A formal comparison of conceptual data modeling languages

C. Maria Keet

Faculty of Computer Science, Free University of Bozen-Bolzano, Italy
keet@inf.unibz.it

Abstract. An essential aspect of conceptual data modeling methodologies is the language's expressiveness so as to represent the subject domain as precise as possible to obtain good quality models and, consequently, software. To gain better insight in the characteristics of the main conceptual modeling languages, we conducted a comparison between ORM, ORM2, UML, ER, and EER with the aid of Description Logic languages of the $\mathcal{DLR}$ family and the new formally defined generic conceptual data modeling language $\mathcal{CM}_{com}$ that is based on $\mathcal{DLR}_{\mathit{ifd}}$. ORM, ER, EER, and UML class diagrams are proper fragments of ORM2 and $\mathcal{CM}_{com}$ has the most expressive common denominator with these languages. $\mathcal{CM}_{com}$ simplifies prospects for automated, online, interoperability among the considered languages so that modelers not only can continue using their preferred modeling language yet be compatible with the other ones, but also have a common ground that eases database and software integration based on commonly used conceptual data models.

1 Introduction

Different conceptual data modeling languages exist for a plethora of reasons. For object-oriented modeling, the main commonly used conceptual modeling languages are (E)ER, UML class diagrams, and ORM(2) that come in various, mostly graphical, notational variants. They, of course, share a common core, but also cater for specifics with the type applications in mind. Several how-to guides for transformations and metamodel comparisons between these, and other, conceptual data modeling languages have been made, most notably [23, 24], but they focus on mappings of the respective graphical elements. Hence, for each new notation—e.g., Barker ER, IE, and IDEF1X [27]—a new mapping scheme has to be identified, resulting in an m:n mesh with $(k-1)k/2$ required mappings among k languages that also have to be maintained, not to mention their implementations in CASE tools and diagram drawing software. A different approach that avoids this problem has been taken by [12, 14, 15] who aim to unify class-based modeling languages through the $\mathcal{DLR}$ family of Description Logic (DL) languages. They focus on information integration, but also define precise model-theoretic semantics for conceptual modeling languages. This approach also opens the avenue for *formal* 1:n mappings based on the language constructs irrespective of the variations in graphical elements, because if one can settle on a comparatively optimal DL language, then this language could function as the common foundation for all graphical variants and even fixed-syntax natural language versions. However, this has been worked out only for restricted versions of ER and frame-based systems, but not full EER and UML class diagrams or ORM/ORM2. In addition, in the meantime more expressive flavours of $\mathcal{DLR}$ have been investigated so as to maximize expressiveness of the language yet stay within a decidable fragment of first order logic. Advantages of the latter are to enable computational support for both integration of conceptual data models and support automated satisfiability and consistency checking to detect modeling errors and derive implicit information so as to enhance the quality of conceptual models [18, 19, 28]. We extend and refine Calvanese et al's [12, 14, 15] approach by, first, integrating previously obtained results regarding advances on and mappings between conceptual modelling languages and characteristics of the corresponding DL languages and, second, taking into account standardized (UML, IDEF1X) and semi-standardized (Barker ER, IE, ORM, ORM2) conceptual data modeling languages as well as their implementations in modeling tools such as VisioModeler, NORMA, CaseTalk, RationalRose, VP-UML, OmniGraffle, and SmartDraw [2–4, 8–14, 16, 18–23, 25–30, 32, 33]. We identify their greatest common denominator, $\mathcal{DLR}_{\mathit{ifd}}$, which will be used to formally define the generic common conceptual data modeling language $\mathcal{CM}_{com}$ that thus has an equi-satisfiable $\mathcal{DLR}_{\mathit{ifd}}$ knowledge base. Second, this $\mathcal{CM}_{com}$ is used to compare ER, EER, UML class diagrams, ORM, and ORM2, where it will be shown in an unambiguous way that UML, ER, EER, and ORM are different proper fragments of ORM2. The results we thus obtain with $\mathcal{CM}_{com}$ simplify computational implementations for interoperability of conceptual data models modeled in different graphical languages; hence, being compatible with established modeling languages and practices yet moving forward toward realization of a logic-based, computer-aided, and conceptual modeling-based information integration framework.

In the remainder of the paper, we first deal with the theoretical background of the main Description Logic languages tailored to formal conceptual modelling (section 2) and introduce the $\mathcal{CM}_{com}$ syntax and semantics. Subsequently, ER, EER, UML class diagrams, ORM, and ORM2 are defined and compared in section 3. We conclude in section 4.

2 Theoretical background

2.1 Description Logics and the $\mathcal{DLR}$ family of languages

Description Logics (DL) languages are decidable fragments of first order logic and are used for logic-based knowledge representation, such as conceptual modelling and ontology development. The basic ingredients of all DL languages are *concepts* and *roles*, where a DL-role is an n-ary predicate ($n \geq 2$). In addition, a DL language has several constructs, thereby giving greater or lesser expressivity and efficiency of automated reasoning over a logical theory. DL knowledge bases are composed of the *Terminological Box* (TBox), which contains axioms at the

Table 1. Semantics of $\mathcal{DLR}$ and $\mathcal{DLR}_{\mathit{ifd}}$.

$\top_n^{\mathcal{I}} \subseteq (\Delta^{\mathcal{I}})^n$	$A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
$\mathbf{P}^{\mathcal{I}} \subseteq \top_n^{\mathcal{I}}$	$(\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
$(\neg \mathbf{R})^{\mathcal{I}} = \top_n^{\mathcal{I}} \setminus \mathbf{R}^{\mathcal{I}}$	$(C_1 \sqcap C_2)^{\mathcal{I}} = C_1^{\mathcal{I}} \cap C_2^{\mathcal{I}}$
$(\mathbf{R}_1 \sqcap \mathbf{R}_2)^{\mathcal{I}} = \mathbf{R}_1^{\mathcal{I}} \cap \mathbf{R}_2^{\mathcal{I}}$	$(\$i/n : C)^{\mathcal{I}} = \{(d_1, ..., d_n) \in \top_n^{\mathcal{I}} \mid d_i \in C^{\mathcal{I}}\}$
$\top_1^{\mathcal{I}} = \Delta^{\mathcal{I}}$	$(\exists[\$i]\mathbf{R})^{\mathcal{I}} = \{d \in \Delta^{\mathcal{I}} \mid \exists(d_1, ..., d_n) \in \mathbf{R}^{\mathcal{I}}.d_i = d\}$
	$(\leq k[\$i]\mathbf{R})^{\mathcal{I}} = \{d \in \Delta^{\mathcal{I}} \mid \lvert\{(d_1, ..., d_n) \in \mathbf{R}_1^{\mathcal{I}} \mid d_i = d\}\rvert \leq k\}$

concept-level, and the *Assertional Box* (ABox) that contains assertions about instances. A TBox corresponds to a formal conceptual data model or can be used to represent a type-level ontology; refer to [6] for more information about DLs and their usages.

For formal conceptual data modelling, we introduce $\mathcal{DLR}$ first [9]. This DL language was specifically developed to provide a formal characterization of conceptual modelling languages to (i) enable automated reasoning over the conceptual data models to improve their quality and that of the resulting software application, and (ii) to use it as unifying paradigm for database integration through integrating their respective conceptual models [12, 14]. Take atomic relations ($\mathbf{P}$) and atomic concepts A as the basic elements of $\mathcal{DLR}$, which allows us to construct arbitrary relations (arity ≥ 2) and arbitrary concepts according to the following syntax:

$\mathbf{R} \longrightarrow \top_n \mid \mathbf{P} \mid (\$i/n : C) \mid \neg\mathbf{R} \mid \mathbf{R}_1 \sqcap \mathbf{R}_2$

$C \longrightarrow \top_1 \mid A \mid \neg C \mid C_1 \sqcap C_2 \mid \exists[\$i]\mathbf{R} \mid \leq k[\$i]\mathbf{R}$

i denotes a component of a relation (the equivalent of an ORM-role); if components are not named, then integer numbers between 1 and n_{max} are used, where n is the arity of the relation. k is a nonnegative integer for cardinality constraints. Only relations of the same arity can be combined to form expressions of type $\mathbf{R}_1 \sqcap \mathbf{R}_2$, and $i \leq n$, *i.e.*, the concepts and relations must be well-typed. The model-theoretic semantics of $\mathcal{DLR}$ is specified through the usual notion of interpretation, where $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, and the interpretation function $\cdot^{\mathcal{I}}$ assigns to each concept C a subset $C^{\mathcal{I}}$ of $\Delta^{\mathcal{I}}$ and to each n-ary $\mathbf{R}$ a subset $\mathbf{R}^{\mathcal{I}}$ of $(\Delta^{\mathcal{I}})^n$, such that the conditions are satisfied following Table 1. A *knowledge base* is a finite set $\mathcal{KB}$ of $\mathcal{DLR}$ (or $\mathcal{DLR}_{\mathit{ifd}}$) axioms of the form $C_1 \sqsubseteq C_2$ and $R_1 \sqsubseteq R_2$. 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), i.e. $C_1^{\mathcal{I}(t)} \subseteq C_2^{\mathcal{I}(t)}$ ($R_1^{\mathcal{I}(t)} \subseteq R_2^{\mathcal{I}(t)}$). $\top_1$ denotes the interpretation domain, $\top_n$ for $n \geq 1$ denotes a subset of the n-cartesian product of the domain, which covers all introduced n-ary relations; hence "$\neg$" on relations means difference rather than the complement. The ($\$i/n : C$) denotes all tuples in $\top_n$ that have an instance of C as their i-th component. The following abbreviations can be used: $C_1 \sqcup C_2$ for $\neg(\neg C_1 \sqcap \neg C_2)$, $C_1 \Rightarrow C_2$ for $\neg C_1 \sqcup C_2$, $(\geq k[i]R)$ for $\neg(\leq k-1[i]R)$, $\exists[i]R$ for $(\geq 1[i]R)$, $\forall[i]R$ for $\neg\exists[i]\neg R$, $R_1 \sqcup R_2$ for $\neg(\neg R_1 \sqcap \neg R_2)$, and $(i : C)$ for $(i/n : C)$ when n is clear from the context.

There are four extensions to $\mathcal{DLR}$. The most interesting in the current scope is $\mathcal{DLR}_{ifd}$ [10], because it captures most, or even all, of the expressivity of common conceptual modelling languages. $\mathcal{DLR}_{ifd}$ has two additional constructs compared to $\mathcal{DLR}$. First, $\mathcal{DLR}_{ifd}$ supports $i$dentification assertions on a concept C, which has the form ($\mathbf{id}$ $C[i_1]R_1, ..., [i_h]R_h$), where each R_j is a relation and each i_j denotes one component of R_j. This gives greater flexibility how to identify DL-concepts, most notably external uniqueness in ORM, ER's weak entity types, and objectification. Second, $\mathcal{DLR}_{ifd}$ supports non-unary $f$unctional $d$ependency assertions on a relation R, which has the form ($\mathbf{fd}$ R $i_1, ..., i_h \rightarrow j$), where $h \geq 2$, and $i_1, ..., i_h, j$ denote components of R (unary **fd**s lead to undecidability [10]), which are useful primarily for UML class diagram's methods and ORM's derived-and-stored fact types. Observe that there is no change in semantic rules because the algorithm for the extensions is checked against a (generalized) ABox [10]. Two extensions in another direction give greater expressiveness for representing knowledge with DL-roles. First, $\mathcal{DLR}_{\mu}$ contains the fixpoint construct for recursive structures over single-inheritance trees of a role [11] and thereby can represent acyclicity, transitivity, asymmetry, and (ir)reflexivity of DL-roles (ORM ring constraints). Second, $\mathcal{DLR}_{reg}$ adds support for *reg*ular expressions over roles, which includes the role composition operator "$\circ$" and reflexive transitive closure [13]. Another direction was taken by [3, 4], who developed a temporal extension to $\mathcal{DLR}$ (named $\mathcal{DLR}_{\mathcal{US}}$), which has additional $\mathcal{U}$ntil and $\mathcal{S}$ince operators for temporal conceptual modelling with the temporal EER $\mathcal{ER}_{VT}$ for the corresponding temporal databases.

2.2 The generic common conceptual data model $\mathcal{CM}_{com}$

With the $\mathcal{DLR}_{ifd}$ syntax and semantics from the previous section, we are now able to define the $\mathcal{CM}_{com}$ conceptual date modeling language. The formalisation adopted here is based on previous presentations [2, 10, 15] extended with EXTK and FD for $\mathcal{DLR}_{ifd}$'s **id** and **fd**, respectively, and making objectification OBJ, subroles ISA_U, role exclusion REX, and disjunctive mandatory roles RDM explicit; that is, given a particular conceptual data model in the generic conceptual data modeling language $\mathcal{CM}_{com}$, then there is an equi-satisfiable $\mathcal{DLR}_{ifd}$ knowledge base. The "new" ISA_U, REX, and RDM can be represented in $\mathcal{DLR}_{ifd}$ as $[r_i]R_i \sqsubseteq [r_i]R_j$, $[r_i]R_i \sqsubseteq \neg[r_i]R_j$, and $C_i \sqsubseteq \sqcup_{i=1}^{n} \exists[r_j]R_i$ among n relations each for the jth role with $j \leq n$, respectively. To date, these constraints have not been used other than for the ORM2 to $\mathcal{DLR}_{ifd}$ mapping [29, 30] because ORM and ORM2 do have these fine-grained notion, unlike UML and ER. Given that they are not harmful at all to UML and (E)ER, they are added to $\mathcal{CM}_{com}$. We first introduce the syntax, illustrate it with an example, and then proceed to the semantics.

Definition 1. (Conceptual Data Model $\mathcal{CM}_{com}$ syntax) *A $\mathcal{CM}_{com}$ conceptual data model is a tuple*

$$\Sigma = (\mathcal{L}, \text{REL}, \text{ATT}, \text{CARD}, \text{ISA}_C, \text{ISA}_R, \text{ISA}_U, \text{DISJ}, \text{COVER}, \text{KEY}, \text{EXTK}, \text{FD}, \text{OBJ} \\ \text{REX}, \text{RDM})$$

such that:

1. *$\mathcal{L}$ 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); the tuple $(\mathcal{C}, \mathcal{A}, \mathcal{R}, \mathcal{U}, \mathcal{D})$ is the* signature *of the conceptual data model Σ.*
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 $\text{CARD} : \mathcal{C} \times \mathcal{R} \times \mathcal{U} \mapsto \mathbb{N} \times (\mathbb{N} \cup \{\infty\})$ 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_C *is a binary relationship $\text{ISA}_C \subseteq \mathcal{C} \times \mathcal{C}$.*
6. ISA_R *is a binary relationship $\text{ISA}_R \subseteq \mathcal{R} \times \mathcal{R}$. ISA_R between relationships is restricted to relationships with the same arity.*
7. ISA_U *is a binary relationship $\text{ISA}_U \subseteq \mathcal{U} \times \mathcal{U}$. ISA_U between roles of relationships is restricted to relationships with the same arity.*
8. DISJ, COVER *are binary relations over $2^{\mathcal{C}} \times \mathcal{C}$, describing disjointness and covering partitions, respectively, over a group of* ISA *that share the same superclass.*
9. KEY *is a function, $\text{KEY} : \mathcal{C} \mapsto \mathcal{A}_i$, that maps a class symbol in $\mathcal{C}$ to its key attribute(s) where $1 < i \leq n$ and n denotes the arity of the relation or total amount of attributes of $\mathcal{C}$.*
10. EXTK *is an identification assertion (external uniqueness / weak entity type) $\text{EXTK} : \mathcal{C} \mapsto \mathcal{R}_i \times \mathcal{U}_j$ where $1 < i \leq n$ and each R_i denotes a relation and each U_j denotes the component of R_i, with $1 \leq j \leq m$ where m denotes the arity of relation R_i (intuitively, such an assertion states that no two instances of C agree on the participation to $R_1, \ldots, R_i$).*
11. FD *is a functional dependency assertion on a relation, $\text{FD} : \mathcal{R} \mapsto \mathcal{U}_i \times j$ where $i \geq 2$ and $U_1, ..., U_i, j$ denote components of R.*
12. OBJ *is an objectification function that maps an n-ary relation symbol $R \in \mathcal{R}$ to n binary relations $r_1, \ldots r_n$ over $\mathcal{C}$ ($R' \in \mathcal{C}$), $\text{OBJ}(R) = \langle [U_1]r_1, \text{CMAX}(R', r_1, U_1) = 1, \text{CMAX}(C_1, r_1, U_2) = 1, \ldots, [U_1]r_n, \text{CMAX}(R', r_n, U_1) = 1, \text{CMAX}(C_n, r_n, U_2) = 1 \rangle$ and $\text{EXTK}(R') = \langle U_1[r_1], \ldots, [U_n]r_n \rangle$.*
13. REX, RDM *are binary relations over $2^{\mathcal{U}} \times \mathcal{U}$, describing disjointness partitions over a group of roles $\mathcal{U}$ of relations in $\mathcal{R}$ of the same arity to which $\mathcal{C}$ participates.*

Observe that attribute symbols $\mathcal{A}$ refer in ORM parlance to value types (dashed ovals) and can participate in a relation REL (depending on the formalization, for ORM and ORM2, one thus has $\mathcal{C} \cup \mathcal{A}$), and FDs are useful for UML methods in UML and derived (-and-stored) fact types in ORM and ORM2, but they could be used for suboptimal modeling of relations that ought to be normalised first [1, 23]. Such variations will be specified in detail in the next section where we compare them in detail. The example below gives a flavour of how the syntax as given in Definition 1 can map to icons in a graphical $\mathcal{CM}_{com}$ conceptual model that uses UML class diagram, EER, and ORM2 notation. In principle, one can map the $\mathcal{CM}_{com}$ syntax to any set of icons or fixed-syntax pseudo-natural language as long as the relation between the $\mathcal{CM}_{com}$ syntax and icons (or pseudo-NL) has been specified. Put differently, with $\mathcal{CM}_{com}$ the mappings between the conceptual modeling languages can be 1:n instead of developing and maintaining m:n mappings.

Example: graphical syntax for $\mathcal{CM}_{com}$. Mappings from $\mathcal{CM}_{com}$ syntax to commonly used UML, EER, and ORM/ORM2 are shown in Fig. 1, where we have, among others, the ISA visualized with a directed arrow (e.g. `Author` ISA `Person`), cardinality constrains, such as $\text{CARD}(\texttt{Author}, \texttt{Writes}, \texttt{auth}) = (1, n)$ with a "`1..*`" in UML, craw's feet and line in EER and blob and line in ORM2 (the reading label and name of the binary predicate are assumed to be equivalent here), and DISJ and COVER with the constraint "`disjoint, complete`" in UML, in EER with double directed arrows and encircled `d`, and in ORM2 with an encircled blob with an X (e.g., `Author`, `Editor` are disjoint and cover `Person`). Last, a $\text{KEY}(\texttt{Person}) = \texttt{id}$ is shown in ORM2 as `(id)`, in a separate oval with underlined `id` in EER and as attribute with domain shown in UML as `id:string`. These constraints have their equivalent representation in $\mathcal{DLR}_{ifd}$ as follows: $\texttt{Author} \sqsubseteq \texttt{Person}$ (subsumption), $\texttt{Author} \sqsubseteq \exists[\texttt{auth}]\texttt{writes}$ (at least one), $\texttt{Author} \sqsubseteq \neg\texttt{Editor}$ (disjoint), $\texttt{Person} \sqsubseteq \texttt{Author} \sqcup \texttt{Editor}$ (covering), and $\texttt{Person} \sqsubseteq \exists^{=1}[\texttt{From}]\texttt{id}$, $\top \sqsubseteq \exists^{\leq 1}[\texttt{To}](\texttt{id} \sqcap [\texttt{From}] : \texttt{Person})$ (key). $\diamondsuit$

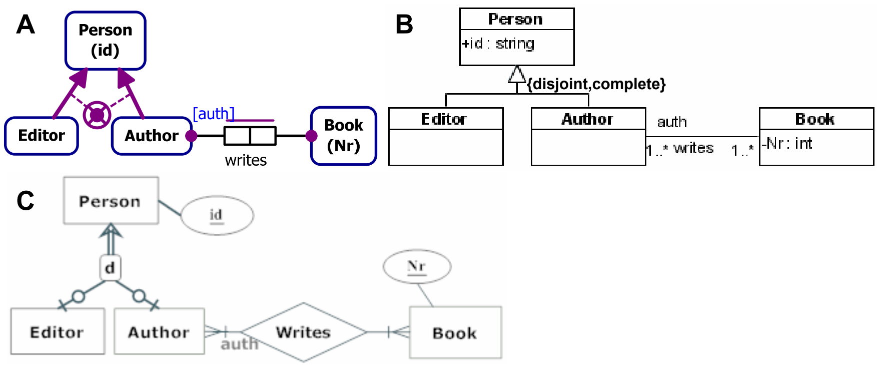


Fig. 1. Examples of graphical syntax for $\mathcal{CM}_{com}$ with ORM2 drawn in NORMA (A), UML class diagram drawn in VP-UML (B), and EER drawn with SmartDraw (C).

The model-theoretic semantics associated with this generic $\mathcal{CM}_{com}$ modeling language is defined as follows.

Definition 2. ($\mathcal{CM}_{com}$ **Semantics**) *Let Σ be a $\mathcal{CM}_{com}$ conceptual data model. An* interpretation *for the conceptual model Σ is a tuple $\mathcal{B} = (\Delta^{\mathcal{B}} \cup \Delta^{\mathcal{B}}_{D}, \cdot^{\mathcal{B}})$, 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* Σ*; and*
- $\cdot^{\mathcal{B}}$ *is a function that maps:*
  - *Every basic domain symbol* $D \in \mathcal{D}$ *into a set* $D^{\mathcal{B}} = \Delta^{\mathcal{B}}_{D_i}$*.*
  - *Every class* $C \in \mathcal{C}$ *to a set* $C^{\mathcal{B}} \subseteq \Delta^{\mathcal{B}}$*—thus* objects *are instances of classes.*
  - *Every relationship* $R \in \mathcal{R}$ *to a set* $R^{\mathcal{B}}$ *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,* $r \in R^{\mathcal{B}} \rightarrow (r = \langle U_1 : o_1, \ldots, U_n : o_n\rangle \wedge \forall i \in \{1, \ldots, n\}. o_i \in C^{\mathcal{B}}_i)$*. 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* $A \in \mathcal{A}$ *to a set* $A^{\mathcal{B}} \subseteq \Delta^{\mathcal{B}} \times \Delta^{\mathcal{B}}_D$*, such that, for each* $C \in \mathcal{C}$*, if* $\text{ATT}(C) = \langle A_1 : D_1, \ldots, A_h : D_h\rangle$*, then,* $o \in C^{\mathcal{B}} \rightarrow (\forall i \in \{1, \ldots, h\}, \exists a_i.\ \langle o, a_i\rangle \in A^{\mathcal{B}}_i \wedge \forall a_i. \langle o, a_i\rangle \in A^{\mathcal{B}}_i \rightarrow a_i \in \Delta^{\mathcal{B}}_{D_i})$*.*

$\mathcal{B}$ *is said a* legal database state *or* legal application software state *if it satisfies all of the constraints expressed in the conceptual data model:*

- *For each* $C_1, C_2 \in \mathcal{C}$*: if* C_1 ISA_C C_2*, then* $C^{\mathcal{B}}_1 \subseteq C^{\mathcal{B}}_2$*.*
- *For each* $R_1, R_2 \in \mathcal{R}$*: if* R_1 ISA_R R_2*, then* $R^{\mathcal{B}}_1 \subseteq R^{\mathcal{B}}_2$*.*
- *For each* $U_1, U_2 \in \mathcal{U}$*,* $R_1, R_2 \in \mathcal{R}$*,* $\text{REL}(R_1) = \langle U_1 : o_1, \ldots, U_n : o_n\rangle$*,* $\text{REL}(R_2) = \langle U_2 : o_2, \ldots, U_m : o_m\rangle$*,* $n = m$*,* $R_1 \neq R_2$*: if* U_1 ISA_U U_2*, then* $U^{\mathcal{B}}_1 \subseteq U^{\mathcal{B}}_2$*.*
- *For each* $R \in \mathcal{R}$ *with* $\text{REL}(R) = \langle U_1 : C_1, \ldots, U_k : C_k\rangle$*: all instances of* R *are of the form* $\langle U_1 : o_1, \ldots, U_k : o_k\rangle$ *where* $o_i \in C^{\mathcal{B}}_i$ *and* $1 \leq i \leq k$*.*
- *For each cardinality constraint* $\text{CARD}(C, R, U)$*, then:*
  $o \in C^{\mathcal{B}} \rightarrow \text{CMIN}(C, R, U) \leq \#\{r \in R^{\mathcal{B}} \mid r[U] = o\} \leq \text{CMAX}(C, R, U)$*.*
- *For all* $C, C_1, \ldots, C_n \in \mathcal{C}$*: if* $\{C_1, \ldots, C_n\}$ DISJ C*, then,*
  $\forall i \in \{1, \ldots, n\}. C_i \text{ ISA } C \wedge \forall j \in \{1, \ldots, n\},\ j \neq i. C^{\mathcal{B}}_i \cap C^{\mathcal{B}}_j = \emptyset$*.*
- *For all* $C, C_1, \ldots, C_n \in \mathcal{C}$*: if* $\{C_1, \ldots, C_n\}$ COVER C*, then,*
  $\forall i \in \{1, \ldots, n\}. C_i \text{ ISA } C \wedge C^{\mathcal{B}} = \bigcup_{i=1}^{n} C^{\mathcal{B}}_i$*.*
- *For each* $C \in \mathcal{C}, A \in \mathcal{A}$ *such that* $\text{KEY}(C) = A$*, then* A *is an attribute and* $\forall a \in \Delta^{\mathcal{B}}_D. \#\{o \in C^{\mathcal{B}} \mid \langle o, a\rangle \in A^{\mathcal{B}}\} \leq 1$*.*
- *For each* $C \in \mathcal{C}$*,* $R_h \in \mathcal{R}$*,* $h \geq 1$*,* $\text{REL}(R_h) = \langle U : C, U_1 : C_1, \ldots, U_k : C_k\rangle$*,* $k \geq 1$*,* $k + 1$ *the arity of* R_h*, such that* $\text{EXTK}(C) = [U_1]R_1, \ldots, [U_h]R_h$*, then for all* $o_a, o_b \in C^{\mathcal{B}}$ *and for all* $t_1, s_1 \in R^{\mathcal{B}}_1$*, ...,* $t_h, s_h \in R^{\mathcal{B}}_h$ *we have that:*

$$\left.\begin{array}{r} o_a = t_1[U_1] = ... = t_h[U_h] \\ o_b = s_1[U_1] = ... = s_h[U_h] \\ t_j[U] = s_j[U],\ \textit{for } j \in \{1, ..., h\},\ \textit{and for } U \neq j \end{array}\right\} \quad \textit{implies } o_a = o_b$$

where o_a *is an instance of* C *that is the* U_j*-th component of a tuple* t_j *of* R_j*, for* $j \in \{1, ..., h\}$*, and* o_b *is an instance of* C *that is the* U_j*-th component of a tuple* s_j *of* R_j*, for* $j \in \{1, ..., h\}$*, and for each* j*,* t_j *agrees with* s_j *in all components different from* U_j*, then* o_a *and* o_b *are the same object.*

– *For each* $R \in \mathcal{R}$, $U_i, j \in \mathcal{U}$, *for* $i \geq 2$, $i \neq j$, $\textsc{rel}(R) = \langle U_1 : C_1, \ldots, U_i : C_i, j : C_j \rangle$, $\textsc{fd}(R) = \langle U_1, \ldots, U_i \rightarrow j \rangle$, *then for all* $t, s \in R^{\mathcal{B}}$, *we have that* $t[U_1] = s[U_1], ..., t[U_i] = s[U_i]$ *implies* $t_j = s_j$.
– *For each* $R, r_1, \ldots, r_n \in \mathcal{R}$, $R', C_1, \ldots, C_n \in \mathcal{C}$, $U_1, \ldots, U_n, u_s, u_t \in \mathcal{U}$, R *has arity* n, $\textsc{rel}$, $\textsc{cmax}$, *and* $\textsc{extk}$ *interpreted as above, such that* $\textsc{obj}(R) = \langle [u_s]r_1, \textsc{cmax}(R', r_1, u_s) = 1, \textsc{cmax}(C_1, r_1, u_t) = 1, \ldots, [u_s]r_n, \textsc{cmax}(R', r_n, u_s) = 1, \textsc{cmax}(C_n, r_n, u_t) = 1 \rangle$, *then*
$\forall i \in \{2, \ldots, n\}.\{U_i, u_s, u_t \in U^{\mathcal{B}} \wedge r_i \in R^{\mathcal{B}} \wedge o_i, r' \in C^{\mathcal{B}} \wedge \langle U_1 : o_1, U_2 : o_2, \ldots, U_n : o_n \rangle \in R^{\mathcal{B}} \mid r_i[u_s] = r' \wedge r_i[u_t] = o_i\}$.
– *For each* $U_i \in \mathcal{U}$, $i \geq 2$, $R_i \in \mathcal{R}$, *each* R_i *has the same arity* m *(with* $m \geq 2$*)*, $C_j \in \mathcal{C}$ *with* $2 \leq j \leq i(m-1)+1$, *and* $\textsc{rel}(R_i) = \langle U_i : C_i, \ldots U_m : C_m \rangle$ *(and, thus,* $R_i \in R_i^{\mathcal{B}}$ *and* $o_j \in C_j^{\mathcal{B}}$*), if* $\{U_1, U_2, \ldots U_{i-1}\}$ $\textsc{rex}$ U_i, *then*
$\forall i \in \{1, \ldots, i\}.o_j \in C_j^{\mathcal{B}} \rightarrow \textsc{cmin}(o_j, r_i, u_i) \leq 1 \wedge u_i \neq u_1 \wedge \ldots \wedge u_i \neq u_{i-1}$ *where* $u_i \in U_i^{\mathcal{B}}$, $r_i \in R_i^{\mathcal{B}}$.
– *For each* $U_i \in \mathcal{U}$, $i \geq 2$, $R_i \in \mathcal{R}$, *each* R_i *has the same arity* m *(with* $m \geq 2$*)*, $C_j \in \mathcal{C}$ *with* $2 \leq j \leq i(m-1)+1$, *and* $\textsc{rel}(R_i) = \langle U_i : C_i, \ldots U_m : C_m \rangle$, *if* $\{U_1, U_2, \ldots U_{i-1}\}$ $\textsc{rdm}$ U_i, *then*
$\forall i \in \{1, \ldots, n\}.o_j \in C_j^{\mathcal{B}} \rightarrow \textsc{cmin}(o_j, r_i, u_i) \geq 1$ *where* $u_i \in U_i^{\mathcal{B}}$, $r_i \in R_i^{\mathcal{B}}$.

3 Comparison

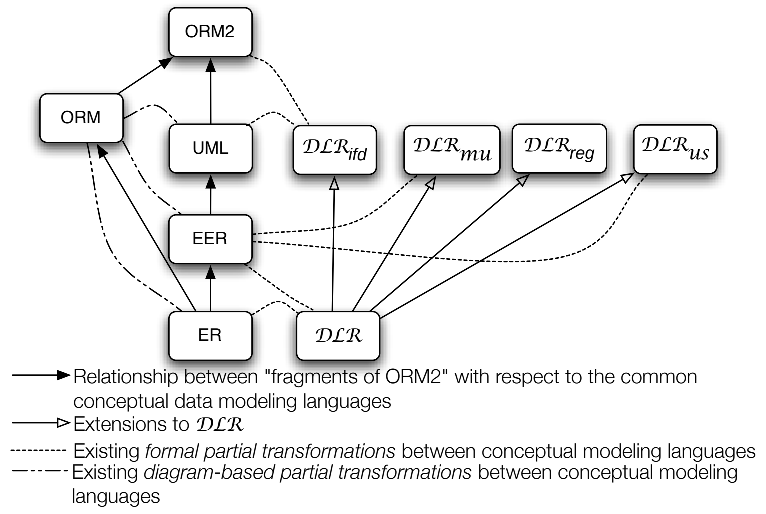
Given that the methodology for comparison of the modeling languages is through a formal language with precise syntax and semantics, one encounters distinct issues of which the two main ones are to

1. Establish what exactly is, or is not, part of “the” EER and “the” ORM languages, and if one should allow textual or OCL constraints to be part of the language, too; and
2. Decide what to do with an officially informal conceptual modeling language (UML [32]) or if there are alternative formalisations (ORM [22, 26]).

They are tightly related issues, because if one has a formal specification of the syntax and semantics of a conceptual modeling language as with $\mathcal{CM}_{com}$, it is utterly unambiguous as to what is, and what is not, in the language. One approach to address this is to also provide the syntax and semantics of ER, EER, UML class diagrams, ORM, and ORM2 in the same fashion as $\mathcal{CM}_{com}$. Alternatively, one could take industry-grade CASE tools and examine what the *de facto* standards are, or a consensus take on it where industry-grade tools, standards, academic prototypes, and research are converging. One can add a further issue, which is terminological (see below) together with historically grown preferences. Given that the more distant goal of the comparison is to have, on the one hand, a *common*, formal, foundation for conceptual data modeling languages, yet on the other hand, interoperability among conceptual models in different graphical syntax—hence, permitting maintaining the diversity—we take the converging approach in the remainder of this section. First, we list common (near-)equivalent terms of UML, ORM, (E)ER, and DL so as to avoid potential confusion. Concerning the main terms, we have, *roughly, and for indicative purpose only*:

- Class, object type, entity type, concept;
- Association, fact type, relationship, role;
- NA (component of an association), role, NA (component of a relationship), role component;
- Disjoint, disjunctive mandatory / exclusive, disjoint, disjoint;
- Complete, covering / total, covering, complete / covering;
- NA, external uniqueness, weak entity type, identification assertion;
- NA (property of an association), ring constraint, NA, role property;
- Method, derived(-and-stored) fact type, NA, functional dependency.

Second, bearing in mind the above considerations, a graphical rendering of the relations between the relevant languages is depicted in Fig.2, which concerns the language constructs they have; that is, the higher up in the figure, the more constructs that language has so that a modeler can represent the universe of discourse more precisely in more detail.



ORM2
ORM
UML
$\mathcal{DLR}_{ifd}$
$\mathcal{DLR}_{mu}$
$\mathcal{DLR}_{reg}$
$\mathcal{DLR}_{us}$
EER
ER
$\mathcal{DLR}$
Relationship between "fragments of ORM2" with respect to the common conceptual data modeling languages
Extensions to $\mathcal{DLR}$
Existing *formal partial transformations* between conceptual modeling languages
Existing *diagram-based partial transformations* between conceptual modeling languages

Fig. 2. Relations between common conceptual data modeling languages and the $\mathcal{DLR}$ family of Description Logic languages; $\mathcal{CM}_{com}$ has an equi-satisfiable $\mathcal{DLR}_{ifd}$ knowledge base. The higher up in the figure, the more constructors available in the language.

3.1 $\mathcal{CM}_{com}$ compared to ER, EER, UML, ORM, and ORM2

ER and EER. Before comparing ER and EER with other languages, we first have to address "the" δ(ER, EER), which asks for definitions of "the" ER and EER. Chen's original proposal for ER does include weak entity types [16] (EXTK), but this is not propagated to all ER tools, diagrammatic variants, and proposed extensions. In addition, not all notational variants have icons for n-ary relations when $n > 2$ and subtyping of entity types (e.g. Barker ER), although IDEF1X [27] does include ISA_C for ER that is generally accepted as new in EER. This seems better structured for EER that certainly contains support for subtyping of entity types (but not of relationships [20, 27]), total & exclusive subtyping, full cardinality constraints, and weak entity types. For practicing conceptual modelers it may be superfluous to note, but FD assertions are used for relational schemas [1], not ER and EER conceptual models, and therefore are not included in the definitions below. We make the following two decisions for defining ER and EER in terms of fragments of $\mathcal{CM}_{com}$.

Definition 3. ($\mathcal{CM}_{ER}$) *A $\mathcal{CM}_{ER}$ conceptual data model is a tuple*
$\Sigma = (\mathcal{L}, \text{REL}, \text{ATT}, \text{CARD}^{-}, \text{KEY}, \text{EXTK})$
adhering to $\mathcal{CM}_{com}$ syntax and semantics except that CARD *is restricted to any of the values $\{\geq 0, \leq 1, \geq 1\}$, denoted in Σ with* CARD^{-}*.*

Definition 4. ($\mathcal{CM}_{EER}$) *A $\mathcal{CM}_{EER}$ conceptual data model is a tuple*
$\Sigma = (\mathcal{L}, \text{REL}, \text{ATT}, \text{CARD}, \text{ISA}_C, \text{DISJ}, \text{COVER}, \text{KEY}, \text{EXTK})$
adhering to $\mathcal{CM}_{com}$ syntax and semantics.

Thus, $\mathcal{CM}_{ER} \subset \mathcal{CM}_{com}$ and $\mathcal{CM}_{EER} \subset \mathcal{CM}_{com}$. The δ(ER, EER) is the difference between $\mathcal{CM}_{ER}$ and $\mathcal{CM}_{EER}$, which is ($\text{CARD}, \text{ISA}_C, \text{DISJ}, \text{COVER}$), and that of δ(EER, $\mathcal{CM}_{com}$) is ($\text{ISA}_R, \text{ISA}_U, \text{OBJ}, \text{REX}, \text{RDM}, \text{FD}$). Recollecting that $\mathcal{DLR}_{\mathit{ifd}}$ is $\mathcal{DLR}$ with identification assertions and functional dependency, and thus that a conceptual data model with equi-satisfiable $\mathcal{DLR}$-knowledge base is a tuple minus EXTK, FD, and OBJ:
$\Sigma_{\mathcal{DLR}} = (\mathcal{L}, \text{REL}, \text{ATT}, \text{CARD}, \text{ISA}_C, \text{ISA}_R, \text{ISA}_U, \text{DISJ}, \text{COVER}, \text{KEY}, \text{REX}, \text{RDM})$
then such a $\mathcal{CM}_{\mathcal{DLR}}$ model is neither equivalent to a $\mathcal{CM}_{ER}$ one nor to a $\mathcal{CM}_{EER}$ one. Obviously, δ(ER,$\mathcal{CM}_{com}$) is the union of δ(ER, EER) and δ(EER, $\mathcal{CM}_{com}$).

UML class diagrams. The main problem with assessing UML class diagrams is that UML officially lacks a formal semantics (see OMG's UML specification [32]). Berardi et al. [8] gave a formal semantics to UML class diagrams both in first order predicate logic and description logic ($\mathcal{DLR}_{\mathit{ifd}}$). This UML-to-$\mathcal{DLR}_{\mathit{ifd}}$ transformation addresses neither UML's OCL fully nor the informally described semantics of UML's composite and shared aggregation constructs that intuitively correspond to proper part-whole and part-whole relations due to presence (absence) of "asymmetric" in the respective descriptions in the UML specification [32]. For both the first order logic and $\mathcal{DLR}_{\mathit{ifd}}$ formalisation of UML's aggregation, the formal semantics of the standard association relation (DL-role) is used; hence, any issues with representing part-whole relations (e.g., [21, 31]) are ignored. This poses no real practical problem, because due—or: thanks—to the lack of formal semantics of the UML specification, one can choose one's own formal semantics for convenience. In addition, most DL languages do not have a primitive for parthood relations anyway (nor have ORM(2) and (E)ER). All other UML class diagram language constructs have been mapped into $\mathcal{DLR}_{\mathit{ifd}}$ (see [8] for the mapping and examples). Here is not the place to assess UML's intentions with the filled and open diamond shapes for composite and shared aggregation, but we note the elusive "extra" with a PW relation that, at least, can be characterized as PW $= \langle U_1 : C_1, U_2 : C_2 \rangle$, hence a binary relation. Further, the *formal* UML and graphical support in Icom [8, 18] enable full access to components of an UML association, but the UML specification [32] and traditional CASE tools such as RationalRose give only limited access to them in the graphical representation (VP-UML now offers restricted cardinality specification of $\{\leq 1, 1, \geq 1\}$), so that one cannot use ISA$_U$; therefore, ISA$_U$ is omitted from the definition below (hence, also REX and RDM are absent).

Definition 5. ($\mathcal{CM}_{UML}$) *A $\mathcal{CM}_{UML}$ conceptual data model is a tuple*
$\Sigma = (\mathcal{L}, \text{REL}, \text{ATT}, \text{CARD}, \text{ISA}_C, \text{ISA}_R, \text{DISJ}, \text{COVER}, \text{KEY}, \text{EXTK}, \text{FD}, \text{OBJ}, \text{PW})$
adhering to $\mathcal{CM}_{com}$ syntax and semantics, except for the aggregation association PW*, with syntax* PW $= \langle U_1 : C_1, U_2 : C_2 \rangle$*, that has no defined semantics.*

Thus, given the absence of full access to association ends and the minimal characterization of PW, *de facto*, $\mathcal{CM}_{UML} \subset \mathcal{CM}_{com}$ up until a proper, coherent semantics is defined for PW and full access to association ends is supported in either the UML specification or mainstream modeling tools, or both. Either way, $\mathcal{CM}_{EER} \subset \mathcal{CM}_{UML}$ holds with a δ consisting of (ISA$_R$, FD, OBJ, PW).

ORM and ORM2. While ORM enjoys the comparative advantage of a having formal characterizations in first order logic [22, 26], variations and additional constructs (primarily ring constraints) have been implemented in tools, such as VisioModeler, and not all extensions in ORM2—exclusive total covering of subtypes, role values, and deontic constraints—have been formalized [25]. Moreover, ORM is already undecidable due to constraints with patterns of the type "constraint x over k ORM-roles" over an n-ary relation, $n \geq 3$, and $k < n$ because they correspond to arbitrary projections (subset over k roles, multi-role frequency, set-equality over k roles, and role exclusion over k roles [29, 30]). On the flip side, an analysis of 168 ORM diagrams with about 1800 constraints made with LogicBlox software revealed that such constructs are rarely used in practice [33], although it could use further analysis as to why this is so. With respect to ORM2 and $\mathcal{CM}_{com}$, $\mathcal{CM}_{com}$ is a proper subset, but, given Halpin's formalization [22], this is not the case for ORM. Due to space limitations, we do not repeat Halpin's formalization here, and omit role values[1] and deontic constraints for they are, at the time of writing, preliminary additions to the language or require deontic logic, respectively.

Definition 6. *A $\mathcal{CM}_{ORM}$ conceptual data model is a tuple*
$\Sigma = (\mathcal{L}, \text{REL}, \text{ATT}, \text{CARD}, \text{ISA}_C, \text{ISA}_R, \text{ISA}_U, \text{KEY}, \text{EXTK}, \text{FD}, \text{OBJ}, \text{REX}, \text{RDM}, \text{JOIN}, \text{KROL}, \text{RING}^-)$
adhering to $\mathcal{CM}_{com}$ syntax and semantics, and, in addition, such that:

- JOIN *comprises the following constraints:* {join-subset, join-equality, join-exclusion} *over ≥ 2 n-ary relations, $n \geq 2$, as defined in [22].*

[1] $\forall x, y (x \in \{v_i, ..., v_k\} \rightarrow (R(x, y) \rightarrow C_i(x) \wedge C_j(y)))$. A candidate approach for transformation into a $\mathcal{DLR}_{ifd}$ representation is proposed in [29].

– KROL *comprises the following constraints:* {subset over k roles, multi-role frequency, set-equality over k roles, role exclusion over k roles} *over an n-ary relation, $n \geq 3$, and $k < n$, as defined in [22].*
– RING$^-$ *comprises the following constraints:* {intransitive, irreflexive, asymmetric}*, as defined in [22].*

Definition 7. *A $\mathcal{CM}_{ORM2}$ conceptual data model is a tuple*

$$\Sigma = (\mathcal{L}, \text{REL}, \text{ATT}, \text{CARD}, \text{ISA}_C, \text{ISA}_R, \text{ISA}_U, \text{DISJ}, \text{COVER}, \text{KEY}, \text{EXTK}, \text{FD}, \text{OBJ}, \text{REX}, \text{RDM}, \text{JOIN}, \text{KROL}, \text{RING})$$

adhering to the syntax and semantics as defined for $\mathcal{CM}_{com}$, and such that:
– KROL *and* JOIN *are as in Definition 6.*
– RING *comprises the following constraints:* {intransitive, irreflexive, asymmetric, antisymmetric, acyclic, symmetric}*, as defined in [22, 23].*

Hence $\mathcal{CM}_{com} \subset \mathcal{CM}_{ORM2}$. Looking at the gap between the two, RING can be met partially with $\mathcal{DLR}_\mu$ and $\mathcal{DLR}_{reg}$, but this comes at the cost that then one can neither represent external uniqueness constraints nor non-unary functional dependencies. Adding **id** and **fd** to $\mathcal{DLR}_\mu$ looks promising, but proof sketches have yet to be worked out in detail.

3.2 Confronting EER, UML class diagrams and ORM2

With the unambiguous syntax and semantics of $\mathcal{CM}_{com}$ and the common conceptual modeling languages, comparing the language constructs of ER, EER, UML class diagrams, ORM, and ORM2 has become almost trivial. There are, however, four finer-grained items that merit attention. First, KEY is for *single attribute* keys (ORM2 reference scheme), whereas for a key consisting of multiple attributes (n-ary fact type where $n \geq 3$) or weak entity types (external uniqueness), one has to use EXTK. In addition, ORM requires that an internal uniqueness constraint over k roles in an n-ary relation has to span $\geq n-1$ roles to have an elementary fact type. For instance,
(i) An entity type and relation as Course($\underline{\text{Course\_Name}}$, Uni_Year, Teacher) is valid in ER, EER, UML, and $\mathcal{CM}_{\mathcal{DLR}}$, but is not permitted in ORM and ORM2 because it is not an elementary fact type;
(ii) A key constraint as Course($\underline{\text{Course\_Name, Uni\_Year}}$, Teacher) is valid in ER, EER, ORM, and ORM2, but it is not in the $\mathcal{DLR}$ family except for $\mathcal{DLR}_{ifd}$ and, hence, $\mathcal{CM}_{com}$.

Second, it might come as a surprise that $\mathcal{CM}_{UML}$ is not a proper fragment $\mathcal{CM}_{ORM}$. This is because we took Halpin's formal characterisation [22], thereby missing support for total and exclusive subtypes (this cannot be represented in VisioModeler 3.1 either). Given the underspecified PW, then clearly $\mathcal{CM}_{UML} \subset \mathcal{CM}_{ORM2}$ holds, where ORM2 also enjoys software support to actually use all additional constructs in NORMA [35]. Third, the difference between conceptual modeling languages with attributes (UML, ER, and EER) and attribute-free languages (ORM and ORM2). This aspect is somewhat hidden with DL languages and $\mathcal{CM}_{com}$: while one has ATT and KEY in $\mathcal{CM}_{com}$, in fact, in $\mathcal{DLR}_{ifd}$

it is represented as a binary relation with predefined cardinality constraints (1 or max 1) just like the ORM(2) reference scheme graphical elements are shown in a compact representation but have a binary relation underneath (cf. Fig.1-A with the `id` for `Person`). In this way, one thus has it both ways with attributes for the languages that need it and attribute-free for the others. Fourth, there are constraints in $\mathcal{CM}_{ORM}$ and $\mathcal{CM}_{ORM2}$ that are not covered in full by $\mathcal{CM}_{com}$. In this respect, one may wonder why the comparison has not been carried out with full first order logic to have most of the desired expressivity (deontic constrains require another logic). This route was chosen, because there are advantages to using DL languages, and $\mathcal{DLR}_{\mathit{ifd}}$ in particular. With the plethora of well-studied DL languages and their correspondences with conceptual data modeling languages, one gains a better insight in the properties of these conceptual modelling languages through their (non-)correspondence with the DLs. From the extant mappings it has become immediately clear that UML in $\mathcal{CM}_{UML}$ is within the decidable fragment of first order logic and, in fact, is in the ExpTime-complete complexity class—because $\mathcal{DLR}_{\mathit{ifd}}$ is [9]. One can choose computationally more appealing subsets in NP or NLogSpace for both UML and (E)ER when one takes certain fragments [2, 28], e.g., without sub-relations, because apparently not all constructs are used always in practice [17, 33]. Regarding $\mathcal{DLR}_{\mathit{ifd}}$, it provides the most expressive common denominator among the languages and thereby provides the, thus far, best trade-off between expressiveness for conceptual data modeling and computation. Although the emphasis is more often on the former for traditional conceptual modeling, computation with conceptual models has the distinct advantage of automated satisfiability and consistency checking with reasoners, such as Racer and Pellet or a special-purpose reasoner [28, 33], before generating a database or other application software (for examples what one gains compared to a validator, consult, e.g., the Icom tutorial page [34]). The usefulness of this is, among others, that any unsatisfiable object types are detected at the conceptual analysis stage instead of finding out in the implementation that some table remains empty or java class never can be instantiated. Moreover, the common formal foundation with $\mathcal{CM}_{com}$ offers an "interchange" to simplify the linking and integration of conceptual data models made in different conceptual data modeling languages[2]. Also, each modeler could, in principle, continue using her preferred conceptual modeling language, yet have diagrams that are fully compatible with the other conceptual modeling languages, as well as automated precise translations. Pushing this even further, having taken a formal approach to comparison of the languages cf. a how-to procedure, CASE tool developers can choose any variation of the diagrammatical rendering of the $\mathcal{CM}_{com}$ syntax and still avail of the transferrable results presented here.

Thus, the comparison updates [15] with the latest research into the $\mathcal{DLR}$ family and by availing of DL languages, the here defined $\mathcal{CM}_{com}$ augments the current informal mapping in e.g. [23] and thereby moves closer to *online*

[2] Icom [18, 19] already supports specification of and reasoning over inter-model assertions for UML class diagrams, and the interface is being extended to also support EER and ORM graphical elements.

interoperability between ORM/ORM2, UML, and EER through a *formal* correspondence between the conceptual modelling languages with the $\mathcal{DLR}$ family as unifying paradigm at the junction.

4 Conclusions

The main commonly used conceptual data modelling languages—ER, EER, UML class diagrams, ORM, and ORM2—were compared on their language constructs through the formally defined $\mathcal{CM}_{com}$ conceptual modeling language as greatest common denominator that also has an equi-satisfiable $\mathcal{DLR}_{\mathsf{ifd}}$ knowledge base. Concerning available constructors in the modeling languages, it was argued that UML, ER, EER, and ORM, are proper fragments of ORM2. The results obtained with $\mathcal{CM}_{com}$ simplifies interoperability of conceptual data models modeled in different graphical languages.

We are currently looking into usage of $\mathcal{DLR}_{\mathcal{US}}$ [3] to further explore feasibility of temporal extensions for languages other than temporal EER ($\mathcal{ER}_{VT}$) and, e.g., for use with dynamic rules [7] and life cycle semantics of part-whole relations [5], and to examine if a "$\mathcal{DLR}_{\mu\mathsf{ifd}}$" remains within the decidable fragment of first order logic to address ORM's ring constraints.

References

1. Abiteboul, S., Hull, R., Vianu, V. *Foundations of databases.* Addison Wesley, USA. 1995.
2. Artale, A., Calvanese, D., Kontchakov, R., Ryzhikov, V., Zakharyaschev, M. Reasoning over Extended ER Models. *Proc. of ER'07*, Springer LNCS 4801, 277-292.
3. Artale, A., Parent, C., Spaccapietra, S. Evolving objects in temporal information systems. *Annals of Mathematics and Artificial Intelligence*, 2007, 50(1-2), 5-38.
4. Artale, A., Franconi, E., Wolter, F., Zakharyaschev, M. A temporal description logic for reasoning about conceptual schemas and queries. In: *Proc. of JELIA '02*, S. Flesca, S. Greco, N. Leone, G. Ianni (Eds.), Springer, 2002, LNAI 2424, 98-110.
5. Artale, A., Keet, C.M. Essential and mandatory part-whole relations in conceptual data models. *Proc of DL'08*. 13-16 May 2008, Dresden, Germany. (in print).
6. Baader, F. Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F. (eds). *Description Logics Handbook*, Cambridge University Press, 2003.
7. Balsters, H., Carver, A., Halpin, T., Morgan, T. Modeling dynamic rules in ORM. *Proc. of ORM 2006*. Springer LNCS 4278, 2006, 1201-1210.
8. Berardi, D., Calvanese, D., De Giacomo, G. Reasoning on UML class diagrams. *Artificial Intelligence*, 2005, 168(1-2):70-118.
9. Calvanese, D., De Giacomo, G. Expressive description logics. In: *The Description Logic Handbook*, Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P. (Eds). Cambridge University Press, 2003. 178-218.
10. Calvanese, D., De Giacomo, G., Lenzerini, M. Identification constraints and functional dependencies in Description Logics. In *Proc. of IJCAI 2001*, 2001, 155-160.
11. Calvanese, D., De Giacomo, G., Lenzerini, M. Reasoning in expressive description logics with fixpoints based on automata on infinite trees. In *Proc. of IJCAI 1999*, 84-89.

12. Calvanese, D., De Giacomo, G., Lenzerini, M., Nardi, D., Rosati, R. Description logic framework for information integration. In *Proc. of KR'98*, 2-13.
13. Calvanese, C., De Giacomo, G., Lenzerini, M. On the decidability of query containment under constraints. In: *Proc. of PODS'98*, 149-158, 1998.
14. Calvanese, D., Lenzerini, M., Nardi, D. Description logics for conceptual data modeling. In: Chomicki, J., Saake, G. (Eds.), *Logics for Databases and Information Systems*. Kluwer, Amsterdam. 1998.
15. Calvanese, D., Lenzerini, M. & Nardi, D. Unifying class-based representation formalisms. *Journal of Artificial Intelligence Research*, 11:199-240, 1999.
16. Chen, P.P. The Entity-Relationship model—Toward a unified view of data. *ACM Transactions on Database Systems*, 1976, 1(1): 9-36.
17. Erickson, J., Siau, K. The complexity of UML: differentiating practical and theoretical complexity. In *Proc. of EMMSAD'04*, Vol 1., 309-316.
18. Fillottrani, P., Franconi, E., Tessaris, S. The new ICOM ontology editor. In: *19th International Workshop on Description Logics (DL'06)*, Lake District, UK. May 2006.
19. Franconi, E., Ng, G. The iCom tool for intelligent conceptual modelling. *7th Int'l Workshop on Knowledge Representation meets Databases*. Berlin, Germany. 2000.
20. Gogolla, M., Hohenstein, U. Towards a semantic view of an Extended Entity-Relationship Model. *ACM Transactions on Database Systems*, 1991, 16(3), 369-416.
21. Guizzardi, G. *Ontological foundations for structural conceptual models*. PhD Thesis, Telematica Institute, Twente University, Enschede, the Netherlands. 2005.
22. Halpin, T.A. *A logical analysis of information systems: static aspects of the data-oriented perspective*. PhD Thesis, University of Queensland, Australia. 1989.
23. Halpin, T. *Information Modeling and Relational Databases*. San Francisco: Morgan Kaufmann Publishers, 2001.
24. Halpin, T. A. Comparing Metamodels for ER, ORM and UML Data Models. In: *Advanced Topics in Database Research*, vol. 3, K. Siau (Ed.), Idea Publishing Group, Hershey PA, USA, Ch. II, pp 23-44.
25. Halpin, T. ORM2. In *Proc. of ORM 2005*. In: OTM Workshops 2005. Halpin, T., Meersman, R. (eds.), Springer LNCS 3762, 676-687.
26. Hofstede, A.H.M. ter, Proper, H.A.. How to Formalize It? Formalization Principles for Information Systems Development Methods. *Information and Software Technology*, 1998, (40(10): 519-540.
27. Integrated Definition Methods (IDEF1X). http://www.idef.com/IDEF1X.html.
28. Kaneiwa, K., Satoh, K. Consistency Checking Algorithms for Restricted UML Class Diagrams. In: *Proc. of FoIKS '06*, Springer LNCS 3861, 2006. 219-239.
29. Keet, C.M. Prospects for and issues with mapping the Object-Role Modeling language into $\mathcal{DLR}_{ifd}$. In: *Proc. of DL'07*, 8-10 June 2007, Bressanone, Italy. CEUR-WS Vol-250, 331-338.
30. Keet, C.M. *Mapping the Object-Role Modeling language ORM2 into Description Logic language* $\mathcal{DLR}_{ifd}$. KRDB Research Centre TR KRDB07-2, Faculty of Computer Science, Free University of Bozen-Bolzano, Italy. 2007. arXiv:cs.LO/0702089v1.
31. Motschnig-Pitrik, R. & Kaasbøll, J. (1999). Part-Whole Relationship Categories and Their Application in Object-Oriented Analysis. *IEEE Transactions on Knowledge and Data Engineering*, 1999, 11(5):779-797.
32. Object Management Group. *Unified Modeling Language: Superstructure*. v2.0. formal/05-07-04. http://www.omg.org/cgi-bin/doc?formal/05-07-04.
33. Smaragdakis, Y., Csallner, C., Subramanian, R. Scalable automatic test data generations from modeling diagrams. In: *ASE'07*, Nov. 5-9, Atlanta, Georgia, USA. 4-13.
34. Icom. http://www.inf.unibz.it/~franconi/icom/.
35. NORMA. http://sourceforge.net/projects/orm/.