = \langle A_1 : D_1, \ldots, A_h : D_h \rangle \in \Sigma$*, then,* $C \sqsubseteq \exists[\texttt{From}]A_1 \sqcap \ldots \sqcap \exists[\texttt{From}]A_h \sqcap \forall[\texttt{From}](A_1 \rightarrow \texttt{To} : D_1) \sqcap \ldots \sqcap \forall[\texttt{From}](A_h \rightarrow \texttt{To} : D_h) \in \mathcal{K}$*;*
- *If* $\text{CARD}(C, R, U) = (m, n) \in \Sigma$*, then,* $C \sqsubseteq \exists^{\geq m}[U]R \sqcap \exists^{\leq n}[U]R \in \mathcal{K}$*;*
- *If* $\{C_1, \ldots, C_n\}$ DISJ $C \in \Sigma$*, then* $\mathcal{K}$ *contains:*
  $C_1 \sqsubseteq C \sqcap \neg C_2 \sqcap \ldots \sqcap \neg C_n$*;*
  $C_2 \sqsubseteq C \sqcap \neg C_3 \sqcap \ldots \sqcap \neg C_n$*;*
  $\ldots$
  $C_n \sqsubseteq C$*;*
- *If* $\{R_1, \ldots, R_n\}$ DISJ $R \in \Sigma$*, then* $\mathcal{K}$ *contains:*
  $R_1 \sqsubseteq R \sqcap \neg R_2 \sqcap \ldots \sqcap \neg R_n$*;*
  $R_2 \sqsubseteq R \sqcap \neg R_3 \sqcap \ldots \sqcap \neg R_n$*;*
  $\ldots$
  $R_n \sqsubseteq R$*;*

– *If* $\{C_1, \ldots, C_n\}$ COVER $C \in \Sigma$, *then* $\mathcal{K}$ *contains:*
  $C_1 \sqsubseteq C$;
  ...
  $C_n \sqsubseteq C$;
  $C \sqsubseteq C_1 \sqcup \ldots \sqcup C_n$;
– *If* $\{R_1, \ldots, R_n\}$ COVER $R \in \Sigma$, *then* $\mathcal{K}$ *contains:*
  $R_1 \sqsubseteq R$;
  ...
  $R_n \sqsubseteq R$;
  $R \sqsubseteq R_1 \sqcup \ldots \sqcup R_n$;
– *If* KEY$(C) = A$, *then,* $\mathcal{K}$ *contains:*
  $C \sqsubseteq \exists^{=1}[\texttt{From}]\Box^* A$;
  $\top \sqsubseteq \exists^{\leq 1}[\texttt{To}](A \sqcap [\texttt{From}] : C)$;
– *If* $C \in \mathcal{C}^S$, *then,* $C \sqsubseteq (\Box^* C) \in \mathcal{K}$ *(similar for* $R \in \mathcal{R}^S$*);*
– *If* $C \in \mathcal{C}^T$, *then,* $C \sqsubseteq (\Diamond^* \neg C) \in \mathcal{K}$ *(similar for* $R \in \mathcal{R}^T$*);*
– *If* $\langle C, A \rangle \in$ S, *then,* $C \sqsubseteq \forall[\texttt{From}](A \rightarrow \Box^* A) \in \mathcal{K}$;
– *If* $\langle C, A \rangle \in$ T, *then,* $C \sqsubseteq \forall[\texttt{From}](A \rightarrow \Diamond^* \neg A) \in \mathcal{K}$.

In the next sections we extend the formalism presented here to capture essential and sharable parts.

# 4 MODELING ESSENTIAL PARTS AND WHOLES

This section presents a formalization of the notion of *essential* part-whole relations. To formalize such properties of part-whole relations we will resort to the formalism introduced in the previous section. As a result, the $\mathcal{ER}_{VT}$ data model will be extended with the possibility to capture such part-whole properties while the description logic $\mathcal{DLR}_{\mathcal{US}}$ will present a corresponding axiomatization for them. A basic building block to achieve the desired formalization is the notion of *status relations*. The formalization of status relations is an original contribution of this chapter. They are in analogy with status classes addressed by (Artale *et al.*, 2007b) and will be useful for modeling essential part-whole relations. We therefore start by introducing status relations in the following subsection and then we proceed by formalizing essential parts and wholes.

## 4.1 Status Relations

*Status relations* extend the notion of *status classes* (Spaccapietra *et al.*, 1998; Etzion *et al.*, 1998; Artale *et al.*, 2007b) to statuses for relations. Status classes—formalized in (Artale *et al.*, 2007b)—constrain the evolution of an instance's membership in a class along its lifespan. According to (Spaccapietra *et al.*, 1998; Artale *et al.*, 2007b), status modeling includes up to four different statuses *scheduled, active, suspended, disabled*, each one entailing different constraints.

Concerning status relations there are two options: (1) to derive a relation's status from the status of the classes participating in the relation, or (2) to explicitly define it on the relation itself, where the latter, in turn, puts constraints on the statuses of the classes. Since we are interested in modeling relations as first-class citizens, we choose to have a means to explicitly model the status of a relation. Therefore, as for classes, we have four different statuses for relations, too—*scheduled, active, suspended, disabled*—each illustrated with an example before we proceed to the formal characterization.

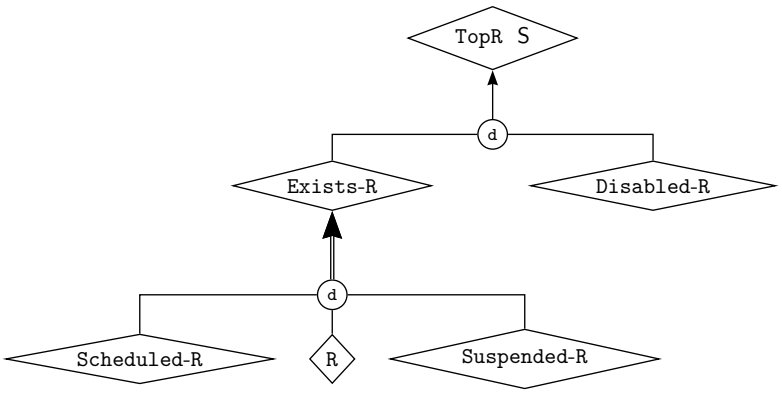


**Fig. 6.** Status relations (from status classes in Artale *et al.* (2007a)).

- Scheduled: a relation is scheduled if its instantiation is known but its membership will only become effective some time later. Objects in its participating classes must be either scheduled, too, be active, or suspended. For instance, a pillar for finishing the interior of the Sagrada Familia in Barcelona is scheduled to become part of that church, i.e., this *part_of* relation between the pillar and the church is scheduled.
- Active: the status of a relation is active if the particular relation fully instantiates the type-level relation: the part is part of the whole. For instance, the Mont Blanc mountain is part of the Alps mountain range, and the country Republic of Ireland is part of the European Union. Only active classes can participate in an active relation.
- Suspended: to capture a temporarily inactive relation. For example, an instance of a CarEngine is removed from the instance of a Car it is part of, for purpose of maintenance at the car mechanic. Note that at the moment of suspension, part $p$ and $w$ must be active, but can upon suspension of the relation be either active or become suspended too, but neither scheduled (see below constraints on scheduled) nor disabled.
- Disabled: to model expired relations that never again can be used. For instance, to represent the donor of an organ who has donated that organ and one wants to keep track of who donated what to whom: say, the heart $p_1$ of donor $w_1$ used to be a structural part of $w_1$ but it will never be again a part of it. The heart, $p_1$, then may have become participant in a new part-of relation with a new whole, $w_2$ where $w_1 \neq w_2$, but the original part-of between $p_1$ and $w_1$ remains disabled. Observe that participating objects can be member of the active, suspended or disabled class.

Status relations apply only to temporal relations (i.e. either temporary or mixed relations according to Definition 1). We assume that active relations involve only active classes and, by default, the name of a relation denotes already its active status—i.e. $\texttt{Active-R} \equiv \texttt{R}$. Disjointness and ISA constraints among the four status relations are analogous to the one for status classes and can be represented in $\mathcal{ER}_{VT}$ as illustrated in Figure 6. In addition to hierarchical constraints, the following constraints hold (we present both the model-theoretic semantics and the correspondent $\mathcal{DLR}_{\mathcal{US}}$ axioms considering, wlog, binary relations):

(ACT) *Active relations involve only active classes.*
$\langle o_1, o_2\rangle \in \texttt{R}^{\mathcal{B}(t)} \rightarrow o_i \in C_i^{\mathcal{B}(t)},\ i = 1, 2$
$\texttt{R} \sqsubseteq \texttt{U}_\texttt{i}\texttt{:C}_\texttt{i},\ i = 1, 2$

(REXISTS) *Existence persists until Disabled.*
$\langle o_1, o_2\rangle \in \texttt{Exists-R}^{\mathcal{B}(t)} \rightarrow \forall t' > t.(\langle o_1, o_2\rangle \in \texttt{Exists-R}^{\mathcal{B}(t')} \vee \langle o_1, o_2\rangle \in \texttt{Disabled-R}^{\mathcal{B}(t')})$
$\texttt{Exists-R} \sqsubseteq \Box^+(\texttt{Exists-R} \sqcup \texttt{Disabled-R})$

(RDisab1) *Disabled persists.*
$\langle o_1, o_2\rangle \in \texttt{Disabled-R}^{\mathcal{B}(t)} \rightarrow \forall t' > t.\langle o_1, o_2\rangle \in \texttt{Disabled-R}^{\mathcal{B}(t')}$
$\texttt{Disabled-R} \sqsubseteq \Box^{+}\texttt{Disabled-R}$
(RDisab2) *Disabled was Active in the past.*
$\langle o_1, o_2\rangle \in \texttt{Disabled-R}^{\mathcal{B}(t)} \rightarrow \exists t' < t.\langle o_1, o_2\rangle \in \texttt{R}^{\mathcal{B}(t')}$
$\texttt{Disabled-R} \sqsubseteq \Diamond^{-}\texttt{R}$
(RSusp1) *Suspended was Active in the past.*
$\langle o_1, o_2\rangle \in \texttt{Suspended-R}^{\mathcal{B}(t)} \rightarrow \exists t' < t.\langle o_1, o_2\rangle \in \texttt{R}^{\mathcal{B}(t')}$
$\texttt{Suspended-R} \sqsubseteq \Diamond^{-}\texttt{R}$
(RSusp2) *Suspended involve Active or Suspended Classes.*
$\langle o_1, o_2\rangle \in \texttt{Suspended-R}^{\mathcal{B}(t)} \rightarrow o_i \in C_i^{\mathcal{B}(t)} \vee o_i \in \texttt{Suspended-C}_\texttt{i}{}^{\mathcal{B}(t)},\ i = 1, 2$
$\texttt{Suspended-R} \sqsubseteq \texttt{U}_\texttt{i} : (\texttt{C}_\texttt{i} \sqcup \texttt{Suspended-C}_\texttt{i}),\ i = 1, 2$
(RSch1) *Scheduled will eventually become Active.*
$\langle o_1, o_2\rangle \in \texttt{Scheduled-R}^{\mathcal{B}(t)} \rightarrow \exists t' > t.\langle o_1, o_2\rangle \in \texttt{R}^{\mathcal{B}(t')}$
$\texttt{Scheduled-R} \sqsubseteq \Diamond^{+}\texttt{R}$
(RSch2) *Scheduled can never follow Active.*
$\langle o_1, o_2\rangle \in \texttt{R}^{\mathcal{B}(t)} \rightarrow \forall t' > t.\langle o_1, o_2\rangle \notin \texttt{Scheduled-R}^{\mathcal{B}(t')}$
$\texttt{R} \sqsubseteq \Box^{+}\neg\texttt{Scheduled-R}$

In the following we denote with $\Sigma_{st}$ the above set of $\mathcal{DLR}_{\mathcal{US}}$ axioms that formalize status relations. In analogy with the logical implications holding for status classes (Artale *et al.*, 2007b), we can derive the following ones for status relations.

**Proposition 1 (Status Relations: Logical Implications).** *Given the set of axioms $\Sigma_{st}$ (Act-RSch2), an n-ary relation (where $n \geq 2$) $R \sqsubseteq U_1 : C_1 \sqcap \ldots \sqcap U_n : C_n$, the following logical implications hold:*

(RAct) *Active will possible evolve into Suspended or Disabled.*
$\Sigma_{st} \models \texttt{R} \sqsubseteq \Box^{+}(\texttt{R} \sqcup \texttt{Suspended-R} \sqcup \texttt{Disabled-R})$
(RDisab3) *Disabled will never become active anymore.*
$\Sigma_{st} \models \texttt{Disabled-R} \sqsubseteq \Box^{+}\neg\texttt{R}$
(RDisab4) *Disabled classes can participate only in disabled relations.*
$\Sigma_{st} \models \texttt{Disabled-C}_\texttt{i} \sqcap \Diamond^{-}\exists[\texttt{U}_\texttt{i}]\texttt{R} \sqsubseteq \exists[\texttt{U}_\texttt{i}]\texttt{Disabled-R}$
(RDisab5) *Disabled relations involve active, suspended, or disabled classes.*
$\texttt{Disabled-R} \sqsubseteq \texttt{U}_i\texttt{:(C}_i \sqcup \texttt{Suspended-C}_i \sqcup \texttt{Disabled-C}_i\texttt{)}$, *for all* $i = 1, \ldots, n$.
(RSch3) *Scheduled persists until active.*
$\Sigma_{st} \models \texttt{Scheduled-R} \sqsubseteq \texttt{Scheduled-R}\,\mathcal{U}\,\texttt{R}$
(RSch4) *Scheduled cannot evolve directly to Disabled.*
$\Sigma_{st} \models \texttt{Scheduled-R} \sqsubseteq \oplus\,\neg\texttt{Disabled-R}$
(RSch5) *Scheduled relations do not involve disabled classes.*
$\texttt{Scheduled-R} \sqsubseteq \texttt{U}_\texttt{i} : \neg\texttt{Dibabled-C}_\texttt{i}$, *for all* $i = 1, \ldots, n$.

The proofs of these logical implications have been presented in (Artale & Keet, 2008).

**Lifespan and related notions.** The lifespan of an object with respect to a class describes the temporal instants (and thus intervals) where the object can be considered a member of that class. We can distinguish between the following notions: ExistenceSpan$_C$, LifeSpan$_C$, ActiveSpan$_C$, Begin$_C$, Birth$_C$, and Death$_C$ depending on the status of the class the object is member of. We briefly report here their definition as presented in (Artale *et al.*, 2007b).

$$\text{ExistenceSpan}_C(o) = \{t \in \mathcal{T} \mid o \in \texttt{Exists-C}^{\mathcal{B}(t)}\}$$

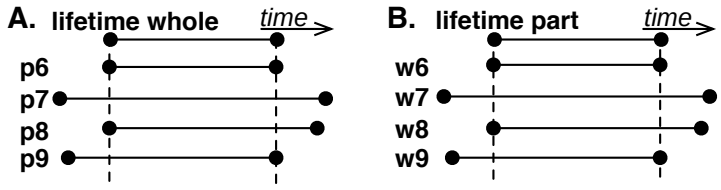


**Fig. 7.** Lifespans of essential parts w.r.t. the whole (A) and vv (B).

$\textsc{LifeSpan}_C(o) = \{t \in \mathcal{T} \mid o \in \mathtt{C}^{\mathcal{B}(t)} \cup \mathtt{Suspended\text{-}C}^{\mathcal{B}(t)}\}$
$\textsc{ActiveSpan}_C(o) = \{t \in \mathcal{T} \mid o \in \mathtt{C}^{\mathcal{B}(t)}\}$
$\textsc{Begin}_C(o) = \mathtt{min}(\textsc{ExistenceSpan}_C(o))$
$\textsc{Birth}_C(o) = \mathtt{min}(\textsc{ActiveSpan}_C(o)) \equiv \mathtt{min}(\textsc{LifeSpan}_C(o))$
$\textsc{Death}_C(o) = \mathtt{max}(\textsc{LifeSpan}_C(o))$

For atemporal classes, $\textsc{ExistenceSpan}_C(o) \equiv \textsc{LifeSpan}_C(o) \equiv \textsc{ActiveSpan}_C(o) \equiv \mathcal{T}$. This concludes the preliminaries. In the next section we will use the notions introduced so far for representing essential parts-whole relations.

### 4.2 Essential Parts and Wholes

Recollecting Guizzardi's (2005) contribution on the formalization of the difference between mandatory and essential parts and wholes we can say that: a part is *mandatory* if the whole cannot exist without it, which can also be verbalized as "the whole has a mandatory part"—i.e. a standard mandatory constraint on the role played by the whole in a part-whole relation. In a symmetric way we can define *mandatory wholes*. A part is *essential* if it is mandatory and cannot change without destroying the whole, i.e. "the whole has an essential part" (in an analogous way we can define *essential wholes*). Furthermore, we say that a part is *exclusive* if it can be part of at most one whole (similarly for *exclusive wholes*). In this section we provide a formalization using $\mathcal{DLR}_{\mathcal{US}}$ axioms of such mandatory, essential and exclusive parts and wholes. Starting from Figure 4, Figure 7-A, shows the various temporal relations that can hold between a whole and its essential part, i.e. the lifespan of the whole is fixed and we consider the different lifespans for its essential parts (Figure 7-B considers a fixed part and the cases for its essential whole)[8].

Let $\mathtt{partOf} \sqsubseteq \mathtt{part{:}P} \sqcap \mathtt{whole{:}W}$ be a generic part-whole relation, the following $\mathcal{DLR}_{\mathcal{US}}$ axioms give a formalization of *mandatory* and *exclusive* parts and wholes:

| | | |
|---|---|---|
| (ManP) | $\mathtt{W} \sqsubseteq \exists[\mathtt{whole}]\mathtt{partOf}$ | *Has Mandatory Part* |
| (ManW) | $\mathtt{P} \sqsubseteq \exists[\mathtt{part}]\mathtt{partOf}$ | *Has Mandatory Whole* |
| (ExlP) | $\mathtt{P} \sqsubseteq \exists^{\leq 1}[\mathtt{part}]\mathtt{partOf}$ | *Is Exclusive Part* |
| (ExlW) | $\mathtt{W} \sqsubseteq \exists^{\leq 1}[\mathtt{whole}]\mathtt{partOf}$ | *Is Exclusive Whole* |

To capture essential parts and wholes, in addition to the above axioms, we will use appropriate subsets of the following axioms.

| | | |
|---|---|---|
| (ConPO) | $\mathtt{Suspended\text{-}partOf} \sqsubseteq \bot$ | *Continuous Parts* |
| (DisP) | $\mathtt{Disabled\text{-}partOf} \sqsubseteq \mathtt{part} : \mathtt{Disabled\text{-}P}$ | *Disabled Part* |
| (DisW) | $\mathtt{Disabled\text{-}partOf} \sqsubseteq \mathtt{whole} : \mathtt{Disabled\text{-}W}$ | *Disabled Whole* |
| (SchPO) | $\mathtt{partOf} \sqsubseteq \Diamond^{-}\mathtt{Scheduled\text{-}partOf}$ | *Scheduled Part-Whole* |
| (SchP) | $\mathtt{Scheduled\text{-}partOf} \sqsubseteq \mathtt{part} : \mathtt{Scheduled\text{-}P}$ | *Scheduled Part* |
| (SchW) | $\mathtt{Scheduled\text{-}partOf} \sqsubseteq \mathtt{whole} : \mathtt{Scheduled\text{-}W}$ | *Scheduled Whole* |

We can now show that the above axiomatization is sufficient to represent the various forms of mandatory and essential parts as shown in Figure 7.

**Theorem 1 (Essential Parts).** *Let* `partOf` $\sqsubseteq$ `part:P` $\sqcap$ `whole:W` *be a generic part-whole relation satisfying* $\Sigma_{st}$*, then,*

1. *p7 holds if* (ManP), (ConPO), (DisW) *hold;*
2. *p9 holds if* (ManP), (ConPO), (DisW), (DisP) *hold;*
3. *p8 holds if* (ManP), (ConPO), (DisW), (SchPO), (SchP) *hold;*
4. *p6 holds if* (ManP), (ConPO), (DisW), (DisP), (SchPO), (SchP) *hold.*

The proof has been presented in (Artale & Keet, 2008). A similar result can be proved considering the various forms of essential wholes.

**Theorem 2 (Essential Wholes).** *Let* `partOf` $\sqsubseteq$ `part:P` $\sqcap$ `whole:W` *be a generic part-whole relation satisfying* $\Sigma_{st}$*, then,*

1. *w7 holds if* (ManW), (ConPO), (DisP) *hold;*
2. *w9 holds if* (ManW), (ConPO), (DisP), (DisW) *hold;*
3. *w8 holds if* (ManW), (ConPO), (DisP), (SchPO), (SchW) *hold;*
4. *w6 holds if* (ManW), (ConPO), (DisP), (DisW), (SchPO), (SchW) *hold.*

Thus, from the axiomatization presented above, the essential parts and wholes in a part-whole relation are always active, cannot be suspended, and when the strict case is allowed (i.e. either p6 or w6 holds) then they are either *both* member of their respective Scheduled class, or both Active, or both member of their respective Disabled classes. Hence, a change of membership from one of the two objects implies *instantaneous* change of the other in the same type of status class.

Note that, in the literature, essential parts are often considered also exclusive. Our modeling of essential parts and wholes can be easily extended by adding to the axiomatization of Theorems 1-2 either the axiom (ExlP) or (ExlW) depending whether we want to capture exclusive essential parts or wholes.

This concludes the formal characterization of the principal combinations in life cycles between essential parts and wholes. The next section provides a solution sketch on how to deal both with suspended part-whole relations and with parts shared among (possibly different) wholes.

### 4.3 Shared Parts and Wholes

Sharing of parts and wholes permits many possibilities, some of which may be more useful in practice than others. Rather than enumerating, formalizing, and, where necessary, proving all theoretically possible options, we discuss the main typical cases and demonstrate they indeed can be characterized *by using the same formal apparatus* and principles that have been introduced in the previous sections. The first step is to add the possibility to suspend parts, wholes, and part-whole relations, and the second step to make explicit that some parts can be part of more than one whole of the same or different type.

**Taking into account suspension.** The first variation that can be added to the basic combinations of Figure 7, is that during some time either the participating part or the whole, or both, is suspended or the part-whole relation is suspended. This offers a wide range of possibilities. For instance, we have a word processing document management

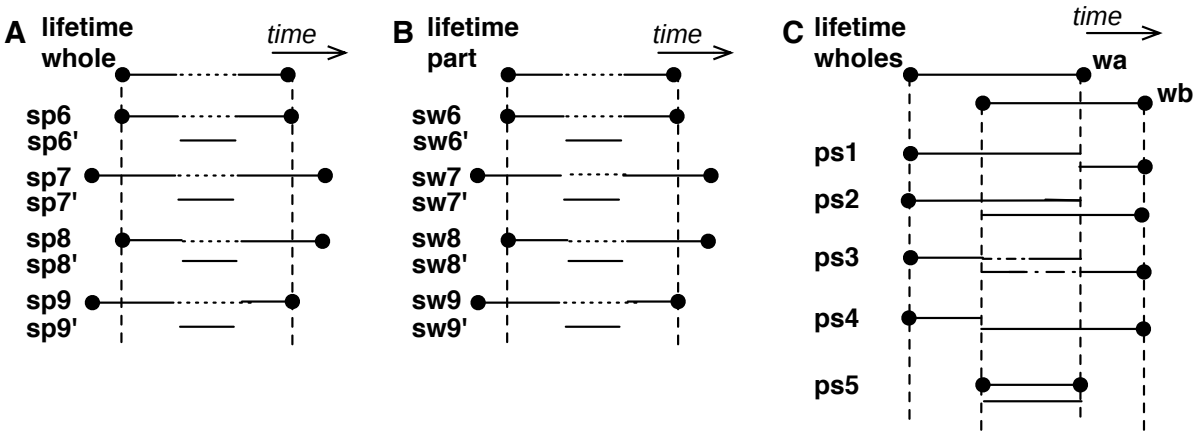


**Fig. 8.** (A) Eight permutations for suspension from the viewpoint of the whole; (B) analogously from the part viewpoint; (C) Solid line as being part of the whole, dotted line as being *not* part of *that* whole but either the part-whole relation is suspended, or $p$ is part of another whole, or both. For ps1-ps5 $w_a, w_b \in W$; the analogous situation where $w_a \in W$ and $w_b \in W'$ s.t. $W \neq W'$ is not drawn.

system where a particular paragraph (part $p$) is blocked—suspended—for use and the particular file (whole $w$) can only be published when the paragraph is member of the active class again. Differently, one could have a defunct pillar (member of `Suspended-Pillar`) in a historical building that is temporary removed for restoration (hence, a relational instance of part-of in `Suspended-partOf`) and the building collapses before the pillar gets restored. An example where both part and whole remain active but only the part-whole relation is suspended can occur for, e.g., a car mechanic's database that records the cars and the parts that are under service for cleaning, i.e., temporarily removed, and the part needs to be re-inserted in the car it was removed from, such as a car ($o_w$) and the car engine ($o_p$) that is structurally part of the car. Thus, the suspension of a relation does not necessarily impose constraints on the permissible memberships of the part or whole in their respective status classes other than (RSusp2).

To structure and fully address all cases, one can—as a start—systematically apply suspension to the standard cases p6-p9 and w6-w9 as depicted in Figure 8-A and B, respectively. In addition, sp6′-sp9′ and sw6′-sw9′ in Figure 8-A and B denote the cases where, even though the whole (part) is suspended, the part (whole) still must remain active. To capture these cases (at least the non primed ones), we need to replace the (ConPo) axiom in both Theorem 1 and 2 with two additional axioms:

(SusP) $\texttt{Suspended-partOf} \sqsubseteq \texttt{part} : \texttt{Suspended-P}$ *Suspended Part*
(SusW) $\texttt{Suspended-partOf} \sqsubseteq \texttt{whole} : \texttt{Suspended-W}$ *Suspended Whole*

This change of axioms from (ConPO) to (SusP) and (SusW), however, does not immediately address our pillar and collapsed building example. That is, we know from the current axiomatization that when we have that at some time the pillar (part $o_p$) and its relation with the historical building ($o_w \in \texttt{W}$) become suspended ($o_p \in \texttt{Suspended-P}$ and $r \in \texttt{Suspended-partOf}$), which is a legal situation thanks to RSusp2 and Act, then when the whole becomes disabled due to the collapse ($o_w \in \texttt{Disabled-W}$), then so must $r \in$ `Disabled-partOf` instantaneously, because of RDisab4. To combine the appropriate set of constraints and formally prove it to be correct with respect to the intended semantics is a topic of current work.

**Explicit sharing of parts among wholes.** We now turn to explicit sharing of parts, where the main variations are depicted in Figure 8-C. Cases ps1-ps5 actually capture two possibilities: either $w_a, w_b \in W$ or $w_a \in W$, $w_b \in W'$ such that $W \neq W'$; henceforth, the latter will be identified with ps1′-ps5′. Observe that there is a principal distinction between parts that are shared "sequentially" and parts that are shared in "parallel". Examples of the former (ps1) could be the heart transplantation between humans $w_a$ and $w_b$, and an example for ps1′ a "multipurpose part" that is reused for another whole, e.g, a screw in a table that is used later for a bookshelf. The opposite case is where $p$ becomes part of $w_b$ as soon as it becomes active (ps4 and ps4′). An example of the latter, concurrent part-of (ps5′), is a seminar being part of both a seminar series and of a graduate course in a teaching database. An example of ps3 with the notion of one of the parthood relations, *contained_in*, is where the contained $p$ switches containers $w_a$ and $w_b$ back and forth or a administrative region in a war zone area (or simply a longer historical time interval) so that it alternately belongs to one country or another. Colloquially, one can reformulate the constraints of ps1, ps3, ps4, ps1′, ps3′, and ps4′ as the part being *sequentially* part of more than one whole in some way, and for ps2, ps5, ps2′, and ps5′ as where the part can be part of more than one whole *concurrently*.

To formally represent ps1-ps5 and ps1′-ps5′, we may not need additional axioms, but recombine in various ways the 12 listed above. For instance, ManW says only that the part must participate in the part-whole relation but does not have a range restriction at all, meaning it could a whole of any type $(W, W', \ldots)$; so for ps1, this means just mandatory participation by the part and if $w_a, w_b \in W$ then there is no mandatory constraint on the whole (because $w_b$ does not have to have the part), whereas if $w_a \in W$, $w_b \in W'$ and $W \neq W'$, then it may be the case that the whole must have a part, but one cannot know this *a priori*[9]. From a conceptual modeling perspective, however, this is undesirable, because one would want to be able to distinguish between the ps1-ps5 and ps1′-ps5′ series. To this end to be utterly explicit, we can add the following axiom.

$$(\text{DisjW})\ \mathtt{W_i} \sqsubseteq \sqcap_{\mathtt{j=i+1}}^{\mathtt{n}} \neg \mathtt{W_j} \quad \textit{Disjoint Wholes}$$

Given the full set of axioms, then ps1-ps5 and ps1′-ps5′ may be formally characterized and proven by taking different subsets of constraints. We have omitted them here due to their length and detail and for several cases it is not easy to find realistic examples. In addition, they obfuscate that, at least in some cases (ps1, ps4) we actually deal with a contracted version of the p1-p5 and w1-w5 cases in Figure 4. It may now be clear that although p2-p4 have the same ratio of the lines as w7-w9, the base axioms for the p-series is distinct from that of the w-series in Theorems 1 and 2 and that they do not necessarily involve essential or mandatoryness but can be optional parts/wholes. This is even more flexible with p1, p5, w1, and w5, where the part (whole) is contingently part of the whole (part), or: they are "independent parts" and "independent wholes" for which one can fix either only a minimal set of constraints where, temporally, almost anything is allowed or choose to be utterly specific so as to capture that and only that life cycle option. We are currently working on defining and proving the meaningful and realistic cases of the suggested options that are depicted informally in Figure 8-C.

Finally, in $\mathcal{ER}_{VT}$ and in $\mathcal{DLR}_{\mathcal{US}}$ we can specify cardinality constraints on the participation of classes into relations. This allows for expressing multiple sharing of parts/wholes like, for example, in specifying that cars must have exactly four wheels as parts:

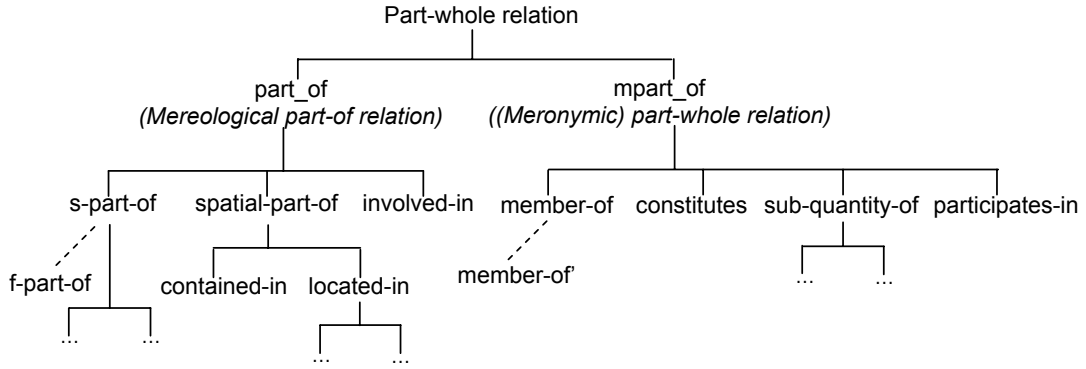


**Fig. 9.** Taxonomy of basic mereological and meronymic part-whole relations; the part-whole relations in the left-hand branch are all transitive, but those in the right-hand branch not necessarily. s-part-of = structural part-of; f-part-of = functional part-of. Dashed lines indicate that the subtype has additional constraints on the participation of the entity types; ellipses indicate several possible finer-grained extensions to the basic part-whole relations. (Source: Keet & Artale (2008))

$\texttt{partOf} \sqsubseteq \texttt{part:Wheel} \sqcap \texttt{whole:Car}$ (typing of the part-of relation)
$\texttt{Car} \sqsubseteq = 4\ \texttt{[whole]partOf.Car}$ (each car has exactly four wheels)
$\texttt{Wheel} \sqsubseteq \leq 1\ \texttt{[part]partOf.Car}$ (each wheel is part of max 1 car)

The constraints introduced in this section can represent all shareability constraints proposed earlier in the related literature (recollect section 2), meets the requirements as laid out in section 2.4, and refines shareability further with notions such as concurrently versus sequentially being part of a whole and temporary suspension of a part-whole relation.

### 4.4 Interaction with Types of Part-Whole Relations

Examples in the previous section for various cases of part-whole life cycles did mention different part-whole relations, such as (spatially) *contained_in*, structural parthood, and location. How these types of part-whole relations interact precisely with the life cycle semantics is an open question and in this section we only provide a flavor of the issues.

Summarizing the taxonomy of types of part-whole relations, we have a diagrammatic rendering in Figure 9 together with the formal characterization of the leaf types, where *part_of* is the parthood relation from Ground Mereology whereas *mpart_of* is neither transitive nor intransitive; refer to (Keet & Artale, 2008) for details on its rationale and the formal characterization, and Keet (2006b) for additional modeling guidelines.

Considering the possible interactions between the part-whole relations and shareability, one directly can note that if something is physically a *proper part* of a whole, such as that a car engine is a proper part of the car, then obviously, this proper part cannot physically be *directly* part of another whole at the same time, and likewise for its subtypes proper containment and proper location. Put differently, in those cases we must enforce, at least, the (ExLP) axiom. In contrast, a proper subprocess can be simultaneously *involved_in* (part of) several grander processes; e.g., a key chemical reaction intersecting in two metabolic pathways. Likewise, we can have, say, a musician $m$ who is concurrently *member_of* a string quartet and of the Royal Philharmonic Orchestra. The situation becomes more complicated with *subquantity_of* and so-called "portions of stuff". Provided one uses measurements for the quantities—say, a syringe full of dissolved morphine taken from the dissolved morphine stock in the bottle—then we can assert at the type level that the part-quantity must have its individual part-whole relation to the stock quantity as

either member of `Suspended-partOf` (in case the liquid in the syringe can be put back in the bottle) or of `Disabled-partOf`. To address such issues fully requires additional temporal constraints, which has been addressed only in part by Bittner & Donnelly (2007) (cf. section 2.4). Last, with the *constitutes* relation we have *no* sharing. These kind of interactions, however, merit further research and a precise characterization of constraints.

# 5 MODELING GUIDELINES

Clearly, while the formal characterization in the previous section provides precise semantics to shareability, one cannot burden the conceptual modeler, let alone the domain expert, with such details. The first step toward modeling guidelines is for the visually-oriented user: either present Figures 7-8 and one can directly point to the appropriate option or use their respective informal descriptions as initial modeling heuristic, such as

(EXAMPLE) Can an instance $p$ of **P** exist before some $w$ of type **W** it will become **part of**?
Yes ⇒ p1, p7, or p9
And can that instance $p$ exist after that $w$ is disabled/deleted?
Yes ⇒ p7
No ⇒ p1 or p9
That $w$ can outlive that $p$?
Yes ⇒ p1
No ⇒ p9

And so forth for the other six cases. The alternatives are a set of questions alike implemented in VisioModeler 3.1, but then tailored to shareability of part-whole relations, or a decision diagram alike proposed for choosing the appropriate type of part-whole relation (Keet, 2006b). We combine these two approaches. The first step is to differentiate between the basic options of essential part/whole, mandatory part/whole, and shareability, through posing a set of closed questions. The questions are formulated in such a way so as to be both uniform in sentence structure and to simplify further processing of the answers. Also, in a real conceptual model, `P` and `W` are replaced by their respective object types in the conceptual model.

(A1) Can an instance $p$ of **P** exist without some $w$ of type **W** it is **part of**?
(A2) Can an instance $p$ of **P** exist without the same $w$ of type **W** it is **part of**?
(A3) Can an instance $p$ of **P** be **part of** more than one whole $w$ at some time?
(B1) Can an instance $w$ of **W** exist when it does not **has part** some $p$ of type **P**?
(B2) Can an instance $w$ of **W** exist when it does not **has part** the same $p$ of type **P**?
(B3) Can an instance $w$ of **W has part** more than one part $p$ of type **P** at some time?

Figure 10 shows the resultant fact types for the six questions when answered with "no". We choose to depict them in ORM2 for three reasons. First, ORM2 is more expressive than either UML class diagrams or EER and ORM has an established transformation to these other conceptual data modeling languages (Halpin, 2001), the usability approach with questions-to-ORM diagram interaction is already an established practice, and it simplifies adding additional icons for the shareability semantics; the proposed icons are listed in Figure 11 and added to sample $\mathcal{ER}_{VT}$ diagrams in Figure 12.[10] *In casu*, for A2/B2, the current graphical ORM2 language is extended with a filled box in the role for essential part or whole (hence, note the different effects of the "some" and "same" in the questions and representation). On can, of course, combine the "no" answers to A1-B3,

two of which are depicted in Figure 10, too. Relating this back to the sets of constraints, then A1 corresponds to a simple ManW constraint and A2+B2 combines the constraints of p6+w6 (see Theorems 1 and 2).

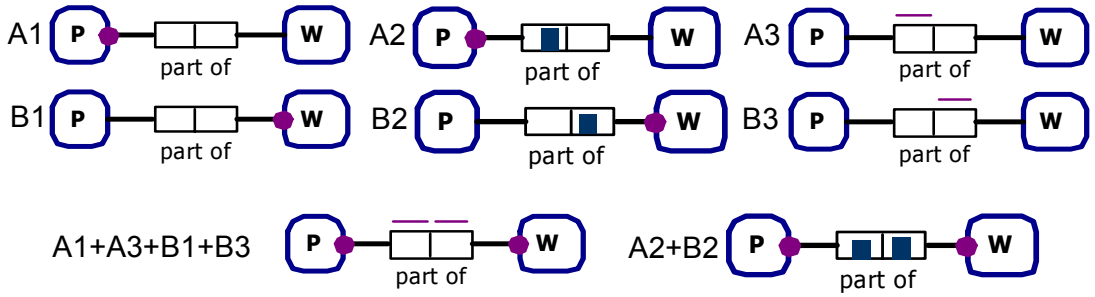


**Fig. 10.** Representations resulting from the answers to questions A1-B3 when answered with "no". Regarding semantics, A2 and B2 have an additional icon (rectangle in the role) to denote essential part/whole; see also Figure 11.

Given that there is a whole list of questions, one can build in intermediate feedback loops, such as asking the modeler after all "yes" on A1-B3:

(FEEDBACK) With a "yes" on A1-B3, either the part or the whole, or both, can be shared. Is that true?

If the answer is, "no", then A1-B3 should be revisited; if the answer is in the affirmative, the modeler can proceed to the second set of questions. The second step is assessment of the sharing of parts, where we can reuse the answers on the previous five questions. For instance, when we have a mandatory but not essential participation on the part-side—a "no" for A1—it is obvious that the part cannot exist independently; i.e., then we must have any of the options p2, p3, or p4 or ps1-ps5′ so that asking questions to cover the remaining options has become irrelevant. With the C1-C5 question series, one can extract from a domain expert if the sharing can/must be sequential or in parallel and if the wholes may be of a different type or not; this selection procedure is depicted in Figure 13 with additional explanatory notes that a CASE tool developer might want to include as an extra service.

(C1) Can an instance $p$ of **P** be **part of** more than one whole $w$ at the same time?
(C2) Regarding the wholes $w_1, ..., w_n$ that $p$ can be **part of**, must $w_1, ..., w_n$ be instances of the same type **W**?
(C3) Can an instance $p$ of **P** also be **part of** only one whole $w$ of type **W**?
(C4) Can an instance $p$ of **P** only become **part of** another whole $w_2$ after whole $w_1$ cease to be active as whole?
(C5) Will an instance $p$ of **P** become **part of** another whole $w_2$ and cease to be part of $w_1$ as soon as $w_2$ becomes active as whole?

A sample diagrammatic representation for ps1′-ps5′ is included in Figure 14; UML and EER are currently less fine-grained and less expressive, but with a $\mathcal{DLR}$ in the background in the CASE tool, this can be added trivially, see e.g. the Icom tool (Fillottrani *et al.*, 2006). Formulating the same series for the perspective of the whole—starting with the "yes" on B3—is left as an exercise to the reader. Third, we add a further dimension: suspension of the part-whole relation or of the parts or wholes. By default, the part-whole relation can be suspended, except where it is explicitly disallowed, i.e. for those cases that include ConPO. To ascertain this, one has to ask at least the following questions.

(D1) Can the **part of** relation be suspended?
If "no" then we have any of the cases that have ConPO; if yes, the constraints for the cases cannot include ConPO.

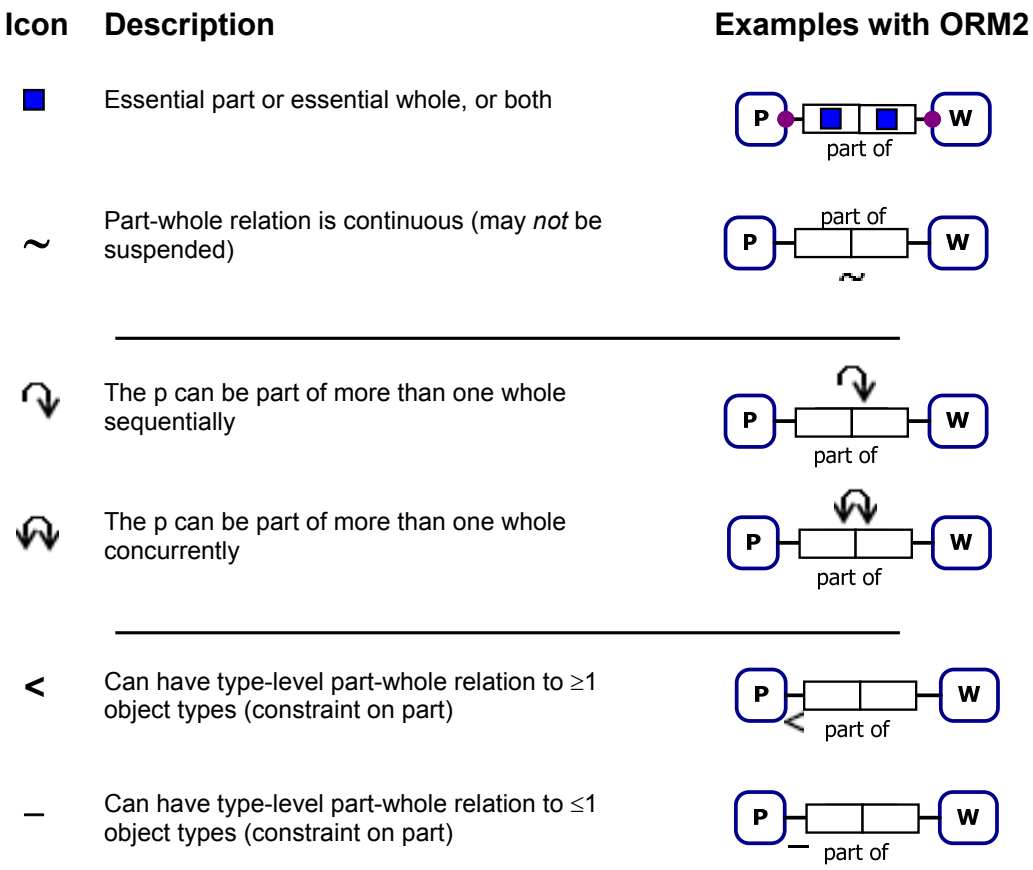


**Fig. 11.** Suggested icons to denote the various aspects of shareability of parts and of the part-whole relation. Example use is demonstrated for ORM2 notation, but also can be added to EER or UML's association relation and classes. Note that ■ and ∼ are new additions for all modeling languages, the arrows make explicit certain temporal behavior, whereas < and − are mainly useful in the light of further model development.

(D2) Can an instance $p$ of **P** that is **part of** $w$ of **W** become suspended?
If "No" then **P** must have a (strong) essential whole. If "yes" then **P** can have a weak essential whole or other shareability options (depending on the answers of prior questions).

(D3) If this $p$ of **P** cannot be suspended, can the $w$ of **W** it is **part of** become suspended?
If "no" then $p$ has an essential whole (w6-w9), if "yes" then $p$ has a strong essential whole (sw6'-sw9').

One could add cross-checks to prevent violation of the constraints, as has been proposed by Motschnig-Pitrik & Kaasbøll (1999) in a modeling guideline mode. These are, however, already covered by the current decision diagram and specific set of constraints for each case and a simple check (set comparison) can be implemented where the manually modeled constraints are compared with the combinations of constraints from the previous section.

At the time of writing, however, question series C1-C5 and D1-D3 are for indicative purpose to gain insight into what actually are the salient cases that recur during the modeling process and which scenarios are more prevalent in a subject domain. Thus, they aid in stimulating domain experts and modelers to assess in more detail which life cycle semantics are more important and realistic, as opposed to providing a forest of constraints where possibly only a third or even less is practically used. After scoping relevance, one then could choose to implement only the useful subset of part-whole shareability.

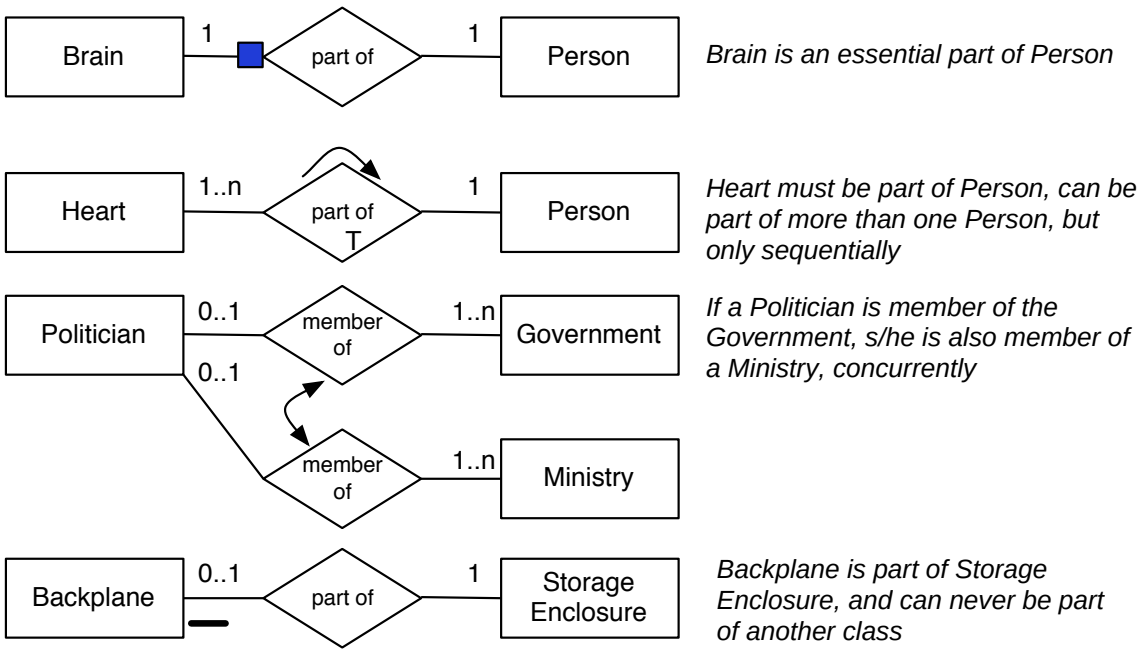


**Fig. 12.** Several examples of the suggested icons in conjunction with $\mathcal{ER}_{VT}$, for conceptual models that are ontology-inspired, intended for organ transplant databases, government administration software, and a hardware manufacturing database, respectively. (That is, it could be modeled differently for different application software, such as permitting coalition governments to create ministerial posts in name only, i.e., without a portfolio and ministry, to please a coalition partner so that the double arrow should be removed). The "T" in the part-of relation between Heart and Person makes explicit it is a temporal relation, as defined in $\mathcal{ER}_{VT}$ (Definition 1).

# 6 FUTURE TRENDS

As the reader may have noticed from section 2, there is much recent research activity on part-whole relations. From one viewpoint, this is good because significant improvements in understanding of part-whole relations are being achieved. From a practitioner's viewpoint, however, this also means that most of these advances have not yet made it into the readily available CASE tools and are thereby primarily of direct use as 'paper exercise' in the analysis and modeling steps of the software development process. Considering some general requirements to implement support for various aspects of part-whole relations, an overall structure and implementation would have to meet several requirements such as to:

i. Ensure the representation is such that one can distinguish between parthood relations of a class (or its instances) and other generic properties (/relations/roles/associations), i.e. to make part-whole and whole-part relations first-class citizens;
ii. Identify unambiguously and model a 'minimal amount' of part-whole relations in the conceptual data models; sub-requirements (Keet & Artale, 2008) comprise representing at least the simplest parthood theory Ground Mereology (Varzi, 2004), expressing ontological categories and their taxonomic relations, having the option to represent transitive and intransitive relations, and to be able to specify the domain and range restrictions (/relata/entity types) for the classes participating in a relation;
iii. Provide a set of combinations of essential, uniqueness, and mandatoryness constraints applicable to the relation;
iv. Clarify and accommodate for other, sometimes called "secondary", properties of part-whole relations, such as functional dependence and completeness;
v. Ensure the inverse, has-part, relation is properly modeled as well;

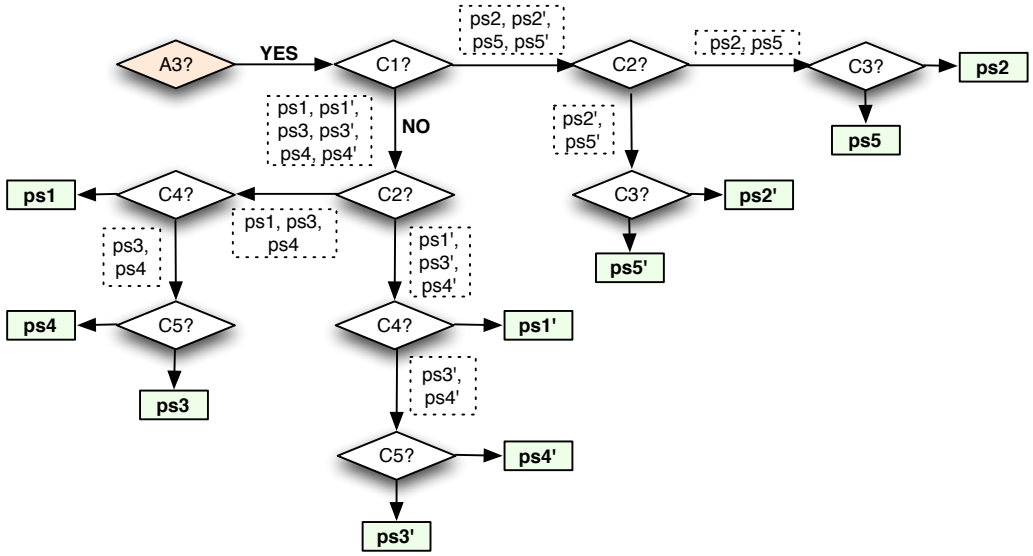


**Fig. 13.** Decision diagram to assess on the appropriate type of shareability; see text for the complete questions in the decision diamonds.

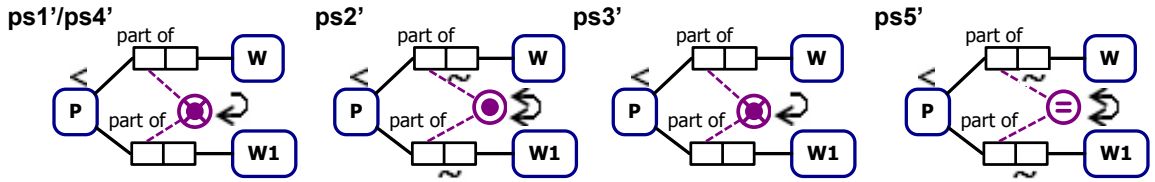


**Fig. 14.** Graphical rendering of constraints ps1′-ps′ in ORM2. Note that the double arrows in ps2′ and ps5′ are redundant in ORM because they can be captured with the regular constraints already, but are required for UML class diagrams and ER. Optionally, one could add yet another new icon so as to have the distinction between ps1′ and ps4′. The $<$ on $P$ is obviously satisfied in the figure, but in the light of conceptual model evolution, useful to make explicit.

vi. Transitivity of parthood relations is enabled where applicable and prohibited for non-transitive part-whole relations;
vii. Address the possibilities and consequences of horizontal interrelations between the parts of a whole;
viii. Develop an underlying unifying paradigm that relates the conceptual modeling language specific constructors, if possible.

At the time of writing, we are far away from meeting all these requirements. Point i is met only informally with UML's aggregation relation. Point ii can be met partially with the taxonomic structure as presented in (Keet, 2006b; Keet & Artale, 2008), and UML's stereotypes and/or conceptual model meta-modeling. So-called secondary properties, inverse relations, transitivity, antisymmetry, and horizontal relations are only to a limited extent possible in some languages (extensions of UML, ORM's ring constraints, but no DL has a constructor for antisymmetry), which could be extended and harmonized further. Full computational support not only for computer-aided conceptual modeling with the guidelines but also including automated satisfiability and consistency checking of a conceptual model, is likely to be difficult due to undecidability—e.g., $\mathcal{DLR}_{\mathcal{US}}$ with time stamping for relations is undecidable whereas without the option to represent temporal persistence of $n$-ary relations, reasoning in $\mathcal{DLR}_{\mathcal{US}}$ is an ExpTime-complete problem (Artale *et al.*, 2002). Dropping the evolution constraints from $\mathcal{DLR}_{\mathcal{US}}$ allows timestamping on relations and is decidable (Artale *et al.*, 2007b). On the other hand, conceptual

modeling emphasizes expressiveness, not computability; constraints can be dropped in a design-level specification and then one at least knows what has been removed and why.

**Open problems.** To give an indication of open problems on part-whole relations, i.e., transforming several requirements from the previous subsection into research and engineering questions, we outline a non-exhaustive list of avenues.

- ⋆ DL: Which subtheory of mereology fits best with any of the extant DLs? Can one of the DL languages be extended, and if so, how and what about its complexity? What about property inheritance across the parthood relation? What are the difference between the intensional and extensional reasoning (behavior of the parthood relation at the TBox and ABox, respectively)?
- ⋆ ER and ORM: what and how to add part-whole relations? What about developing more expressive versions of ER and ORM that include the parthood relation, alike the DL languages with difference in expressiveness and complexity? How to make it usable for the modeler?
- ⋆ Applied parthood relations: what lessons can be learned from practical use of part-whole relations in specific subject domains such as bio-ontologies and geographical information systems? If any, can this be fed back into mereology to extend mereological theories? Does usage of the part-whole relation across subject domains reveal domain specific intricacies that cannot be generalised to domain-independent characteristics?

Notwithstanding these gaps, many aspects of the part-whole relations can be modeled with extant conceptual modeling languages or require only minor extensions. This situation changes if one were to require efficient reasoning over conceptual data models, but as long as the focus is on expressiveness to enable the subject domain as good as possible, then this poses no problem.

# 7 CONCLUSIONS

The main questions addressed in this chapter were: which type of shareability and which lifetime aspects are possible, what is the formal semantics for sharability, and how to model these kind of differences? In order to solve these issues, we merged and extended advances in representing part-whole relations in UML class diagrams, EER, ORM, and DL languages with formal conceptual data modeling for temporal databases (temporal EER) and ORM's usability features. First, the different semantics of part-whole shareability notions were formally characterized by availing of the temporal Description Logic $\mathcal{DLR}_{\mathcal{US}}$, which was hitherto only used for temporal EER ($\mathcal{ER}_{VT}$), and having extended it with the original notion of *status relations* so as to capture unambiguously and at the conceptual layer the different life cycle semantics of parts and wholes and what happens to the part-whole relation. These formally defined constraints can represent all shareability constraints proposed earlier in the related literature, meets the requirements as laid out in section 2.4, and refines shareability further with notions such as concurrently versus sequentially being part of a whole and temporary suspension of a part-whole relation. By having used $\mathcal{DLR}_{\mathcal{US}}$ as foundational mechanism to represent the different shareability semantics, the results obtained are easily transferrable to UML class diagrams and ORM/ORM2. Second, the shareability options were transformed into three complimentary conceptual modeling guidelines: visuals, a simple list of questions, and a decision diagram so as to easily navigate to the appropriate constraints and their shareability cases of the relations and its participating parts and wholes.

Several issues, however, have not been addressed, both from the theoretical and practical side. These include, but are not limited to, across-time part-whole relations, their precise interaction with various types of part-whole relations, and software support to simplify modeling of part-whole relations.

# Bibliography


Albert, M., Pelechano, V., Fons, J., Ruiz, M. & Pastor, O. (2003). Implementing UML Association, Aggregation, and Composition. A Particular Interpretation Based on a Multidimensional Framework. In Eder, J. and Missikoff, M. (ed.), *Proceedings of CAiSE'03 LNCS Vol 2681* (pp. 143-158). Berlin: Springer Verlag.

Álvarez, A.T. & Alemán, J.L.F. (2000). Formally modeling UML and its evolution: a holistic approach. In *Fourth International Conference on Formal methods for open object-based distributed systems IV* (pp. 183-206). Amsterdam: Kluwer Academic Publishers.

Artale, A., Calvanese, D., Kontchakov, R., Ryzhikov, V. & Zakharyaschev, M. (2007a). Complexity of Reasoning over Entity Relationship Models. *Proc. of DL-07, CEUR WS Vol 250*.

Artale, A., Franconi, E., Guarino, N. & Pazzi, L. (1996a). Part-Whole Relations in Object-Centered Systems: an Overview. *Data and Knowledge Engineering*, *20*(3), 347-383.

Artale, A., Franconi, E. & Guarino, N. (1996b). Open Problems for Part-Whole Relations. In: *Proceedings of 1996 International Workshop on Description Logics (DL-96)* (pp 70-73). Cambridge, MA: AAAI Press.

Artale, A. & Franconi, E. (1999). Temporal ER modeling with description logics. In *Proc. of the Int. Conf. on Conceptual Modeling (ER'99)*. Berlin: Springer-Verlag.

Artale, A., Franconi, E. & Mandreoli, F. (2003). Description Logics for Modelling Dynamic Information. In Chomicki, J., van der Meyden, R. & Saake, G. (eds.), *Logics for Emerging Applications of Databases*. Berlin: Springer-Verlag.

Artale, A., Franconi, E., Wolter, F. & Zakharyaschev, M. (2002). A temporal description logic for reasoning about conceptual schemas and queries. In S. Flesca, S. Greco, N. Leone, G. Ianni (eds.), *Proceedings of the 8th Joint European Conference on Logics in Artificial Intelligence (JELIA-02), LNAI vol 2424* (pp. 98-110). Berlin: Springer Verlag.

Artale, A., Guarino, N., & Keet, C.M. (2008). Formalising temporal constraints on part-whole relations. In Brewka, G, Lang, J (Eds.), *11th International Conference on Principles of Knowledge Representation and Reasoning (KR'08)*. Cambridge, MA: AAAI Press.

Artale, A. & Keet, C.M. (2008). Essential and Mandatory Part-Whole Relations in Conceptual Data Models. *Proceedings of the 21st International Workshop on Description Logics (DL'08), CEUR WS Vol 353*, Dresden, Germany, 13-16 May 2008.

Artale, A., Parent, C. & Spaccapietra, S. (2006). Modeling the evolution of objects in temporal information systems. In: *4th International Symposium on Foundations of Information and Knowledge Systems (FoIKS-06), LNCS vol 3861 (pp. 22-42)*. Berlin: Springer-Verlag.

Artale, A., Parent, C. & Spaccapietra, S. (2007b). Evolving objects in temporal information systems. *Annals of Mathematics and Artificial Intelligence (AMAI)*, *50*(1-2), 5-38.

Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D. & Patel-Schneider, P.F. (eds). (2003). *Description Logics Handbook*. Cambridge: Cambridge University Press.

Barbier, F., Henderson-Sellers, B., Le Parc-Lacayrelle, A. & Bruel, J.-M. (2003). Formalization of the whole-part relationship in the Unified Modelling Language. *IEEE Transactions on Software Engineering*, *29*(5), 459-470.

Berardi, D., Calvanese, D. & De Giacomo, G. (2005). Reasoning on UML class diagrams. *Artificial Intelligence*, *168*(1-2), 70-118.

Bittner, T. & Donnelly, M. (2005). Computational ontologies of parthood, componenthood, and containment, In Kaelbling, L. (ed.) *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence 2005 (IJCAI05)* (pp. 382-387). Cambridge, MA: AAAI Press.

Bittner, T. & Donnelly, M. (2007). A temporal mereology for distinguishing between integral objects and portions of stuff. In *Proceedings of the Twenty-second AAAI Conference on Artificial intelligence (AAAI'07)* (pp. 287-292). Cambridge, MA: AAAI Press.

Borgo, S. & Masolo, C. (in press). Full mereogeometries. *Journal of Philosophical Logic.*

Calvanese, D. & De Giacomo, G. (2003). Expressive description logics. In Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P. (Eds) *The Description Logic Handbook: Theory, Implementation and Applications* (pp. 178-218). Cambridge: Cambridge University Press.

Calvanese, C., De Giacomo, G. & Lenzerini, M. (1998a). On the decidability of query containment under constraints. In: *Proceedings of the 17th ACM SIGACT SIGMOD SIGART Symposium on Principles of Database Systems (PODS'98)* (pp. 149-158).

Calvanese, D., Lenzerini, M. & Nardi, D. (1998b). Description logics for conceptual data modeling. In J. Chomicki and G. Saake, editors, *Logics for Databases and Information Systems.* Amsterdam: Kluwer.

Calvanese, D., Lenzerini, M. & Nardi, D. (1999). Unifying class-based representation formalisms. *Journal of Artificial Intelligence Research*, *11*, 199-240.

Chomicki, J. & Toman, D. (1998). Temporal logic in information systems. In J. Chomiki and G. Saake (Eds.). *Logics for databases and information systems.* Amsterdam: Kluwer.

Etzion, O., Gal, A., & Segev, A. (1998). Extended update functionality in temporal databases. In O. Etzion, S. Jajodia, and S. Sripada (eds) *Temporal Databases - Research and Practice LNCS* (pp 56-95). Berlin: Springer-Verlag.

Fillottrani, P., Franconi, E. & Tessaris, S. (2006). The new ICOM ontology editor. In *19th International Workshop on Description Logics (DL 2006)*, Lake District, UK. May 2006.

Franconi, E. & Ng, G. (2000). The iCom tool for intelligent conceptual modeling. *7th International Workshop on Knowledge Representation meets Databases (KRDB'00)*, Berlin, Germany. 2000.

Gerstl, P. & Pribbenow, S. (1995). Midwinters, end games, and body parts: a classification of part-whole relations. *International Journal of Human-Computer Studies*, *43*, 865-889.

Guarino, N., and Welty, C. (2000). A formal ontology of properties. In Dieng, R., ed., *Proceedings of EKAW '00*, LNCS. Berlin: Springer Verlag.

Guizzardi, G. (2005). *Ontological foundations for structural conceptual models.* PhD Thesis, Telematica Institute, Twente University, Enschede, the Netherlands.

Guizzardi, G. (2007). Modal Aspects of Object Types and Part-Whole Relations and the de re/de dicto distinction. *19th International Conference on Advances in Information Systems Engineering (CAiSE) LNCS 4495.* Berlin: Springer-Verlag.

Halpin, T. (1999). UML Data Models from an ORM Perspective (Part 8). *Journal of Conceptual Modeling*, Issue 8, April 1999. Stable URL http://www.inceoncept.com/jcm.

Halpin, T. (2001). *Information Modeling and Relational Databases.* San Francisco: Morgan Kaufmann Publishers.

Halpin. T. (2007). Subtyping revisited. In Pernici, B., Gulla, J. (eds.) *Proceedings of CAiSE'07 Workshops* (pp. 131-141). Academic Press.

Hawley, K. (2004). Temporal Parts. In Zalta, E.N. (ed.) *The Stanford Encyclopedia of Philosophy (Winter 2004 Ed.).* Stable URL http://plato.stanford.edu/archives/win2004/entries/temporal-parts/.

Hodgkinson, I.M., Wolter, F. & Zakharyaschev, M. (2000). Decidable fragments of first-order temporal logics. *Annals of pure and applied logic*, *106*, 85-134.

Horrocks, I., Kutz, O. & Sattler, U. (2006). The Even More Irresistible $\mathcal{SROIQ}$. In: *Proceedings of the 10th International Conference of Knowledge Representation and Reasoning (KR-2006)*, Lake District, UK, 2006.

Johansson, I. (2004). On the transitivity of the parthood relation. In Hochberg, H. and Mulligan, K. (eds.) *Relations and predicates* (pp. 161-181). Frankfurt: Ontos Verlag.

Keet, C.M. (2006a). *Introduction to part-whole relations: mereology, conceptual modeling and mathematical aspects* (Tech. Rep. No. KRDB06-3). KRDB Research Centre, Faculty of Computer Science, Free University of Bozen-Bolzano, Italy.

Keet, C.M. (2006b). Part-whole relations in Object-Role Models. *2nd International Workshop on Object-Role Modelling (ORM 2006)*, Montpellier, France, Nov 2-3, 2006. In Meersman, R., Tari, Z., Herrero, P. et al. (Eds.) *OTM Workshops 2006 LNCS Vol 4278* (pp. 1116-1127). Berlin: Springer-Verlag.

Keet, C.M. (2007). Prospects for and issues with mapping the Object-Role Modeling language into $\mathcal{DLR}_{ifd}$. *20th International Workshop on Description Logics (DL'07) CEUR-WS Vol 250* (pp. 331-338) 8-10 June 2007, Bressanone, Italy.

Keet, C.M. (2008). A formal comparison of conceptual data modeling languages. *13th International Workshop on Exploring Modeling Methods in Systems Analysis and Design (EMMSAD'08) CEUR-WS Vol 337* (pp. 25-39). Montpellier, France, 16-17 June 2008.

Keet, C.M. & Artale, A. (2007). Representing and Reasoning over a Taxonomy of Part-Whole Relations. *Applied Ontology – Special Issue on Ontological Foundations for Conceptual Models*, *3*(1), in print.

Masolo, C., Borgo, S., Gangemi, A., Guarino, N. & Oltramari, A. (2003). *Ontology Library*. WonderWeb Deliverable D18 (ver. 1.0, 31-12-2003). http://wonderweb.semanticweb.org. 2003.

Motschnig-Pitrik, R. & Kaasbøll, J. (1999). Part-Whole Relationship Categories and Their Application in Object-Oriented Analysis. *IEEE Transactions on Knowledge and Data Engineering*, *11*(5), 779-797.

Object Management Group. *Unified Modeling Language: Superstructure*. v2.0. formal/05-07-04. http://www.omg.org/cgi-bin/doc?formal/05-07-04.

Odell, J.J. (1998). *Advanced Object-Oriented Analysis & Design using UML*. Cambridge: Cambridge University Press.

Opdahl, A.L., Henderson-Sellers, B., Barbier, F. (2001). Ontological analysis of whole-part relationships in OO-models. *Information and Software Technology*, *43*(6), 387-399.

Parent, C., Spaccapietra, S. & Zimányi, E. (2006). *Conceptual modeling for traditional and spatio-temporal applications—the MADS approach*. Berlin: Springer Verlag.

Pontow, C. & Schubert, R. (2006). A mathematical analysis of theories of parthood. *Data & Knowledge Engineering*, *59*, 107-138.

Sattler, U. (1995). A concept language for an engineering application with part-whole relations. In Borgida, A., Lenzerini, M., Nardi, D., Nebel, B. (Eds.) *Proceedings of the international workshop on description logics (DL'95)* (pp. 119-123).

Sattler, U. (2000). Description Logics for the Representation of Aggregated Objects. In Horn, W. (Ed.) *Proceedings of the 14th European Conference on Artificial Intelligence (ECAI2000)*. Amsterdam: IOS Press.

Schulz, S., Hahn, U. & Romacker, M. (2000). Modeling Anatomical Spatial Relations with Description Logics.In Overhage, J.M. (ed.) *Proceedings of the AMIA Symposium 2000* (pp. 779-83).

Shanks, G., Tansley, E. & Weber, R. (2004). Representing composites in conceptual modeling. *Communications of the ACM*, *47*(7), 77-80.

Simons, P. (1987). *Parts: a study in Ontology*. Oxford: Clarendon Press.

Smith, B., Ceusters, W., Klagges, B., Köhler, J., Kumar, A., Lomax, J., Mungall, C., Neuhaus, F., Rector, A.L. & Rosse, C. (2005). Relations in biomedical ontologies. *Genome Biology*, *6*:R46.

Spaccapietra, S., Parent, C. & Zimanyi, E. (1998). Modeling time from a conceptual perspective. In *Int. Conf. on Information and Knowledge Management (CIKM98)*.

Varzi, A.C. (2004). Mereology. In Zalta, E.N. (ed.) *The Stanford Encyclopedia of Philosophy (Fall 2004 Ed.)*. Stable URL http://plato.stanford.edu/archives/fall2004/entries/mereology/.

Varzi, A.C. (2006a). Spatial reasoning and ontology: parts, wholes, and locations. In Aiello, M., Pratt-Hartmann I., van Benthem J. (eds.) *The Logic of Space*. Dordrecht: Kluwer Academic Publishers.

Varzi, A.C. (2006b). A Note on the Transitivity of Parthood. *Applied Ontology*, *1*, 141-146.

Vieu, L. & Aurnague, M. (2005). Part-of Relations, Functionality and Dependence. In M. Aurnague, M. Hickmann, L. Vieu (eds.) *Categorization of Spatial Entities in Language and Cognition*. Amsterdam: John Benjamins.

Winston, M.E., Chaffin, R. & Herrmann, D. (1987). A taxonomy of partwhole relations. *Cognitive Science*, *11*(4), 417-444.

## Notes

[1]Other recurring topics that will receive comparatively little attention are transitivity of part-whole relations (Johansson, 2004; Varzi, 2004, 2006b), analysis of types of part-whole relations (Gerstl & Pribbenow, 1995; Keet, 2006b; Keet & Artale, 2008; Odell, 1998; Winston *et al.*, 1987), horizontal relations between parts, and automated reasoning with part-whole relations. For a comprehensive introduction to such sub-topics from different perspectives, see, e.g., (Artale *et al.*, 1996a; Guizzardi, 2005; Keet, 2006a; Simons, 1987).

[2]proper part hood is usually defined in terms of the parthood relation ($\forall x, y(proper\_part\_of(x,y) \triangleq part\_of(x,y) \wedge \neg part\_of(y,x))$), but also can be taken as primitive and then to have parthood defined in terms of proper parthood ($\forall x, y(part\_of(x,y) \triangleq proper\_part\_of(x,y) \vee x = y)$).

[3]Downward distributive: there are properties of the whole that the parts inherit; upward distributive: the whole inherits properties from its parts. Alternatively, they are called property inheritance through parts and property refinement through parts.

[4]In addition to these requirements, the usual ones for including basic properties of the parthood relation remain as well. That is, transitivity, reflexivity and antisymmetry for parthood, and transitivity, irreflexivity and asymmetry for proper parthood relations (see for a discussion and feasibility Keet & Artale (2008)).

[5]Temporal parts in the sense of 4-dimensionalism (Hawley, 2004) is out of scope for common information systems modeling.

[6]More precisely, UML without the part-of, EER, and a subset of ORM and ORM2 have a correspondence with $\mathcal{DLR}_{ifd}$ (Berardi *et al.*, 2005; Calvanese *et al.*, 1998a, 1999; Keet, 2007, 2008), which is $\mathcal{DLR}$ with additional identification and non-unary functional dependency constraints.

[7]Following the snapshot paradigm, $\mathcal{T}_p$ is a set of time points (or chronons) and $<$ is a binary precedence relation on $\mathcal{T}_p$, the flow of time $\mathcal{T} = \langle \mathcal{T}_p, < \rangle$ is assumed to be isomorphic to either $\langle \mathbb{Z}, < \rangle$ or $\langle \mathbb{N}, < \rangle$. Thus, standard relational databases can be regarded as the result of mapping a temporal database from time points in $\mathcal{T}$ to atemporal constructors, with the same interpretation of constants and the same domain.

[8]Cases p1-p5 shown earlier in Figure 4 are not essential parts/wholes; we return to this point at the end of this section.

[9]Optional participation ($\geq 0$) of either the part or the whole in the part-whole relation means that either (ManP) or (ManW) is not included in the list of constraints.

[10]We are open to better suggestions (the icons have not been examined by modelers and domain experts). Basically, for each constraint ManP-DisjW there could be some explicit on in the diagrammatic modeling language, even though some are already covered in the language itself, such as XOR in ORM2. It is beyond the current scope to provide a fixed graphical and pseudo-NL syntax for all major conceptual modeling languages.