Evidence-Based Languages for Conceptual Data Modelling Profiles

Pablo Rubén Fillottrani[1,2] and C. Maria Keet[3(✉)]

[1] Departamento de Ciencias e Ingeniería de la Computación, Universidad Nacional del Sur, Bahía Blanca, Argentina
prf@cs.uns.edu.ar
[2] Comisión de Investigaciones Científicas, La Plata, Provincia de Buenos Aires, Argentina
[3] Department of Computer Science, University of Cape Town, Cape Town, South Africa
mkeet@cs.uct.ac.za

Abstract. To improve database system quality as well as runtime use of conceptual models, many logic-based reconstructions of conceptual data modelling languages have been proposed in a myriad of logics. They each cover their features to a greater or lesser extent and are typically motivated from a logic viewpoint. This raises questions such as what would be an evidence-based common core and what is the optimal language profile for a conceptual modelling language family. Based on a common metamodel of UML Class Diagrams (v2.4.1), ER/EER, and ORM/2's static elements, a set of 101 conceptual models, and availing of computational complexity insights from Description Logics, we specify these profiles. There is no known DL language that matches exactly the features of those profiles and the common core is small (in the tractable $\mathcal{ALNI}$). Although hardly any inconsistencies can be derived with the profiles, it is promising for scalable runtime use of conceptual data models.

1 Introduction

Database and information system development and use can be aided by conceptual data models that have a logic-based underpinning, both at the analysis stage and during runtime. Automated reasoning over isolated conceptual data models, such as EER and UML Class Diagrams, to improve their quality and avoid bugs aims to tackle this problem by various means. Notably, Description Logics (DLs) is used (among many: [1,5,10]), but also other techniques, such as constraint programming [8], OCL [27], CLIF [26], and Alloy [7]. There are also scenarios for using the models at runtime, such as for scalable test data generation [28] and for designing [6] and executing [13] queries with the conceptual model's vocabulary rather than quirky database table names and columns. Logic-based approximations of conceptual models are used also for querying databases during the stage of query compilation [32].

All these efforts face the same issue: how to formalise the diagrams in which logic? Even just zooming in on DLs shows that at times it is claimed that any

T. Morzy et al. (Eds.): ADBIS 2015, LNCS 9282, pp. 215–229, 2015.
DOI: 10.1007/978-3-319-23135-8_15

one of the languages in the $\mathcal{DLR}$ family is good for representing and unifying the conceptual data modelling languages [12], the much leaner DL-Lite family of languages [1], or using $\mathcal{SROIQ}$ (OWL 2 DL) instead [33]. While one could choose one's pet language, from a scientific viewpoint, it would be good to know *which DL (or other logic) is most appropriate, and why?* Here, 'most appropriate' is cast in the light of the needs from the viewpoint of the modelling languages, and what features those conceptual modellers bother to use in their models. This raises the following questions:

1. What is the profile of a common core of language features among the main conceptual data modelling languages (CDMLs)?
2. Is there an optimal language profile to capture each of UML, ER and EER, and ORM and ORM2, based on a set of publicly available diagrams?
3. Are any language features missing from the many extant DL languages, given a set of actual conceptual models, or too much in any case?

To establish a common core, harmonisation of terminology across CDMLs is needed. This has been done with a unifying metamodel of the static, structural entities (including constraints) of UML v2.1.4 (Class Diagrams), ER, EER, ORM, and ORM2 [16,20]. The feature overlap that can be determined from the metamodel is augmented by a classification of the entities in the models of a dataset of 101 conceptual data models of the three language families. These models were collected from projects, scientific papers, textbooks, and online diagrams; the dataset and analysis are available online [15,21].Together with the known computational complexity of various DL languages and formalisation trade-offs, a common core and profiles for each of the UML, ER/EER, and ORM/ORM2 families have been specified. This ranges from $\mathcal{ALNI}$ of the core to the "$\mathcal{DLR}_{\mathit{ifd}}$ without disjointness and completeness" for ORM2, with a good approximation with $\mathcal{CFDI}_{nc}^{\forall-}$. Remarkably, these conceptual model profiles/DL fragments are all tractable, and therewith are very suitable for scalable runtime usage of conceptual models. The only possible complication are the (sparsely occurring) advanced datatype constraints, which DLs do not support well, and promising computational complexity results are yet to be obtained. The remainder of the paper is structured as follows. We first introduce preliminaries about the metamodel, the dataset, and fundamental formalisation choices (Sect. 2). The core and CDML profiles are described and motivated in Sects. 3 and 4, respectively. Related works are analysed in each profile section. We discuss in Sect. 5 and conclude in Sect. 6.

2 Preliminaries

To put the profiles in context, we first describe the input we used, being the unifying metamodel, the dataset, motivation for the logic chosen, and some insights from philosophy that clarifies CDML formalisations. We assume the reader is familiar with the basic DL notation; see [4] for details.

2.1 Unifying Metamodel and Dataset

As the three CDML families under consideration—UML v2.1.4 Class Diagrams, ER and EER (henceforth abbreviated as (E)ER), and ORM and ORM2 (henceforth abbreviated as ORM/2)—originate from different sub-fields in database and information systems development, they each have their own vocabulary with syntactic and semantic differences. This has been investigated and a terminology comparison table and a unified metamodel are presented in [20], which therewith facilitates cross-language comparisons as well as categorisation of entities of models in those languages into the harmonised terminology. Further, it neatly demonstrates the intersection of entities across the languages, which has been extended in [16] also with constraints. Its top-type is Entity, which has four direct subclasses: Relationship with 11 subclasses, Role, Entity type with 9 subclasses, and Constraint with 49 subclasses. All entities also have constraints specified among them on how they may be used, e.g., each relationship must have at least two roles and a disjoint object type constraint is only declared on class subsumptions.

We have used this metamodel to classify the entities of the models in a set of 101 UML, (E)ER, and ORM/2 models. Their average 'model size' (vocabulary+subsumption) is about 50 entities/model, with at total of 8036 entities of which 5191 (i.e., 64 %) are entities that were classified in an entity (language feature) that appears in all three language families and 1108 (13.8 %) in ones that are unique to a language family (e.g., UML's aggregation) [21]. While one would prefer industry models, they are not publicly available. Only one paper presents quantitative results on industry models, being a set of 168 ORM diagrams that were made by a single engineer in the proprietary modelling tool from LogicBlox [28]. Our model data for ORM is similar to theirs [21].

2.2 General Logic-Based Reconstruction Design Choices

There are two important considerations: which logic family to use, and what to do with the relationships.

Concerning the language(s) to create a logic-based reconstruction of the three main CDML families under consideration, and to compare them, one could go for some 'arbitrary' very expressive logic, such as FOL, or one of its serialisations (e.g., Common Logic's CLIF), or *a priori* a decidable one (DLs) with CDM features ($\mathcal{DLR}$ family of DLs) or in line with the Semantic Web (OWL species). There is no best fit with respect to various requirements, as the comparison in Table 1 demonstrates, other than that DLs give us a view on decidability and computational complexity of concept satisfiablity, which is therefore chosen.

A formalisation decision that applies to each CDML family concerns the relationships, which is due to two distinct ontological commitments as to what they are, being the so-called *standard view* and *positionalism*. The standard view uses directionality—or: a natural language 'reading' direction—of the relationship where the participating objects have a fixed order, as formalised with the n-ary predicate ($n \geq 2$), conflating the verbalisation with the name of the relationship. In the positionalist commitment, relationships have (are composed of)

Table 1. Selection of languages, requirements, and their evaluation for formalising UML, (E)ER, and ORM/2; "–": negative evaluation; "+": positive. (OntoIOP is in the process of standardisation with OMG, which aims to link logical theories represented in the same or different languages.)

$\mathcal{DLR}_{ifd}$	OWL 2 DL	FOL
– no implementation	+ several reasoners, relatively scalable	– few reasoners, not really scalable
– no interoperability	+ linking with ontologies doable	– no interoperability with existing infrastructures
– no integration	+ 'integration' with OntoIOP	+ 'integration' with OntoIOP
+ formalisation exist	– formalisation to complete	± formalisation exist
+ little feature mismatch	– what to do with OWL 2 DL features not in the CDM languages and vv.	+ little feature mismatch
– modularity infrastructure	+ modularity infrastructure	– modularity infrastructure
± EXPTIME-complete	± N2EXPTIME-complete	– undecidable

argument places that are entities of themselves, which are filled by the participating objects, and those positions have no order in the relationship; refer to [18,22] for theoretical details. The three selected CDML families are positionalist [20]—UML associations have association ends, ORM/2 fact types have roles, (E)ER has components of a relationship—, but most DLs are standard view, except for the $\mathcal{DLR}$ family. The $\mathcal{DLR}$ family has only one proof-of-concept implementation [9], however, whereas the former do in so far as they are OWL 2 DL or proper fragments thereof. Therefore, we need to assess how to convert positionalist relationships into standard view ones. There are several options, each with its trade-offs that may affect the complexity of the language. We use the diagrams in Fig. 1 as illustration to discuss them.

The UML standard v2.4.1 [25] and earlier versions require named association ends (DL role components), like the `teacher` and `taughtBy` in Fig. 1, but not a name of the association (DL role). Options to formalise it:

(1) make each association end a DL role, `teacher` and `taughtBy`, then choose:
   (a) declare them inverse of each other with $\texttt{teacher} \equiv \texttt{taughtBy}^{-}$,
   (b) do not declare them inverses.
(2) choose to 'bump up' either `teacher` or `taughtBy` from association end to DL role, and use the other through a direct inverse (`ObjectInverseOf()` in OWL 2) and omit the extension of the vocabulary with the other (e.g., `teacher` and $\texttt{teacher}^{-}$ cf. adding also `taughtBy`).

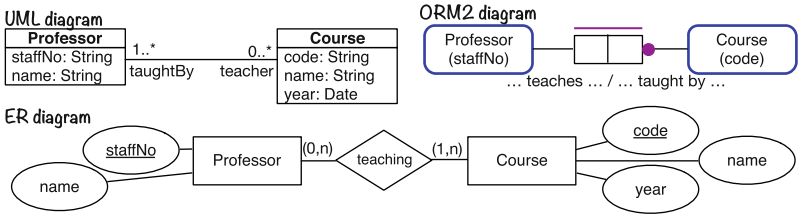

Fig. 1. Sample UML, EER and ORM2 diagrams, representing that a course is taught by at least one professor, and a professor may teach zero or more courses (for space limitations, some value types in the ORM diagram are suppressed).

The explicit inverses (Item 1a) is essentially a workaround for having made two relationships where only one existed, trying to keep the two somehow related so as to make up for the 'splitting'. Arguably, declaring them inverses is not strictly necessary, and omitting it could be considered comparable to omitting the identification constraint across the roles of a reification of an n-ary into n binaries in OWL, which is generally tolerated. Either way, one can deterministically and automatically generate a formalisation of the UML Class Diagram.

Item 2's need for a choice among association ends can be done economical in the formalisation by taking the one that requires a cardinality constraint; in the example, the preference is for taughtBy, not teacher (the latter has only a 0..∗), generating a domain and a range axiom for `taughtBy`, and a $\texttt{Course} \sqsubseteq \exists \texttt{taughtBy}.\texttt{Professor}$. This can be automated for cases like the example, but not if Professor were to have also a 1..∗ (or more) multiplicity, which then would make it an arbitrary choice again, and therewith, still not a single, unique formalisation.

In favour of the latter main option, is that it has been shown that using Item 2-inverses compared to Item 1a-inverses results in better automated reasoner performance, reducing time by more than a third [19]. Adding inverses to a language may change its computational complexity, however, and a few popular ones do not have inverses; e.g., $\mathcal{ALCQ}$ and $\mathcal{ALCQI}$ are both PSpace-complete [29], and OWL 2 EL [24] does not have inverse object properties.

ER is also positionalist, but it has a different practical issue cf. UML. It is customary to give the relationship a name that is 'non-directional', like the teaching in Fig. 1 or its infinitive, rather than naming the relationship components. Morphing it into the standard view then requires either:

(i) a renaming of the relationship to prevent an ambiguous DL role name in the formalisation, or
(ii) an arbitrary domain and range assignment.

This user-mediated step favours using an `ObjectInverseOf()` rather than adding a second new name if more than one cardinality constraint is not $(0, n)$, but this means also here it cannot be guaranteed it will result in exactly one formalisation of the diagram. (Some UML models have association names, not named

association ends, but the same problem does not exist, for an association name has a filled arrow-tip for the reading direction.)

ORM's fact type readings can be useful candidates for naming DL roles, but only one is required in a diagram, not n for the n participating entities. The software assigns auto-generated identifiers to the ORM roles and to the fact types (relationships) by default, but a modeller also can name them, which is then shown in the diagram. Due to this freedom in modelling, one single rule is not possible, but a sequence of possible cases—and choices—is needed. Thus, it cannot be guaranteed that there will be only one formalisation.

In sum, no matter which formalisation option is chosen regarding relationships, the CDML families each require their own transformation algorithm, and due to the options, it is possible to construct different profiles based on the formalisation choices. We will return to this in Sects. 4 and 5.

3 Core Profile

The Core Profile is composed by the elements of the metamodel that belong to the three main families of languages: UML Class Diagrams, (E)ER and ORM/2, and that are extensively used in the analysed models. Interoperability of model semantics between models expressed in these different modelling languages can be assured by restricting models to this set of entities. An important criterion here was to find a 'simple' a language as possible whilst covering the main common entities used in conceptual data models.

- Object Type C. This is represented by concept C in DL.
- Binary Relationship R between object types C and D. This is represented by a DL role R together with the inclusion assertion $\top \sqsubseteq \forall R.C \sqcap \forall R^{-}.D$ to type the relationship. This formalisation reflects the standard view of relationships. We restrict it to binary relationships only, because general n-ary relationships are rarely used in the whole set of analyzed models. (The (E)ER and ORM/2 models exhibit a somewhat higher incidence of n-aries, so they are included in the respective profiles; see below.)
- Attribute a of datatype T within an object type C, including the transformation of ORM's Value type following the transformation rule given in [17]. This is represented in a DL by a role a between concepts C and T, together with the inclusion axiom $C \sqsubseteq \forall a.T$. Formalisation of CDM datatypes in DL as concrete domains or datatypes [3,23] generally translate a datatype into a DL concept, and a datatype value as a DL nominal or instance, which lead to high undecidability results. Although datatypes and concepts share some properties (both can participate in inheritance and conjunction, both can be attached with cardinality constraints), there are also important differences between them: a datatype cannot participate in relationships, cannot be defined by quantifiers or negation over other datatypes, while concepts cannot be composed (which is not the same as union) and cannot be filtered with facets. Identity of a nominal is inherently different as identity of a datatype value, and this is reflected in counting quantifiers.

- Subsumption between two object types C and D. This is represented in DL by the inclusion axiom $C \sqsubseteq D$.
- Object Type cardinality $m..n$ in relationship R with respect to object type. This is represented by the inclusion axiom $C \sqsubseteq \geq n\, R'.\top \sqcap \leq m\, R'.\top$ where R' is either R or R^- depending on C being the first or the second object type in R. R is a unique name in the conceptual model (otherwise qualified cardinality is needed).
- Mandatory constraint. This is a special case of the previous one, with $n = 1$. It is interpreted as $C \sqsubseteq \exists R'.\top$, with R' as before.
- Single identification (in object types with respect to an attribute, and 1:1 mandatory). Let C be an object type identified by attribute a. Then this is interpreted in a DL by the inclusion axioms $C \sqsubseteq \exists a.\top \sqcap \leq 1\, a.\top$ and $\top \sqsubseteq \leq 1\, a^-.C$

In total, all the entities in the core profile sum up to 87.57 % of the entities in all the analysed models, covering The following entities, despite that they belong to all three CDMLs, are not part of the core profile because of their very low participation in the dataset: Role and Relationship Subsumption, Completeness constraints, and Disjointness constraints. Note that this means that it is not possible to express union of concepts in this Core Profile.

Reasoning over this Core Profile is quite simple. Since completeness and disjointness constraints are not present, negation cannot be directly expressed. It is possible to code negation only with cardinality constraints [4, chapter 2], but then we need to reify each negated concept as a new idempotent role. Another form of getting contradiction in this context is by setting several cardinality constraints on the same relationship participation, which is unusual in modelling languages. In any case, the main reasoning problems on the conceptual model only are class subsumption and class equivalence. The description logic $\mathcal{ALNI}$ (which is called $\mathcal{PL}_1$ in [14], and has polynomial subsumption) is expressive enough to represent this profile, since we only need $\top$, $\sqcap$, inverse roles and cardinalities constraints. Its data complexity is unknown.

The core profile shows that a relatively small set of entities concentrates most usage on conceptual models, and that these entities are consistent by assuming just single pairs of maximum and minimum cardinality constraints.

4 Specific CDML Profiles

We describe first the extension of the core so as to cover UML Class Diagrams v2.4.1, and subsequently (E)ER and ORM/2.

4.1 UML Class Diagram Profile

The UML Class Diagram Profile is composed by the Core Profile plus the following entities:

- Shared, Composite Aggregation. No axiomatisation is added for these relationships since the UML 2.4.1 standard [25] does not include additional static constraints, so they are coded as simple binary relationships.
- Subsumption between two UML associations R and S. Since we only have binary relationships, this can be represented in DL as role inclusion $R \sqsubseteq S$.
- Attributive Property Cardinality and Attribute Value Constraint. Cardinalities on attributes can be represented as cardinalities on relationships, but in order to represent value constraints it is necessary to include in the formalisation some datatype facilities to define new datatypes. The attribute is assigned a new datatype which is derived from the original one plus the constraining facets (in terms of XML Schema) on its values.

In total, 99.44 % of all the elements in the analysed UML models are covered by this profile. To formalise this profile in DLs we need to add role hierarchies and datatypes (concrete domains) to the $\mathcal{ALNI}$ logic for the Core Profile. This yields the logic $\mathcal{ALNHI}(\mathcal{D})$ that, as far as we know, has not been studied yet. If we assume unique names and some reasonable (at least from the conceptual modelling point of view) restrictions on the interaction between role inclusions and cardinality constraints, we can represent this profile in $DL\text{-}Lite_{core}^{\mathcal{HN}}$, which is NLOGSPACE for subsumption and AC^0 for data complexity [2].

Typical UML elements like qualified relationship, completeness constraints, and disjointness constraints do not belong to this profile. On the one hand, it is possible to say that the extra expressiveness that is not being used by modellers limit the formal meaning of their models. But since two of these rarely used features are necessary for proving the EXPTIME-hardness of reasoning on UML class diagrams [5], then reasoning over such limited diagrams becomes much more efficient.

4.2 (E)ER Profile

The (E)ER Profile is composed by the Core Profile plus the following entities:

- Composite and Multivalued attribute. Multivalued attributes can be represented with attribute cardinality constraints, and composite attributes with the inclusion of the union datatype derivation operator.
- Weak Object Type, Weak Identification. Each object type (entity type) in (E)ER is assumed by default to have at least one identification constraint. In order to represent external identification, we can use functionality constraints on roles as in $\mathcal{DLR}_{\mathit{ifd}}$ [11], or in $\mathcal{CFD}$ [31].
- Ternary relationships. This is described below and in Algorithm 1.
- Associative Object type. This is formalised by the reification of the association as a new DL concept with two binary relationships.
- Multiattribute identification. This can be formalised as a new composite attribute with single identification.

99.06 % of all the elements in the set of (E)ER models belong to this profile.

The only DL language family with arbitrary n-aries and the advanced identification constraints is $\mathcal{DLR}_{\mathit{ifd}}$, which happens to be positionalist. However, the DL role components are not strictly needed for (E)ER, and one may wish to pursue an n-ary DL without DL role components but with identification constraints, like in the $\mathcal{CFD}$ family of languages. Therefore, we provide here Algorithm 1, which summarises the procedure to go from (E)ER straight to the standard view. The main steps involve binaries vs. higher arities, and recursive ones that generally do have their named relationship components vs 'plain' binaries that have only the relationship named.

Algorithm 1. *(E)ER to standard view and common core*

D_R: domain of R; R_R range of R; n set of R-components
if R is binary **and** $D_R \neq R_R$ **then**
  Rename R to two 'directional' readings, Re_1 and Re_2
  Make Re_1 and Re_2 a DL role each
  Type role with $\top \sqsubseteq \forall Re_1.D_R \sqcap \forall Re_1^-.R_R$
  Declare inverses with $Re_1 \equiv Re_2^-$
else
  if R is binary **and** $D_R = R_R$ **then**
    for all i, with $i \in n$ **do**
      if i is named **then**
        $Re_i \leftarrow i$
      else
        $Re_i \leftarrow$ user-added label or auto generated label
      end if
      Make Re_i a DL role
    end for
    Type one Re_i, i.e., $\top \sqsubseteq \forall Re_i.D_R \sqcap \forall Re_i^-.R_R$
    Declare inverses among all Re_i
  end if
else
  Reify R into $R' \sqsubseteq \top$
  for all i, $3 \geq i \geq n$ **do**
    $Re_i \leftarrow$ user-added label or auto generated label
    Make Re_i a DL role,
    Type Re_i as $\top \sqsubseteq \forall Re_i.R' \sqcap \forall Re_i^-.R_R$, where R_R is the player ((E)ER entity type) in n
    Add $R' \sqsubseteq \exists Re_i.\top$ and $R' \sqsubseteq \leq 1\, Re_i.\top$
  end for
  Add external identifier $\top \sqsubseteq \leq 1\, (\sqcup_i Re_i)^-.R'$
end if

Using this translation, and since we do not have covering constraints in the profile, we can represent the (E)ER Profile in the description logic $DL\text{-}Lite_{core}^{\mathcal{N}}$ [2] which has complexity NLOGSPACE for the satisfiability problem. This low complexity is in no small part thanks to its unique name assumption, whereas

most logics operate under no unique name assumption. A similar result is found in [1] for ER_{ref}, but it excludes composite attributes and weak object types.

4.3 ORM/2 Profile

For ORM, there is no good way to avoid the ORM roles (DL role components), as they are used for several constraints that have to be included. They can be transformed away (discussed below) such that an ORM/2 Profile is obtained by joining the features of the Core Profile. The following entities from the unifying metamodel are added, noting that the starred ones include the formalisation after the transformation from positionalist to standard view:

- Unary role, which is formalised as a boolean attribute. ⋆
- Subsumption between roles; formalised by using DL role hierarchies. ⋆
- n-ary relationships ($n \geq 2$). This is formalised similarly as for (E)ER (see Algorithm 1).
- Subsumption between relationships. This is formalised with an inclusion assertion between the reified concepts.
- Disjoint constraints between ORM roles R and S. This is formalised as two inclusion assertion for roles: $R \sqsubseteq \neg S$ and $S \sqsubseteq \neg R$. ⋆
- Nested object type. The nested object type is identified with the reified concept of the relationship.
- Value constraints. We need to define a new datatype with the constraints, as done in UML profile.
- Disjunctive mandatory constraint for object type C in roles R_i. This is formalised as the inclusion axiom $C \sqsubseteq \sqcup_i \exists R_i$. ⋆
- Internal Uniqueness constraint for roles $R_i, 1 \leq i \leq n$ over relationship objectified with object type R as described below. We need an identification axiom (**id** $C\ 1\,R_1 \ldots 1\,R_n$) as in $\mathcal{DLR}_{\mathit{ifd}}$.
- External Uniqueness constraint between roles $R_i, 1 < i \leq n$ not belonging to the same relationship. Let C be the connected object type between all the R_i, if it exists, or otherwise a new object type representing the reification of a new n-ary relationship between the participating roles. Then we can formalise the constraint with the identification axiom (**id** $C\ 1\,R_1 \ldots 1\,R_n$).
- External identification. This is the same as the previous one, with the exception that we are now sure such C exists (i.e., the mandatoryness is added cf. simple uniqueness).

This profile contains 98.69 % of all the elements in the analysed ORM/2 models. This is still a high coverage considering the assortment of entities available in the language. We decided not to include any ring constraint in this profile. Although the irreflexivity constraint counts for almost half of all appearances of ring constraints, its participation is still too low to be relevant.

In order to formalise this ORM/2 profile we need both n-aries and identification constraints, as in the (E)ER Profile. It differs from the (E)ER profile, in that ORM needs the argument positions for some constraints. We map this positionalist commitment into a standard view. This is motivated by the observation

that typically fact type readings are provided, not user-named ORM role names, and only 9.5 % of all ORM roles in the 33 ORM diagrams in our dataset had a user-defined name, with a median of 0. We process the fact type (relationship R) readings and ignore the role names as follows. $\mathcal{DLR}$'s relationship is typed, w.l.o.g. as binary and in $\mathcal{DLR}$-notation, as $R \sqsubseteq [r_c]C \sqcap [r_d]D$, with r_c and r_d variables for the ORM role names and C and D the participating object types. Let $read_1$ and $read_2$ be the fact type readings, like the teaches and taughtby in Fig. 1, then use $read_1$ to name DL role Re_1 and $read_2$ to name DL role Re_2, and type R as $\top \sqsubseteq \forall Re_1.C \sqcap \forall Re_2.D$. This turns, e.g., a disjoint constraints between ORM roles r_c of relationship R and s_c of S into $Re_1 \sqsubseteq \neg Se_1$ and $Se_1 \sqsubseteq \neg Re_1$.

Concerning complexity of the ORM/2 Profile, this is not clear either. The EXPTIME-complete $\mathcal{DLR}_{\mathit{ifd}}$ is the easiest fit, but contains more than is strictly needed: neither concept disjointness and union are needed (but only among roles), nor its **fd** for complex functional dependencies. The PTIME $\mathcal{CFDI}_{nc}^{\forall -}$ [30] may be a better candidate if we admit a similar translation as the one given in Algorithm 1, but giving up arbitrary number restrictions and disjunctive mandatory on ORM roles.

5 Discussion

As mentioned in Sect. 2.2, other design choices could have led to another 'core profile'. This concerns two choices in particular: i) we used inverses and therewith could avoid qualified cardinality restrictions (thanks to typing of the relationship), and ii) transforming the positionalist into a standard view representation. The advantages are that there are clear indications that the current core profile is computationally better behaved and it can be used more easily with most implemented languages. A disadvantage is that for the ORM/2 profile, a positionalist DL language is needed and for (E)ER, it would make it easier for it fits more nicely with a known language ($\mathcal{DLR}_{\mathit{ifd}}$). Transformations are very well doable, as shown in Algorithm 1, but it adds an extra step in any implementation. The alternative is to create a 'positionalist core', but this is likely to be computationally less well-behaved, and does not enjoy wide software support when it comes to formal characterisations of the CDMLs.

5.1 On Missing and 'Useless' Features

As will be clear, there is no 'ideal' DL language for the CDMLs, not one that captures exactly and only the needed features, that is positionalist to avoid forcing the artificial standard view encoding, and has a usable implementation. The major mismatches regarding implementations have to do with n-aries, DL role components, advanced identifiers, and attributes with their data types, dimensions and value constraints. Data types are being investigated for DLs (e.g., [3,23]), and the results obtained here may serve as a motivational use case. Further, dimensional value types are yet to be addressed; e.g., a ternary 'attribute',

say, height, consisting of the class it is measured for, the data type, and its measurement unit. Others could be a 'nice to have', notably arbitrary n-aries and, for ease of transformation algorithms from CDMLs to a logic, more implementations of positionalism (a DL with DL role components).

Viewed from the perspective as to what can safely be omitted from a logic for CDMLs, then, notably, nominals—computationally costly—are certainly not needed (recall also Sect. 3), and disjointness and completeness are used remarkably few times. Whether the latter is due to a real perceived irrelevance for conceptual data modelling or merely due to unfamiliarity by modellers is a separate line of investigation. The few disjointness and completeness constraints encountered, however, were predominantly in models taken from courses, textbooks, and from the UML standard. Also, there are multiple relationships in the models where properties, such as transitivity and reflexivity, certainly could apply, and if one can declare them (as in ORM) it is done, but it is unclear why it has been done so few times (23 in total in the 33 ORM/2 models). A conjecture is that this is due to their limited implementation support.

5.2 Answering the Research Questions

The results obtained in the previous sections provide the answers to the three questions posed in the introduction. Concerning question 1, the profile of a common core of language features among the main CDMLs has been specified in Sect. 3, covering UML Class diagrams v2.4.1, ER and EER, and ORM and ORM2. Although an important criterion was to keep the logic as 'simple' as possible, it is, perhaps, remarkable that a language with such low expressiveness as $\mathcal{ALNI}$ sufficed when taking into account the intersection of the languages and the usage of the CDML features in actual conceptual data models. $\mathcal{ALNI}$ in in PTIME, and possibly even better computationally well-behaved with the unique name assumption, as no unique name assumption together with number restrictions increases complexity, as shown with the DL-Lite family in [2]. Either way, this makes it certainly promising for scalable implementations for interoperability or conceptual model-mediated analysis and management of large scale data systems, including Ontology-Based Data Access, and to augment query compilation with the 'background knowledge' of the conceptual model, meeting requirements such as aimed for in [6,13,17,28,32].

Regarding question 2 on CDML profiles: based on the profiles defined that took input from the set of 101 conceptual models, there is no optimal language with known complexity that matches exactly a CDML profile to capture each of the UML, (E)ER, and ORM/2 languages (recall Sect. 4), where each profile had its own version of a mismatch. Of the three profiles, the one for UML is closest to the core profile, mainly thanks to the removal of relational properties of the aggregation associations from the UML standard (transitivity and asymmetry were asserted in earlier versions of the standard), and that qualified associations were hardly used.

This brings us to the answer to Question 3 on missing features and too many. A shortcoming of the available DLs is the limited support for constraints on

datatypes. Conversely, nominals, negation and union, and most relational properties do not seem to be needed. Using nominals to encode values is suboptimal (see Sect. 3), whose in-depth argument is omitted due to space limitations.

6 Conclusions

Conceptual data model language-specific profiles for their logic-based reconstruction have been defined, as well as a common core. No CDM profile matches fully with an existing DL language. The common core capturing most entities occurring in the dataset of models amounts to $\mathcal{ALNI}$ which is PTIME. This means that efficient translations between models in these languages can be done preserving most of their elements and meaning. Even for the most expressive CDM language ORM2, the vast majority of entities can be captured with a $\mathcal{DLR}_{\mathit{ifd}}$ without disjointness and union or $\mathcal{CFDI}_{nc}^{\forall -}$ with arbitrary number restrictions. No features are really missing from any DL, other than advanced datatype constraints, but rather tend to have too many constructs. Given the absence of negation, there is little TBox reasoning of interest, other than cardinality constraints. The lean common core and profiles pave the way for a modelling-informed single language for model interoperability, and for their runtime usage in scalable databases and information systems.

Acknowledgments. This work is based upon research supported by the National Research Foundation of South Africa (Project UID90041) and the Argentinean Ministry of Science and Technology.

References

1. Artale, A., Calvanese, D., Kontchakov, R., Ryzhikov, V., Zakharyaschev, M.: Reasoning over extended er models. In: Parent, C., Schewe, K.-D., Storey, V.C., Thalheim, B. (eds.) ER 2007. LNCS, vol. 4801, pp. 277–292. Springer, Heidelberg (2007)
2. Artale, A., Calvanese, D., Kontchakov, R., Zakharyaschev, M.: The DL-Lite family and relations. J. Artif. Intell. Res. **36**, 1–69 (2009)
3. Artale, A., Ryzhikov, V., Kontchakov, R.: DL-Lite with attributes and datatypes. In: Proceedings of ECAI 2012, pp. 61-66. IOS Press (2012)
4. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F. (eds.): The Description Logics Handbook - Theory and Applications, 2nd edn. Cambridge University Press, Cambridge (2008)
5. Berardi, D., Calvanese, D., De Giacomo, G.: Reasoning on UML class diagrams. Artif. Intell. **168**(1–2), 70–118 (2005)
6. Bloesch, A.C., Halpin, T.A.: Conceptual queries using ConQuer-II. In: Embley, D.W. (ed.) ER 1997. LNCS, vol. 1331, pp. 113–126. Springer, Heidelberg (1997)
7. Braga, B.F.B., Almeida, J.P.A., Guizzardi, G., Benevides, A.B.: Transforming OntoUML into Alloy: towards conceptual model validation using a lightweight formal methods. Innov. Sys. Softw. Eng. **6**(1–2), 55–63 (2010)

8. Cadoli, M., Calvanese, D., De Giacomo, G., Mancini, T.: Finite model reasoning on uml class diagrams via constraint programming. In: Basili, R., Pazienza, M.T. (eds.) AI*IA 2007. LNCS (LNAI), vol. 4733, pp. 36–47. Springer, Heidelberg (2007)
9. Calvanese, D., Carbotta, D., Ortiz, M.: A practical automata-based technique for reasoning in expressive description logics. In: Proceedings of IJCAI 2011, pp. 798–804. AAAI Press (2011)
10. Calvanese, D., De Giacomo, G., Lenzerini, M.: On the decidability of query containment under constraints. In: Proceedings of PODS 1998, pp. 149–158 (1998)
11. Calvanese, D., De Giacomo, G., L., M.: Identification constraints and functional dependencies in description logics. In: Proceedings of IJCAI 2001, pp. 155–160. Morgan Kaufmann (2001), seattle, Washington, USA, August 4–10 (2001)
12. Calvanese, D., Lenzerini, M., Nardi, D.: Unifying class-based representation formalisms. J. Artif. Intell. Res. **11**, 199–240 (1999)
13. Calvanese, D., Keet, C.M., Nutt, W., Rodríguez-Muro, M., Stefanoni, G.: Web-based graphical querying of databases through an ontology: the WONDER system. In: Proceedings of ACM SAC 2010, pp. 1389–1396. ACM (2010)
14. Donini, F., Lenzerini, M., Nardi, D., Nutt, W.: Tractable concept languages. In: Proceedings of IJCAI 1991, vol. 91, pp. 458–463 (1991)
15. Fillottrani, P.R., Keet, C.M.: Ontology-driven unification of conceptual data modelling languages (2012–2015). http://www.meteck.org/SAAR.html
16. Fillottrani, P.R., Keet, C.M.: KF metamodel formalisation. Technical report 1412.6545v1 December 2014, arxiv.org
17. Fillottrani, P.R., Keet, C.M.: Conceptual model interoperability: a metamodel-driven approach. In: Bikakis, A., Fodor, P., Roman, D. (eds.) RuleML 2014. LNCS, vol. 8620, pp. 52–66. Springer, Heidelberg (2014)
18. Fine, K.: Neutral relations. Philos. Rev. **109**(1), 1–33 (2000)
19. Keet, C.M., d'Amato, C., Khan, Z.C., Lawrynowicz, A.: Exploring Reasoning with the DMOP Ontology. In: Proceedings of ORE 2014. CEUR-WS, vol. 1207, pp. 64–70 (2014)
20. Keet, C.M., Fillottrani, P.R.: Toward an ontology-driven unifying metamodel for UML class diagrams, EER, and ORM2. In: Ng, W., Storey, V.C., Trujillo, J.C. (eds.) ER 2013. LNCS, vol. 8217, pp. 313–326. Springer, Heidelberg (2013)
21. Keet, C.M., Fillottrani, P.: An analysis and characterisation of publicly available conceptual models. In: Proceedings of ER 2015. LNCS, Springer (2015). (in print)
22. Leo, J.: Modeling relations. J. Phil. Logic **37**, 353–385 (2008)
23. Lutz, C., Areces, C., Horrocks, I., Sattler, U.: Keys, nominals, and concrete domains. J. Artif. Intell. Res. **23**, 667–726 (2005)
24. Motik, B., Grau, B.C., Horrocks, I., Wu, Z., Fokoue, A., Lutz, C.: OWL 2 Web Ontology Language Profiles. W3C recommendation, W3C 27 October 2009. http://www.w3.org/TR/owl2-profiles/
25. Object management group: superstructure specification. standard 2.4.1, object management group (2012). http://www.omg.org/spec/UML/2.4.1/
26. Pan, W., Liu, D.: Mapping object role modeling into common logic interchange format. In: Proceedings of ICACTE 2010, vol. 2, pp. 104–109. IEEE Computer Society (2010)
27. Queralt, A., Artale, A., Calvanese, D., Teniente, E.: OCL-Lite: finite reasoning on UML/OCL conceptual schemas. Data Knowl. Eng. **73**, 1–22 (2012)
28. Smaragdakis, Y., Csallner, C., Subramanian, R.: Scalable satisfiability checking and test data generation from modeling diagrams. Autom. Softw.Eng. **16**, 73–99 (2009)

29. Tobies, S.: Complexity results and practical algorithms for logics in knowledge representation. Ph.D. thesis, RWTH-Aachen, Germany (2001)
30. Toman, D., Weddell, G.: On adding inverse features to the description logic $CFD^{\forall}_{\mathrm{nc}}$. In: Pham, D.-N., Park, S.-B. (eds.) PRICAI 2014. LNCS, vol. 8862, pp. 587–599. Springer, Heidelberg (2014)
31. Toman, D., Weddell, G.E.: Applications and extensions of ptime description logics with functional constraints. In: IJCAI. pp. 948–954 (2009)
32. Toman, D., Weddell, G.E.: Fundamentals of Physical Design and Query Compilation. Synthesis Lectures on Data Management. Morgan & Claypool, San Rafael (2011)
33. Wagih, H.M., Zanfaly, D.S.E., Kouta, M.M.: Mapping object role modeling 2 schemes into $\mathcal{SROIQ}(\mathcal{D})$ description logic. Int. J. of Comp. Theory Eng. **5**(2), 232–237 (2013)