Evidence-based lean logic profiles for conceptual data modelling languages

Pablo Rubén Fillottrani[a,b], C. Maria Keet[c,*]

[a] *Departamento de Ciencias e Ingeniería de la Computación, Universidad Nacional del Sur, Bahía Blanca, Argentina*
[b] *Comisión de Investigaciones Científicas, Provincia de Buenos Aires, Argentina*
[c] *Department of Computer Science, University of Cape Town, Cape Town 7701, South Africa. Tel: +27 (0)21 650 2667*

Abstract

Multiple logic-based reconstruction of conceptual data modelling languages such as EER, UML Class Diagrams, and ORM exists. They mainly cover various fragments of the languages and none are formalised such that the logic applies simultaneously for all three modelling language families as unifying mechanism. This hampers interchangeability, interoperability, and tooling support. In addition, due to the lack of a systematic design process of the logic used for the formalisation, hidden choices permeate the formalisations that have rendered them incompatible.

We aim to address these problems, first, by structuring the logic design process in a methodological way. We generalise and extend the DSL design process to apply to logic language design more generally and, in particular, by incorporating an ontological analysis of language features in the process. Second, availing of this extended process, of evidence gathered of language feature usage, and of computational complexity insights from Description Logics (DL), we specify logic profiles taking into account the ontological commitments embedded in the languages. The profiles characterise the minimum logic structure needed to handle the semantics of conceptual models, enabling the development of interoperability tools. 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.

Keywords: Conceptual modelling, modelling languages, language profiles, modelling language use

1. Introduction

Many conceptual data modelling languages (CDMLs) have been proposed over the past 40 years by several research communities (e.g., relational databases, object-oriented software) and for a range of motivations, such as spatial entities in geographic information systems, ontology-driven or not, and aiming for simplicity and leanness vs expressiveness. Assessing the modelling features of CDMLs over time, it exhibits a general trend toward an increase in modelling features; e.g., UML has identifiers since v2.4.1 [1], ORM 2 has more ring constraints than ORM [2, 3], and EER also supports entity type subsumption and disjointness cf. ER [4, 5]. Opinion varies about this feature richness and its relation to model quality [6] and fidelity of capturing all the constraints specified by the customer, and asking modellers and domain experts which features they think they use, actually use, and need showed discrepancies between them [7]. From the quantitative viewpoint, it has been shown that advanced features are being used somewhere by someone, but they are used relatively very few times, as both the quantitative analysis of CDML feature usage in 168 ORM diagrams [8], and 101 EER, UML, and ORM diagrams has shown [9]. With such insight into feature usage, it is possible to define an *appropriate* logic as underlying foundation of a CDML so as to not only clarify semantics but also use it for computational tasks. Logic-based reconstructions open up the

*Corresponding author
URL: prf@cs.uns.edu.ar (Pablo Rubén Fillottrani), mkeet@cs.uct.ac.za (C. Maria Keet)

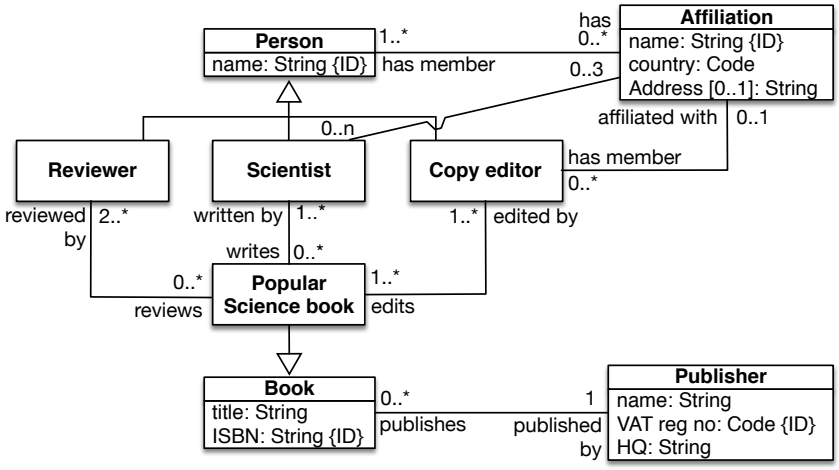

Figure 1: Sample UML Class Diagram containing all possible constraints of the Standard Core Profile, $\mathcal{DC}_s$, which emanated from the evidence-based profile specifications.

opportunity for, among others, automated reasoning over a model to improve its quality (e.g., [10, 11]) and runtime usage of the models, such as conceptual and visual query formulation [12, 13, 14] and optimisation of query compilation [15].

Concerning the 'appropriate' logic, many logic-based reconstructions have been proposed over the years (discussed below), which can be grouped into either the Description Logics based approach that propose logics from viewpoint of computational complexity, rather than needs and usages by modeller, or the as-expressive-as-needed approach, such as in full first-order predicate logic. Neither of these proposals, however, has taken a methodological approach to language design and brush over several thorny details of CDMLs, such as which core types of elements to formalise with their own semantics (aggregation, association ends), whether to include n-aries (when $n \geq 2$, not $n = 2$), and various advanced constraints. This has resulted into an embarrassment of the riches of logic-based reconstructions, which also hampers the actual use of logic-based conceptual data models in information systems for aforementioned tasks and therewith risk sliding into disuse.

We aim to address these problems in this paper. First, we will adapt and extend Frank's DSL design methodology [16] into one suitable for language design more broadly, including conceptual data modelling languages, and informed by language design decision points emanating from ontology. Such ontology-driven language design decisions include, among others, positionalism of relations, the conceptual/computational trade-off and 3-dimensionalist vs. 4-dimensionalist. To the best of our knowledge, this is the first inventarisation of parameters of ontological commitments of language design of information and knowledge representation languages. Second, we apply this to the design of logics for several conceptual data modelling languages that is informed by the language feature usage reported in [9]. These logic 'profiles' cover the most often appearing features used, containing those features that cover about 99% of the features used in conceptual data models. The outcome is a so-called 'positionalist' and a 'standard view' core profile, and three language family profiles formalised in a DL, most of which have a remarkable low computational complexity. An example of UML Class Diagram that can be fully reconstructed into the standard view core profile (more precisely: $\mathcal{DC}_s$) is included in Fig. 1. It has a logical underpinning thanks to the knowledge base construction rules and three algorithms we propose in this paper, and therewith also has grounded equivalents in EER and ORM notation.

This paper builds upon several previous papers on this topic [17, 18, 19, 20, 9]. The main novel contributions in this paper are: i) methodological language design, including ontological aspects involved in it; ii) a new positionalist core profile; and iii) the profiles have been defined with a formal syntax and semantics now

as well. The remainder of the paper is structured as follows. The state of the art and related works are discussed in Section 2. Section 3 presents our first contribution, which is a first inventarisation and discussion of ontological issues that affect language design. Our second main contribution is the logic-based profiles, which are described in Section 4. We close with a discussion in Section 5 and conclusions in Section 6.

2. Related work

Many conceptual data modelling languages have been proposed over the past 40 years; e.g., UML [1], EER [21, 4, 5] and its flavours such as Barker ER and IE notation, ORM [22, 3] and its flavours such as CogNIAM and FCO-IM, MADS [23], and Telos [24]. Some of those are minor variants in notation, whereas others have to a greater or lesser extent a different number of features. Some 'families' of languages can be identified, which basically still run along the lines from which subfield it was originally developed. Notably, ER and EER originate from the relational database community [21], UML Class Diagrams from object-oriented programming [1], and ORM [3] bears similarities with semantic networks and can be used for conceptual modelling for both database and object-oriented programming, and more recently also business rules. Each 'family' has their own set of preferred tools and community of users.

Besides these three main groups, some CDMLs have been developed specifically for additional features (e.g., temporal extensions) or somewhat revised graphical notations of the elements, such as different colours and a so-called 'craw's feet' icon vs ..n or ..* for 'many' multiplicity/cardinality. We will not address this here, but instead will focus on the underlying language features from a logic-based perspective to which the best graphical elements could be attached as 'syntactic sugar' (see, e.g., [25, 26] for this approach), and language design.

2.1. Logic-based reconstructions of CDMLs

A lot of effort has gone into trying to formalise conceptual models for two principal motivations: 1) to be more precise to improve a model's quality and 2) runtime usage of conceptual models. Most works have focussed on the first motivation. Notably, various DLs have been used for giving the graphical elements a formal semantics and for automated reasoning over them, such as [27, 10, 22, 28, 29, 30], although also other logics are being used, including OCL [11], CLIF [31], Alloy [32], and Z [33]. There are also different approaches, which are more interested in the verification problem, notably using some variant of constraint programming [34, 35].

Zooming in on DLs, the $\mathcal{ALUNI}$ language has been used for a partial unification of CDMLs [25], whereas other languages are used for particular modelling language formalisations, such as *DL-Lite* and $\mathcal{DLR}_{\mathsf{ifd}}$ for ER [27] and UML [10], or OWL for ORM and UML [36]. These logic-based reconstructions are typically incomplete with respect to the CDML features they cover, such as omitting ER's 'keys' (identifiers) [25] or n-aries proper [27, 36], among many variants. Also, multiple formalisations in multiple logics for one conceptual modelling language have been published. ORM formalisations can be found in, among others, [37, 22, 28, 38, 36], noting that full ORM and ORM2 (henceforth abbreviated as 'ORM/2') is undecidable due to arbitrary projections over n-aries and the acyclic role constraints (and probably antisymmetry). Even for the more widely-used ER and EER (henceforth abbreviated as '(E)ER'), multiple logic-based reconstructions exist from the modeller's viewpoints [21, 4, 5] and from the logician's vantage points with the $\mathcal{DLR}$ family [39, 40, 41] and *DL-Lite* family [42] of languages.

The second principal reason for formal foundations of CDMLs, runtime usage, comprises a separate track of works, which looks as very lean fragments. The driver for language design here is computational complexity and scalability, and the model is relegated to so-called 'background knowledge' of the system, rather than the prime starting point for software development. Some of the typical runtime usages are: scalable test data generation for verification and validation [43, 8] and ontology-mediated query answering that involves, among others, user-oriented design and execution of queries [44, 12, 13, 14], querying databases during the stage of query compilation [15] and recent spatio-temporal stream queries that avail of ontology-based data access [45, 46] with a variant of *DL-Lite* [42].

In sum, many logics are used for many fragments of the common CDMLs, where the fragments have been chosen for complexity or availability reasons rather than for which features a modeller actually uses.

2.2. Language design

There are many aspects to the design of a representation language, with many choices and decisions. This is not a new challenge and there is the notion of requirements engineering for language design. For instance, a recent paper by [47] proposes the vGREL approach for steps 1 and 2/3 of the overall language design pipeline proposed by Frank [16] and some of the 26 guidelines by [48] are applicable as well. We include here a summary of Frank's waterfall model for domains-specific modelling languages [16], as shown in Figure 2, with the steps we focus on in this paper highlighted. We will step through these states in brief and with respect to applicability and related works on CDML design.

For step 1, the *scope* is conceptual modelling languages, and the *goals* are at least model interoperability and, at least to some extent if feasible, runtime use of conceptual models for so-called 'intelligent systems' or ontology or conceptual-model driven information systems.

Regarding steps 2 and 3, to the best of our knowledge, there is no requirements catalogue for CDMLs (step 2/3a), but several use cases (step 2/3b) have been reported in scientific literature; e.g., the termbanks for multiple languages requiring interoperability among ER and UML [49] and runtime integration of information about food in the Roman Empire with ORM [13]. Assigning priorities (step 2/3c) has been done for several languages, but mostly not explicitly. For instance, a priority may be the computational complexity of the problem of satisfiability checking, or scalability in the presence of large amounts of data (OWL 2 QL) or a scalable TBox (OWL 2 EL) [50], or to have as many language features as possible to cover all corner cases, such as the "narcissist" in medicine as the single example for local reflexivity in $\mathcal{SROIQ}$ [51]. Tables with alternative sets of conflicting requirements are available, such as the pros and cons of several logics for formalising conceptual models [17] and

trade-offs for representing the KGEMT mereotopology in various logics [52]. For the CDMLs, it has yet to be decided how to assign priorities. One could survey industry [53], but it has been shown in at least one survey that modellers do not know the features well enough to be a reliable source [7]. Thus, existing works fall short on providing answers to steps 2 and 3.

There are many papers describing a language specification (step 4), with the DLs the most popular by number of papers. Most of these papers do not have a metamodel, however. Regarding existing metamodels one may be able to reuse for the language specification: there are proposals especially in the conceptual modelling community, spanning theoretical accounts (e.g., [54, 49]), academic proof-of-concept implementations [55, 56, 57], and industry-level applications, such as in the Eclipse Modeling Framework[1]. The UML diagrams in the OWL and UML standards [58, 1] are essentially metamodels as well. To enable a comparison between CDMLs, a unified metamodel is required, which reduces the choice to [55, 56, 49, 57]. The most recent metamodel [49] covers all the static structural components in unifying UML Class Diagrams, ER and EER, and ORM and ORM2 at the metamodel layer cf. the subsets in [55, 56, 57], and has both a glossary of elements and the constraints among them. It was developed in UML Class Diagram notation for communication [49] and formalised for precision [59].

Step 5 in the language design pipeline—design of a graphical notation—is straightforward for the current scope, for it will be mapped onto the graphical notation of UML Class diagrams, EER, and ORM2. The architecture of the tool (step 6) is concurrently being worked on [18], and only after that can one do step 7, which is thus outside of the current scope.

While the 7-step waterfall process for domain-specific languages is generally applicable for logic-based CDML design as well, some ontological analysis during steps 2-4 should improve the outcome, to which we shall return in Section 3. The case for, and benefits of, using insights from ontology (analytic philosophy) to advance the modelling has been well-documented [60, 61], with many papers detailing improvements on precision of representing the information; e.g., deploying the UFO foundational ontology to improve the UML 2.0 metamodel [62] and examining the nature of relationships [63, 64], and more general philosophical assessments about conceptual models, such as regarding concepts [65] and 3D versus 4D conceptual models [66].

[1] `https://www.eclipse.org/modeling/emf/`

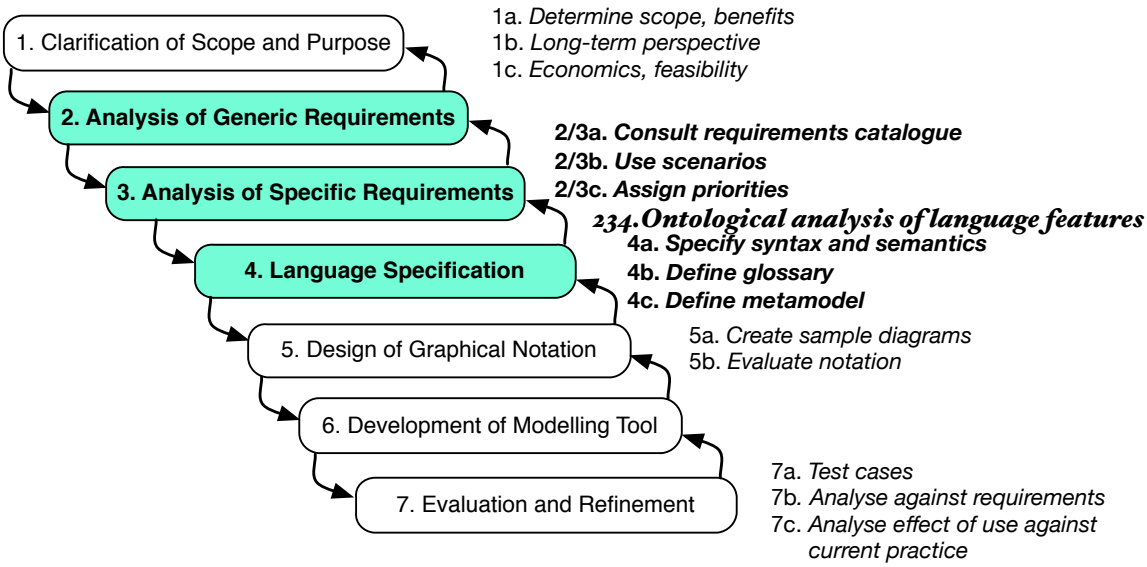

Figure 2: Language design, adapted from [16], where the focus of this paper is highlighted in bold and shared (green). The "234. Ontological analysis of language features" has been added to the 7-step process, which is elaborated on in Section 3.

Thus, current resources fall short especially on a clear requirements specification and priority-setting for CDMLs and on ontology-driven language design. We will contribute to filling these gaps in the following two sections.

2.3. Quantitative assessments on language feature use

To the best of our knowledge, there are only two studies with a quantitative approach to CDML feature usage, which are on ORM diagrams used in industry and developed by one modeller [8] and on publicly available EER, UML, and ORM diagrams [9], whose ORM data are similar to those reported in [8]. The diagrams in the dataset of [9] were analysed using a unified metamodel [49], which facilitated cross-language comparisons and categorisation of entities in those languages into the harmonised terminology. A relevant selection of the terminology across the languages is included in Table 1. This metamodel's top-type is Entity that has four immediate subclasses: Relationship with 11 subclasses, Role, Entity type with 9 subclasses, and Constraint that has 49 subclasses (i.e., across the three CDML families, there are 49 different types of constraint). In addition to the hierarchy, the entities have constraints declared among them to constrain their use; e.g., each relationship must have at least two roles and a disjoint object type constraint is only declared on class subsumptions.

This metamodel was used to classify the entities of the diagrams in a set of 101 UML, (E)ER, and ORM/2 models that were publicly available[2]. The average size of the diagram (vocabulary+subsumption) is about 50 entities/diagram, totalling to 8036 entities, of which 5191 (i.e., 64%) are entities that were classified in an entity (language feature) that is included in all three language families and 1108 (13.8%) in ones that are unique to a language family (e.g., UML's aggregation) [9]. The obtained usage frequency for each entity, together with the design choices described in Section 3, sustain the logic profiles given in Section 4.

3. Design choices for logic-based profiles

One could simply take the evidence of which CDML features are used most, and design a logic for it, or pick a logic one likes and make the best out of a logic-based reconstruction. This has resulted in an

[2]the models and analysis are available from `http://www.meteck.org/swdsont.html`, and not within the scope of this paper

Table 1: Terminology of the languages considered (relevant selection).

metamodel term term	**UML Class Diagram**	**EER**	**ORM/FBM**	**DL**
Relationship	Association	Relationship	Fact type	Role
Role	Association/ member end	Component of a relationship	Role	Role component
Entity type	Classifier	–	Object type	–
Object type	Class	Entity type	Nonlexical object type/Entity type	Concept
Attribute	Attribute	Attribute	–	–
Value type	–	–	Lexical object type/Value type	–
Data type	LiteralSpecification	–	Data type	Concrete domain

'embarrassment of the riches' in the plethora of logic-based reconstructions, and even more so when the formalisations are examined in detail. None is the same. The reason for this is that there are several design choices that each can result in a different logic of different computational complexity, using different reconstruction algorithms, with varying tooling support. Such choices have not been state upfront, but they have to be piecemeal reconstructed by anyone interested in logic foundations for conceptual models, because such decisions are sparsely discussed in the literature. This brings us to the "4" of the "234. Ontological analysis of language features" extension to [16]'s waterfall procedure for language development (recall Fig. 2 in Section 2.2). The step 4, "language specification", concerns affordances and features of the logic, such as 1) the ability to represent the conceptualisation more or less precisely with more or less constraints[3], and 2) whether the language contributes to support, or even shape, the conceptualisation and one's data analysis for the conceptual data model for an information system, or embeds certain philosophical assumptions and positions. Regarding the latter, we identified several decision points, which may not yet be an exhaustive list, but it is the first and most comprehensive list to date. They are as follows, and explained and discussed for CDMLs afterward:

1. Should the CDML be 'truly conceptual' irrespective of the design and implementation or also somewhat computational? That is, whether the language should be constrained to permit representation of only the *what* of the universe of discourse vs. not only *what* but also some *how* in the prospective system. The typical example is whether to include data types for attributes or not.

2. Are the roles that objects play fundamental components of relationships, i.e., should roles be elements of the language?

3. Will refinements of the kinds of general elements—that then have their own representation element—result in a different (better) conceptual model? For instance,

   (a) to have not just Relationship but also an extra element for, say, parthood;

   (b) to have not just Object type but also refinements thereof so as to indicate ontological distinctions between the kind of entities, such as between a rigid object type and the role it plays (e.g., Person vs Student, respectively);

   (c) if only binary relationships are allowed, the modeller may assume there are only binary relations in the world and reifications of n-aries vs the existence of n-aries ($n \geq 2$) proper;

[3]this is distinct from subject domain coverage; to illustrate the difference: being able to represent, say, "has part =2 eyes" vs only "has part $\geq$ 1 eyes" concerns *precision*, whereas omitting information about the eyes concerns *coverage*.

(d) if only object types are allowed, the modeller may assume everything is a object in the world, though one may argue that 'stuff', such as Wine and Gold, is disjoint from object, and thus would have to be represented with a different element.

4. Does one have a 4-dimensionalist view on the world (space-time worms) and thus a language catering for that or are there only 3-dimensional objects with, perhaps, a temporal extension?

5. What must be named? The act of naming or labelling something amounts to identifying it and its relevance; conversely, if it is not named, perhaps it is redundant.

Little is known about what effects the different decisions may have. The data analysis of [9] indicates that binaries vs n-aries (Item 3c) and just plain relationship vs also with aggregation (Item 3a) does indeed make a difference at least for UML vs (E)ER and ORM/2, in that UML class diagrams have disproportionately fewer n-aries and more aggregation associations than (E)ER and ORM/2. Regarding the former, it is known that n-aries in UML class diagrams are hard to read due to the look-across notation [67] and it uses a different visual element (diamond vs line), compared to (E)ER and ORM/2 that use the same notation for both binaries and n-aries; an investigation into the 'syntactic sugar' of the graphical elements is outside the current scope. The other options in Item 3 are philosophically interesting, but unambiguous industry requirements for them are sparse, except for 'stuff', because quantities of amounts of matter are essential to traceability of ingredients in, among others, the food production process and pills and vaccines production (e.g., [68, 69]). In this case, distinguishing between a countable object and a specific quantity of an amount of matter can more precisely relate, say, a specific, numbered, capsule of liquid medicine and the stock that the medicine came from, which, could aid investigations in case of contamination (e.g., the capsule was not cleaned properly vs a contaminated batch of liquid). To date, there is no CDML that includes this distinction other than one proposal for a UML stereotype by [70] and a set of relations for stuffs that could refine UML's aggregation association [71].

The only proposal for refinements of Object Type (Item 3b) is OntoUML [72], which adds, among others, the stereotypes "role" and "phase". More generally, the approach amounts to taking a foundational (top-level) ontology, such as UFO, GFO, or DOLCE, and to use some or all of those core categories to refine UML's class or ER or ORM's entity type. Choosing one foundational ontology may result in the situation where one's diagrams or conceptual data modelling language ends up to be incompatible with one that adheres to another foundational ontology [73]. At the time of writing, it is not clear where or how exactly such refinements provide benefits, other than the typical example of preventing bad hierarchies that OntoClean [74] seeks to resolve. For instance, to not declare, say, Person as a subclass of Employee, because the former is a sortal and the latter is a role that the former plays for a limited duration. Instead, it either should be the other way around (Employee is a Person) or sideways (Employee is a role played by Person). Declaring each Object Type with their respective category and firing ontological rules, such as 'a sortal cannot be subsumed by a role', could then at detect automatically such quality issues in a diagram. From the viewpoint of logics and formalising it, something like this can be done with second-order logic, many-valued logics, or—easier on the logic, but ontologically unclean—have the elements in the diagrams be subsumed by the entities in a foundational ontology.

Conceptual models and features for implementation decisions (Item 1). It is known theoretically at least that incorporating design and implementation decisions reduces potential for interoperability and integration with other systems. Two common examples are declaring data types of attributes (in UML and ORM/2) and entity type-specific attributes (in (E)ER and UML) vs value types that can be shared among entity types (ORM). ER and EER do not have data types, and its selection is pushed down to the design or implementation stage in the waterfall methodologies of database development; e.g., whether some attribute length should be recorded in `integer` or `float` is irrelevant in the data analysis stage. Value types, in contrast to attributes, can be reused by multiple entity types and can easily be reverted into entity types with minimal disruption to the diagram; e.g., a value type length means the same thing regardless whether it is used for the length of a Sofa or a Table for some furniture database. Of course, one can convert between

attributes and value types [17], but having a value type element in the language clearly enables extra analysis regarding common semantics in a model.

Practically, for the design of a logic, the inclusion of value types entails that one has two types of object types: one that can relate only to other object types and one that relates to a data type. That, is not simply only one unary predicate in FOL or 'concept' in DL for everything, but another, disjoint one, that must relate to a data type specifically. The inclusion of attributes, or not, affects the language insofar as one wants to create, or already has, a type of relation that relates an object type to a data type. For instance, OWL's Data Property.

Ontological commitments for relationships (Item 2). Let us first illustrate the possible decisions that Item 2 asks for. Irrespective of the representation decision, there is some relationship, say, teach that holds between Professor and Course. With the 'just predicates' decision, there is assumed to be no teach relationship, or at least not represented as such, but there would be at least one predicate, teaches or taught by in which Professor and Course participate in that specific order or in reverse, respectively. With the 'there are roles too' decision, then Professor would play the role [lecturer] in the relationship teach and Course would play the role [subject] in the relationship teach; hence, role is an element in the language and deemed to be part of the so-called 'fundamental furniture' of the universe. This distinction between predicates-only and roles-too has been investigated by philosophers, and are there called *standard view* and *positionalist*, respectively. It has been argued that the positionalist commitment with roles (argument places) is ontologically a better way characterising relations than the standard view that enforces an artificial ordering of the elements in the relation and requires inverses[4], and it is also deemed better with respect to natural language interaction and expressing more types of constraints [75, 63, 20, 76].

This has been discussed at some length in [17] from a computational perspective, because the main issue is that the CDMLs are all positionalist, yet most logics use predicates with the standard view. To address this impasse, there are several options. One could commit to a logic-based reconstruction into a positionalist language, be this the $\mathcal{DLR}$ family that is used for a partial reconstruction of ER [39], UML class diagrams [10], and ORM [63], roles in the Z formalisation by [33], or one that is yet to be designed. One also could deny positionalism in some way and force it into a standard view logic. For instance, one could change the [lecturer] and [subject] roles into a teaches and/or a taught by predicate and declare them as inverses if both are included in the vocabulary, or pick one of the two and represent the other implicitly through taught by$^-$ or teaches$^-$, respectively. Sampling decisions made in related works showed that, e.g., Pan and Liu [31] use a hybrid of both roles and predicates for ORM and its reading labels also may be 'promoted' to relationships [36], the original ORM formalisation was without roles in the language [22], and UML's association ends are sometimes ignored as well (e.g., [32, 30]), but not always [10].

Exploring the conversion strategies brings one to the computational complexity of the logic. Mostly, adding inverses does not change the worst-case computational complexity of a language; e.g., $\mathcal{ALCQ}$ and $\mathcal{ALCQI}$ are both PSpace-complete [77], but a notable exception is the OWL 2 EL profile that does not have inverse object properties [50].

3D vs. 4D (Item 4). The 3-dimensionalist approach takes objects to be 3-dimensional in space where the objects are wholly present at each point in time (i.e., they do not have temporal parts). Statements are true at the 'present', whilst being ignorant of the object in the past and future. If one wants to deal with time, it is added as an orthogonal dimension, as in, e.g., [78, 79]. The 4-dimensionalist approach, sometimes also called 'fluents', takes entities to exist in four dimensions, being space+time, they are not wholly present at each point in time, and do have temporal parts. Any statements can be about not only the present, but also the past and future (e.g. [80, 66]). A typical example in favour of 4D is to represent as accurately as possible the notion of a holding or supra-organisation [66], such as Alphabet and Nestlé: these companies exist for some time and keep their identity all the while they acquire and sell other (subsidiary) companies. In a 3D-only representation, one would have a record of which companies they own now, but not whether they

[4]and anti-positionalist is argued to be better than positionalist [75, 76], but anti-positionalist is unpractical at this stage.

are the same ones as last year (unless temporal information is added in some way; e.g., through database snapshots or time stamps).

The predominant choice of conceptual data modelling languages is 3D, although temporal extensions do exist especially for ER (e.g., [78, 79]). We could not find any evidence that can explain why this is the case.

Naming things (Item 5). The act of naming things entails the interaction between natural language and ontology, and their interaction. We do not seek to discuss that millennia-old philosophical debate here, but one that is applied within the current context. Naming elements happens differently across the three CDML families. For instance, in UML class diagrams, the association ends must be named but the association does not necessarily have to be named, which typically is the other way around in (E)ER except for recursive relations, whereas ORM diagrams commonly only have *reading labels* of a relationship rather than naming either the roles or the relationship themselves. ORM thus clearly distinguishes between the conceptual layer and the natural language layer, but such workings are not taken into account explicitly in any of the formalisations, nor has it been explicitly decided whether it should be (other than in a model for roles in [20]). A systematic solution to this natural language $\leftrightarrow$ ontology interaction has been investigated in the context of the Semantic Web, where the natural language dimension has its own extension on top of an OWL file [81], although it still relies on naming classes and object properties. What the three CDML families do have in common is that object types are always named, hence, at least part of that separation of ontology and lexicon might be of use in praxis.

For designing a logic, this may not matter much, but it will have an effect on the algorithms to reconstruct the diagrams into a logical theory.

Assessment of popular languages and their commitments. We assessed the relatively frequently occurring logics for formalising conceptual data models on whether it would be possible to choose a 'best fit' language. This comparison is included in Table 2. The first section of the table summarises the main design decisions discussed in the preceding paragraphs, whereas the second part takes into consideration non-ontological aspects with an eye on practicalities. Such practicalities include among others, scalability, tooling support, and whether it would be easily interoperable with other systems. Regarding the latter, we include the Distributed Ontology, Model, and Specification Language DOL that was recently standardised by the Object Management Group, which provides a metalanguage to link models in various logics, including up to second order logic[5].

The first section in the table suggests $\mathcal{DLR}_{\mathit{ifd}}$ and FOL are good candidates for logic-based reconstructions of conceptual data models; the second section has more positive evaluations for $DL\text{-}Lite_{\mathcal{A}}$ and OWL 2 DL. Put differently: neither of them is ideal, so one may as well design one's own language for the best fit.

4. Logic-based profiles for conceptual data modelling languages

We now proceed to define logics to characterise the minimalist semantics of each of the three families of CDMLs. This takes into account the ontological considerations discussed in the previous section as well as the evidence from [9] and the requirement to have a coverage of around 99% of the used entities and constraint. Because of afore-mentioned ontological reasons in favour of roles as well as that all three CDML families are positionalist, a Positionalist Core is defined despite its current lack of implementation support (Section 4.1.1). Afterward, a standard view Standard Core and language-specific profiles are defined in Sections 4.1.2-4.1.5. An overview of the upcoming definitions and algorithms is shown in Fig. 3.

4.1. Profiles

Positionalism is the underlying commitment of the relational model and a database's physical schema, as well as of the main CMDLs. It has been employed in Object-Role Modelling (ORM) and its precursor

[5] `http://www.omg.org/spec/DOL/` and `http://dol-omg.org/`.

Table 2: Popular logics for logic-based reconstructions of CDMLs assessed against a set of requirements; "–": negative evaluation; "+": positive; "NL-logic": natural language interaction with the logic; "OT refinement": whether the language permits second order or multi-value logics or can only do refinement of object types through subsumption.

DL-Lite$_{\mathcal{A}}$	$\mathcal{DLR}_{\mathsf{ifd}}$	**OWL 2 DL**	**FOL**
Language features			
– standard view	+ positionalist	– standard view	– standard view
– with datatypes	– with datatypes	– with datatypes	+ no datatypes
– no parthood primitive	– no parthood primitive	– no parthood primitive	– no parthood primitive
– no n-aries	+ with n-aries	– no n-aries	+ with n-aries
+ 3-dimensionalism	+ 3-dimensionalism	+ 3-dimensionalism	+ 3-dimensionalism
– OT refinement with subsumption	– OT refinement with subsumption	– OT refinement with subsumption	– OT refinement with subsumption
– no NL-logic separation	– no NL-logic separation	± partial NL-logic separation	– no NL-logic separation
– very few features; large feature mismatch	+ little feature mismatch	± some feature mismatch, with overlapping sets	+ little feature mismatch
– logic-based reconstructions to complete	+ logic-based reconstructions exist	– logic-based reconstructions to complete	± logic-based reconstructions exist
Computation and implementability			
+ PTIME (TBox); AC^0 (ABox)	± EXPTIME-complete	± N2EXPTIME-complete	– undecidable
+ very scalable (TBox and ABox)	± somewhat scalable (TBox)	± somewhat scalable (TBox)	– not scalable
+ several reasoners	– no implementation	+ several reasoners	– few reasoners
+ linking with ontologies doable	– no interoperability	+ linking with ontologies doable	– no interoperability with widely used infrastructures
+ 'integration' with DOL	– no integration with DOL	+ 'integration' with DOL	+ 'integration' with DOL
+ modularity infrastructure	– modularity infrastructure	+ modularity infrastructure	– modularity infrastructure

NIAM for the past 40 years [3], UML Class Diagram notation requires association ends as roles, and Entity-Relationship (ER) Models have relationship components [17]. On the other hand, First Order Logic and most of its fragments, notably standard DLs [82], do not exhibit roles (DL role components) among its vocabulary. In order to be able to do reasoning, conceptual schemas written in these CMDLs are generally translated into a DL by removing roles, and thus losing the connection that exists when the same role is played by different concepts. For example, the lives at role may be played by a person in a house rent relationship by a person in a house mortgage relationship. This role name is relevant for querying lets say the real estate owners in the database, so it is relevant for the model's intended meaning but the translation splits them into two different relationships. Therefore, we consider relevant to design a positionalist core profile that preserves roles as first class citizens among the DL vocabulary. In case reasoning over advanced modelling features is needed, it is possible to switch to the standard core profile with the cost of losing this connection. This translation is given in Algorithm 1.

4.1.1. Positionalist Core Profile

In this section we define the logic that describes the positionalist core profile.

Definition 1. *Given a conceptual model in any of the analysed CDMLs, we construct a* knowledge base *in* $\mathcal{DC}_p$ *by applying the rules:*

1. *we take the set all of object types A, binary relationships P, datatypes T and attributes a in the model as the basic elements in the knowledge base.*

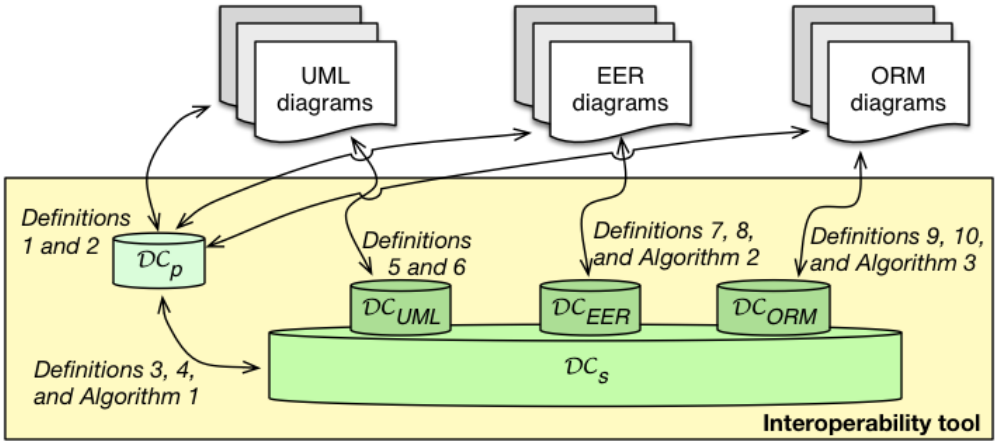

Figure 3: Sketch of the orchestration between the profiles and algorithms.

2. *for each binary relationship P formed by object types A and B, we add to the knowledge base the assertions $\geq 1[1]P \sqsubseteq A$ and $\geq 1[2]P \sqsubseteq B$.*

3. *for each attribute a of datatype T within an object type A, including the transformation of ORM's Value Type following the rule given in [83], we add the assertion $A \sqsubseteq \exists a.T \sqcap \leq 1a$.*

4. *subsumption between two object types A and B is represented by the assertion $A \sqsubseteq B$.*

5. *for each object type cardinality $m..n$ in relationship P with respect to its i-th component A, we add the assertions $A \sqsubseteq \leq n[i]P \sqcap \geq m[i]P$.*

6. *we add for each mandatory constraints of a concept A in a relationship P the axiom $A \sqsubseteq \geq 1[1]P$ or $A \sqsubseteq \geq 1[2]P$ depending on the position played by A in P. This is a special case of the previous one, with $n = 1$.*

7. *for each single identification in object type A with respect to an attribute a of datatype T we add the axiom $\mathsf{id}\, A\, a$.*

This construction is linear in the number of elements in the original conceptual model, so reasoning complexity on the theory is the same as on the conceptual model. We restrict it to binary relationships only, because general n-ary relationships are rarely used in the whole set of analysed 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. Also, we allow only one such constraint for each component, multiple cardinality constraints over the same component in a relationship are used very rarely.

$\mathcal{DC}_p$ can be represented by the following DL syntax. Starting from atomic elements, we can construct binary relations R, arbitrary concepts C and axioms X according to the rules:

$$
\begin{aligned}
C &\longrightarrow \top \,|\, A \,|\, \leq k[i]R \,|\, \geq k[i]R \,|\, \forall a.T \,|\, \exists a.T \,|\, \leq 1\, a \,|\, C \sqcap D \\
R &\longrightarrow \top_2 \,|\, P \,|\, (i : C) \\
X &\longrightarrow C \sqsubseteq D \,|\, \mathsf{id}\, C\, a
\end{aligned}
$$

where $i = 1, 2$ and $0 < k$. For convenience of presentation, we generally use the numbers 1 and 2 to name the role places, but they can be any number or string and do not impose an order. Whenever necessary we note with U the set of all role names in the vocabulary, with $\texttt{from}, \texttt{to} \in U$ fixed argument places for attributes such that [`from`] is the role played by the concept, and [`to`] the role played by the datatype. These names must be locally unique in each relationship/attribute.

Table 3: Semantics of $\mathcal{DC}_p$.

$\top^{\mathcal{I}} \subseteq \Delta_C^{\mathcal{I}}$	$(\leq k[i]R)^{\mathcal{I}} = \{c \in \Delta_C^{\mathcal{I}} \mid \#\{(d_1, d_2) \in R^{\mathcal{I}}.d_i = c\} \leq k\}$
$A^{\mathcal{I}} \subseteq \top^{\mathcal{I}}$	$(\geq k[i]R)^{\mathcal{I}} = \{c \in \Delta_C^{\mathcal{I}} \mid \#\{d_1, d_2) \in R^{\mathcal{I}}.d_i = c\} \geq k\}$
$\top_2^{\mathcal{I}} = \top^{\mathcal{I}} \times \top^{\mathcal{I}}$	$(\exists a.T)^{\mathcal{I}} = \{c \in \Delta_C^{\mathcal{I}} \mid \exists b \in T^{\mathcal{I}}.(c, b) \in a^{\mathcal{I}}\}$
$P^{\mathcal{I}} \subseteq \top_2^{\mathcal{I}}$	$(\forall a.T)^{\mathcal{I}} = \{c \in \Delta_C^{\mathcal{I}} \mid \forall v \in \Delta_T^{\mathcal{I}}.(c, v) \in a^{\mathcal{I}} \rightarrow v \in T^{\mathcal{I}}\}$
$T^{\mathcal{I}} \subseteq \Delta_T^{\mathcal{I}}$	$(\leq 1\, a)^{\mathcal{I}} = \{c \in \Delta_C^{\mathcal{I}} \mid \#\{(c, v) \in a^{\mathcal{I}}\} \leq 1\}$
$a^{\mathcal{I}} \subseteq \top^{\mathcal{I}} \times \Delta_T^{\mathcal{I}}$	$(i : C)^{\mathcal{I}} = \{(d_1, d_2) \in \top_2^{\mathcal{I}} \mid d_i \in C^{\mathcal{I}}\}$
$(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$	

Although this syntax represents all $\mathcal{DC}_p$ knowledge bases, there are sets of formula following the syntactic rules that are not $\mathcal{DC}_p$ knowledge bases since they are not result of any translation of a valid conceptual model. For example, the knowledge base $\{A \sqsubseteq \exists a.T \sqcap \geq 3a\}$ is not a $\mathcal{DC}_p$ knowledge base, because the syntactic production rules are only introduced to provide a proper semantic characterisation.

Now we introduce the semantic characterisation

Definition 2. *An* $\mathcal{DC}_p$ interpretation $\mathcal{I} = (\cdot_{\mathcal{C}}^{\mathcal{I}}, \cdot_{\mathcal{T}}^{\mathcal{I}}, \cdot^{\mathcal{I}})$ *for a knowledge base in* $\mathcal{DC}_p$ *consists of a set of objects* $\Delta_C^{\mathcal{I}}$*, a set of datatype values* $\Delta_T^{\mathcal{I}}$*, and a function* $\cdot^{\mathcal{I}}$ *satisfying the constraints shown in Table 3. It is said that* $\mathcal{I}$ satisfies *the assertion* $C \sqsubseteq D$ *iff* $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$*; and it* satisfies *the assertion* $\mathsf{id}\, C\, a$ *iff exists* T *such that* $C^{\mathcal{I}} \subseteq (\exists a.T \sqcap \leq 1a)^{\mathcal{I}}$ *(mandatory 1) and for all* $v \in T^{\mathcal{I}}$ *it holds that* $\#\{c \mid c \in C^{\mathcal{I}} \wedge (c, v) \in a^{\mathcal{I}}\} \leq 1$ *(inverse functional).*

In total, all the entities in the core profile sum up to 87.57% of the entities in all the analysed models, covering 91,88% of UML models, 73.29% of ORM models, and 94.64% of EE/EER models. Conversely, the following have been excluded from the core despite the feature overlap, due to their low incidence in the model set: Role (DL role component) and Relationship (DL role) Subsumption, and Completeness and Disjointness constraints. This means that it is not possible to express union and disjointness of concepts in a $\mathcal{DC}_p$ knowledge base obtained by formalising a conceptual model. Clearly they can be expressed by combinations of the constructors in $\mathcal{DC}_p$, but this is not possible if we follow the previous construction rules. Since completeness and disjointness constraints are not present, reasoning in this core profile is quite simple.

This logic $\mathcal{DC}_p$ can be directly embedded into $\mathcal{DLR}$ (attributes are treated as binary relationships, and identification constraint over attributes can represented as in [41]) which gives EXPTIME worst case complexity for satisfiability and logical implication. A lower complexity would be expected due to the limitations in the expressivenes. For example, completeness and disjointness constraints are not present, and negation cannot be directly expressed. It is possible to code negation only with cardinality constraints [82, chapter 3], but then we need to reify each negated concept as a new idempotent role, which is not possible to get from the $\mathcal{DC}_p$ rules. Another form of getting contradiction in this context is by setting several cardinality constraints on the same relationship participation, which is also disallowed in the rules. In any case, the main reasoning problems on the conceptual model are only class subsumption and class equivalence on the given set of axioms.

In spite of all these limitations, no simpler positionalist DL has been introduced. In order to get lower complexity bounds we need to translate a $\mathcal{DC}_p$ TBox to a standard (non-positionalist) logic, like $\mathcal{DC}_s$ below.

4.1.2. Standard Core Profile

Considering formalisation choices such as the positionalism of the relationships [63, 76] and whether to use inverses or qualified cardinality constraints, a *standard core profile* has been specified [17]. In case the original context is a positionalist language, a translation into a standard (role-less) language is required. Algorithm 1 (adapted from [17]) does this work in linear time in the number of elements of vocabulary. The main step involves recursive binary relations that generally do have their named relationship components vs 'plain' binaries that have only the relationship named.

Definition 3. *Given a conceptual model in any of the analysed CDMLs, we construct a* knowledge based *in* $\mathcal{DC}_s$ *by applying algorithm 1 to its* $\mathcal{DC}_p$ *knowledge base.*

Algorithm 1 *Positionalist Core to Standard Core*

P an atomic binary relationship; D_P domain of P; R_P range of P
if $D_P \neq R_P$ **then**
  Rename P to two 'directional' readings, Pe_1 and Pe_2
  Make Pe_1 and Pe_2 a DL relation (role)
  Type the relations with $\top \sqsubseteq \forall Pe_1.D_P \sqcap \forall Pe_1^-.R_P$
  Declare inverses with $Pe_1 \equiv Pe_2^-$
else
  if $D_P = R_P$ **then**
    if $i = 1, 2$ is named **then**
      $Pe_i \leftarrow i$
    else
      $Pe_i \leftarrow$ user-added label or auto generated label
    end if
    Make Pe_i a DL relation (role)
    Type one Pe_i, i.e., $\top \sqsubseteq \forall Pe_i.D_P \sqcap \forall Pe_i^-.R_P$
    Declare inverses with $Pe_i \equiv Pe_2^-$
  end if
end if

Table 4: Semantics of $\mathcal{DC}_s$.

$\top^{\mathcal{I}} \subseteq \Delta_C^{\mathcal{I}}$	$(\forall R.A)^{\mathcal{I}} = \{c_1 \in \Delta_C^{\mathcal{I}} \vert \forall c_2.(c_1, c_2) \in R^{\mathcal{I}} \rightarrow c_2 \in A^{\mathcal{I}}\}$
$A^{\mathcal{I}} \subseteq \top^{\mathcal{I}}$	$(\exists R.A)^{\mathcal{I}} = \{c_1 \in \Delta_C^{\mathcal{I}} \vert \exists c_2.(c_1, c_2) \in R^{\mathcal{I}} \wedge c_2 \in A^{\mathcal{I}}\}$
$\top_2^{\mathcal{I}} = \top^{\mathcal{I}} \times \top^{\mathcal{I}}$	$(\leq k\, R)^{\mathcal{I}} = \{c_1 \in \Delta_C^{\mathcal{I}} \vert \#\{c_2 \vert (c_1, c_2) \in R^{\mathcal{I}}\} \leq k\}$
$P^{\mathcal{I}} \subseteq \top_2^{\mathcal{I}}$	$(\geq k\, R)^{\mathcal{I}} = \{c_1 \in \Delta_C^{\mathcal{I}} \vert \#\{c_2 \vert (c_1, c_2) \in R^{\mathcal{I}}\} \geq k\}$
$T^{\mathcal{I}} \subseteq \Delta_T^{\mathcal{I}}$	$(\forall a.T)^{\mathcal{I}} = \{c \in \Delta_C^{\mathcal{I}} \vert \forall v.(c, v) \in a^{\mathcal{I}} \rightarrow v \in T^{\mathcal{I}}\}$
$a^{\mathcal{I}} \subseteq \top^{\mathcal{I}} \times \Delta_T^{\mathcal{I}}$	$(\exists a.T)^{\mathcal{I}} = \{c \in \Delta_C^{\mathcal{I}} \vert \exists v.(c, v) \in a^{\mathcal{I}} \wedge v \in T^{\mathcal{I}}\}$
$(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$	$(\leq 1\, a)^{\mathcal{I}} = \{c \in \Delta_C^{\mathcal{I}} \vert \#\{(c, v) \in a^{\mathcal{I}}\} \vert \leq 1\}$
	$(R^-)^{\mathcal{I}} = \{(c_2, c_1) \in \Delta_C^{\mathcal{I}} \times \Delta_C^{\mathcal{I}} \vert (c_1, c_2) \in R^{\mathcal{I}}\}$

Again, the algorithm is linear in the number of binary relationships in the knowledge base, not affecting complexity results when reasoning.

Once this conversion step is done, the formalisation of the standard core profile is described as follows. It includes inverse relations to keep connected both relationships generated by reifying roles. Take atomic binary relations (P), atomic concepts (A), and simple attributes (a) as the basic elements of the core profile language $\mathcal{DC}_s$, which allows us to construct binary relations and arbitrary concepts according to the following syntax:

$$C \longrightarrow \top_1 \,|\, A \,|\, \forall R.A \,|\, \exists R.A \,|\, \leq k\, R \,|\, \geq k\, R \,|\, \forall a.T \,|\, \exists a.T \,|\, \leq 1\, a.T \,|\, C \sqcap D$$
$$R \longrightarrow \top_2 \,|\, P \,|\, P^-$$
$$X \longrightarrow C \sqsubseteq D \,|\, \mathsf{id}\, C\, a$$

Definition 4. *A* $\mathcal{DC}_s$ interpretation *for a knowledge base in* $\mathcal{DC}_s$ *is given by* $\mathcal{I} = (\cdot_{\mathcal{C}}^{\mathcal{I}}, \cdot_{\mathcal{T}}^{\mathcal{I}}, \cdot^{\mathcal{I}})$*, with* $\Delta_c^{\mathcal{I}}$ *the domain of interpretation for concepts,* $\Delta_T^{\mathcal{I}}$ *the domain of datatype values, and the interpretation function* $\cdot^{\mathcal{I}}$ *satisfying the conditions in Table 4.* $\mathcal{I}$ satisfies *an axiom* X *as in* $\mathcal{DC}_p$*.*

From the perspective of reasoning over $\mathcal{DC}_s$, this is rather simple and little can be deduced: negation cannot be directly expressed here either, as discussed for $\mathcal{DC}_p$. This leaves the main reasoning problem of class subsumption and class equivalence here as well. At most the DL $\mathcal{ALNI}$ (called $\mathcal{PL}_1$ in [84]) is expressive enough to represent this profile, since we only need $\top$, $\sqcap$, inverse roles and cardinality constraints; $\mathcal{PL}_1$ has polynomial subsumption, but its data complexity is unknown. That said, using a similar encoding

of conceptual models as given in Section 4.1.1, the language can be reduced further to $DL\text{-}Lite_{core}^{(\mathcal{HN})}$ which is NLOGSPACE with some restrictions on the interaction between role inclusions and number restrictions, and the Unique name Assumption (UNA). Observe that adding class disjointness to this language would result in a high jump in the complexity, since then the reduction would be to $DL\text{-}Lite_{bool}^{(\mathcal{HN})}$ which is NP-hard [85].

4.1.3. UML Class diagram Profile

The profile for UML Class Diagrams strictly extends $\mathcal{DC}_s$. It was presented extensively in [17].

Definition 5. *A* knowledge base *in $\mathcal{DC}_{UML}$ from a given conceptual model in UML is obtained by adding to its $\mathcal{DC}_s$ knowledge base the following formulas and axioms:*

1. *for each attribute cardinality $m..n$ in an attribute a of datatype T within an object type A we add the assertion $A \sqsubseteq \leq n\, a.T \sqcap \geq m\, a.T$.*
2. *for each binary relationship subsumption between relationships R and S we add the axiom $R \sqsubseteq S$.*

The syntax is as in $\mathcal{DC}_s$, with the additions highlighted in bold face for easy comparison:

$$
\begin{aligned}
C &\longrightarrow \top \mid A \mid \forall R.A \mid \exists R.A \mid \leq k\, R \mid \geq k\, R \mid \forall a.T \mid \exists a.T \\
C &\longrightarrow \boldsymbol{\leq k\, a.T \mid \geq k\, a.T} \mid C \sqcap D \\
R &\longrightarrow \top_2 \mid P \mid P^- \\
X &\longrightarrow C \sqsubseteq D \mid \mathbf{R} \sqsubseteq \mathbf{S} \mid \mathsf{id}\, C\, a
\end{aligned}
$$

With this profile, we cover 99.44% of all the elements in the UML models of the test set. Absence of rarely used UML-specific modelling elements, such as the qualified association (relationship), completeness and disjointness among subclasses do indeed does limit the formal meaning of their models. On the positive side from a computational viewpoint, however, is that adding them to the language bumps up the complexity of reasoning over the models (to EXPTIME-hardness [10]); or: the advantage of their rare use is that reasoning over such limited diagrams has just becomes much more efficient than previously assumed to be needed.

Definition 6. *A $\mathcal{DC}_{UML}$* interpretation *for a $\mathcal{DC}_{UML}$ knowledge base is a $\mathcal{DC}_s$ interpretation $\mathcal{I}$ that also satisfies $R \sqsubseteq S$ if and only if $R^{\mathcal{I}} \subseteq S^{\mathcal{I}}$, with $(\leq k\, a.T)^{\mathcal{I}} = \{c \in \Delta_C^{\mathcal{I}} \mid \#\{a \in T^{\mathcal{I}} \mid (c,a) \in a^{\mathcal{I}}\} \leq k\}$ and $(\geq k\, a.T)^{\mathcal{I}} = \{c \in \Delta_C^{\mathcal{I}} \mid \#\{a \in T^{\mathcal{I}} \mid (c,a) \in a^{\mathcal{I}}\} \geq k\}$.*

Compared to $\mathcal{DC}_s$, role hierarchies have to be added to the $\mathcal{ALNI}$ logic of the Core Profile, which yields the logic $\mathcal{ALNHI}$. To the best of our knowledge, this language has not been studied yet. If we adjust it a little by assuming unique names and some, from the conceptual modelling point of view, reasonable restrictions on the interaction between role inclusions and cardinality constraints, then the UML profile can be represented in the known $DL\text{-}Lite_{core}^{\mathcal{HN}}$, which is NLOGSPACE for subsumption and AC^0 for data complexity [85]. Also, if one wants to add attribute value constraints to this profile then reasoning over concrete domains is necessary. The interaction of inverse roles and concrete domains is known to be highly intractable, just adding them to $\mathcal{ALC}$ gives NEXPTIME-hard concept satisfiability [86].

4.1.4. ER and EER Profile

The profile for ER and EER Diagrams also extends $\mathcal{DC}_s$.

Definition 7. *A* knowledge base *in $\mathcal{DC}_{EER}$ from a given conceptual model in EER is obtained by adding to its its $\mathcal{DC}_s$ knowledge base the following formulas and axioms:*

1. *we include atomic ternary relationships in the basic vocabulary.*
2. *for each attribute cardinality $m..n$ in an attribute a of datatype T within an object type A we add the assertion $A \sqsubseteq \leq n\, a.T \sqcap \geq m\, a.T$.*

3. *for each weak identification of object type A through relationship P in which it participates as the i_3-th component, the we add the assertion* $\mathsf{fd}\, R\, i_1, i_2 \rightarrow i_3$*, such that* $1 \leq i, i_1, i_2 \leq 3$ *and are all different.*

4. *associative object type are formalised by the reification of the association as a new DL concept with two binary relationships.*

5. *multiattribute identification is formalised as a new composite attribute with single identification.*

This profile was presented extensively in [17] and is here recast in shorthand DL notation. The syntax is as in $\mathcal{DC}_s$, with the additions highlighted in bold face for easy comparison:

$$
\begin{aligned}
&C \longrightarrow \top \,|\, A \,|\, \forall R.A \,|\, \exists R.A \,|\, \leq k\, R \,|\, \geq k\, R \,|\, \forall a.T \,|\, \exists a.T \\
&C \longrightarrow \leq \boldsymbol{k}\, a.T \,|\, \geq \boldsymbol{k\, a.T} \,|\, C \sqcap D \\
&R \longrightarrow \top_n \,|\, P \,|\, P^- \\
&X \longrightarrow C \sqsubseteq D \,|\, \mathsf{id}\, C\, a \,|\, \mathsf{fd}\, \mathbf{R}\, \mathbf{i_1}, \mathbf{i_2} \rightarrow \mathbf{i_3}
\end{aligned}
$$

where $n = 2, 3$ and all $i_j = 1, 2, 3$ and different.

Definition 8. *An* interpretation $\mathcal{I}$ *satisfies a knowledge base in* $\mathcal{DC}_{EER}$ *is it is a* $\mathcal{DC}_s$ *interpretation, and satisfies* $\mathsf{fd}\, R\, i_1, i_2 \rightarrow i_3$ *iff for all* $r, s \in R^{\mathcal{I}}$ *it holds that if* $[i_1]r = [i_1]s$ *and* $[i_2]r = [i_2]s$ *then* $[i_3]r = [i_3]s$*, , with* $(\leq k\, a.T)^{\mathcal{I}} = \{c \in \Delta^{\mathcal{I}}_C | \, \#\{a \in T^{\mathcal{I}} | (c, a) \in a^{\mathcal{I}}\} \leq k\}$ *and* $(\geq k\, a.T)^{\mathcal{I}} = \{c \in \Delta^{\mathcal{I}}_C | \, \#\{a \in T^{\mathcal{I}} | (c, a) \in a^{\mathcal{I}}\} \geq k\}$*.*

This profile covers relative frequent EER modelling entities such as composite and multivalued attributes, weak object types and weak identification, ternary relationships, associated objet types and multiattribute identification in addition to those of the standard core profile. This profile can capture 99.06% of all the elements in the set of (E)ER models. Multivalued attributes can be represented with attribute cardinality constraints, and composite attributes with the inclusion of the union datatype derivation operator. 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 (weak object types), we can use functionality constraints on roles as in $\mathcal{DLR}_{\mathsf{ifd}}$ [41] and its close relative $\mathcal{DLR}^+$ [87] or in $\mathcal{CFD}$ [88]. Ternary relationships are explicitly added to the profile. If we want to preserve the identity of these relationships in the DL semantics, then we need to restrict to logics in the $\mathcal{DLR}$ family. Otherwise, it is possible to convert ternaries into concepts by reification, as described in Algorithm 2, using three traditional DL roles and therefore allowing the translation into logics such as $\mathcal{CFD}$. Since associative object types do not impose new static constraints on the models, they are formalised by reification of the association as a new DL concept with two binary relationships. Finally, multiattribute identification can be represented as a new composite attribute with single identification.

This profile presents an interesting challenge regarding existing languages. The only DL language family that has arbitrary n-aries and the advanced identification constraints needed for the weak entity types is the positionalist $\mathcal{DLR}_{\mathsf{ifd}}$. However, $\mathcal{DLR}_{\mathsf{ifd}}$ also offers DL role components that are not strictly needed for (E)ER, so one could pursue a binary or n-ary DL without DL role components but with identification constraints, the latter being needed of itself and for reification of a n-ary into a binary (Algorithm 2). The $\mathcal{CFD}$ family of languages may seem more suitable, then. Using Algorithm 2's translation, and since we do not have covering constraints in the profile, we can represent the (E)ER Profile in the description logic $DL\text{-}Lite^{\mathcal{N}}_{core}$ [85] which has complexity NLOGSPACE for the satisfiability problem. This low complexity is in no small part thanks to its UNA, whereas most logics operate under no unique name assumption. A similar result is found in [27] for ER_{ref}, but it excludes composite attributes and weak object types.

4.1.5. ORM and ORM2 Profile

Unlike for the ER and EER profile, 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. Therefore, to realise this profile, we must transform the ORM positionalist commitment into a standard view, as we did in Algorithm 1. This

Algorithm 2 *Equivalence-preserving n-ary into a binary conversion*

vD_P: domain of P; R_P range of P; n set of P-components
Reify P into $P' \sqsubseteq \top$
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 $\top \sqsubseteq \forall Re_i.P' \sqcap \forall Re_i^-.R_P$, where R_P is the player ((E)ER entity type) in n
  Add $P' \sqsubseteq \exists Re_i.\top$ and $P' \sqsubseteq \leq 1\, Re_i.\top$
end for
Add external identifier $\top \sqsubseteq \leq 1\, (\sqcup_i Re_i)^-.P'$

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 P) readings and ignore the role names following Algorithm 3. $\mathcal{DLR}$'s relationship is typed, w.l.o.g. as binary and in $\mathcal{DLR}$-notation, as $P \sqsubseteq [r_c]C \sqcap [r_d]D$, 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, then use $read_1$ to name DL role Re_1 and $read_2$ to name DL role Re_2, and type P 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 P and s_c of S into $Re_1 \sqsubseteq \neg Se_1$ and $Se_1 \sqsubseteq \neg Re_1$.

Algorithm 3 *ORM/2 to standard view and common core.*

P an atomic relationship
if P is binary **then**
  Take fact type readings F
  if there is only one fact type reading **then**
    $Re_1 \leftarrow F$
    Type Re_1 with domain and range
    Create Re_2
    Declare Re_1 and Re_2 inverses
  else
    Assign one reading to Re_1 and the other to Re_2
    Type Re_1 with domain and range accordingly
    Declare Re_1 and Re_2 inverses
  end if
else
  P is n-ary with $n > 2$
  Reify P into $P' \sqsubseteq \top$, like in Algorithm 2, with for the n binaries using the fact type readings as above
end if

The profile for ORM/2 Diagrams was presented in [17], and a more detailed version including a text-based mapping as a restricted "ORM2$_{cfd}$" was developed in [19] using $\mathcal{CFDI}_{nc}^{\forall -}$ as underlying logic, yet that could cover only just over 96% of the elements in the set of ORM models, whereas this one reaches 98.69% coverage.

Definition 9. *A* knowledge base *in $\mathcal{DC}_{ORM}$ from a given conceptual model in ORM2 is obtained by adding to its $\mathcal{DC}_s$ knowledge base the following formulas and axioms:*

1. *each n-ary relationship is reified as in Algorithm 3.*
2. *each unary role is formalised as a boolean attribute.*
3. *each subsumption between roles R, S is represented by the formula $R \sqsubseteq S$.*

4. *each subsumption between relationships is represented as a subsumption between the reified concepts.*

5. *each disjoint constraint between roles R and S is formalised as two inclusion axioms for roles: $R \sqsubseteq \neg S$ and $S \sqsubseteq \neg R$.*

6. *each nested object type is represented by the reified concept of the relationship.*

7. *each value constraint is represented by a new datatype that constraint.*

8. *each disjunctive mandatory constraint for object type A in roles R_i is formalised as the inclusion axiom $A \sqsubseteq \sqcup_i \exists R_i$.*

9. *each internal uniqueness constraint for roles $R_i, 1 \leq i \leq n$ over relationship objectified with object type A is represented by the axiom $\mathsf{id}\, A\, 1R_1, \ldots, 1R_n$*

10. *each external uniqueness constraint between roles $R_i, 1 < i \leq n$ not belonging to the same relationship can be formalised with the axiom $\mathsf{id}\, A\, 1R_1, \ldots, 1R_n$, where A is 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.*

11. *each external identification is represented as the previous one, with the exception that we are now sure such A exists. (i.e., the mandatoryness is added cf. simple uniqueness).*

This slightly more comprehensive one is here recast in shorthand DL notation, with the additions highlighted in bold face for easy comparison:

$$
\begin{aligned}
&C \longrightarrow \top_1 \,|\, A \,|\, \forall R.A \,|\, \exists R.A \,|\, \leq k\, R \,|\, \geq k\, R \,|\, \forall a.T \,|\, \exists a.T \,|\, \leq 1\, a.T \\
&C \longrightarrow C \sqcap D \,|\, \mathbf{C} \sqcup \mathbf{D} \\
&R \longrightarrow \top_2 \,|\, P \,|\, P^- \,|\, \neg \mathbf{R} \\
&X \longrightarrow C \sqsubseteq D \,|\, \mathbf{R} \sqsubseteq \mathbf{S} \,|\, \mathsf{id}\, C\, a \,|\, \mathsf{id}\, \mathbf{C}\, \mathbf{R_1} \ldots \mathbf{R_n}
\end{aligned}
$$

Definition 10. *A $\mathcal{DC}_{ORM}$* interpretation *for a $\mathcal{DC}_{ORM}$ knowledge base is a $\mathcal{DC}_s$ interpretation $\mathcal{I}$ with the constraints that $(C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}}$, and $(\neg R)^{\mathcal{I}} = \top_2^{\mathcal{I}} \backslash R^{\mathcal{I}}$. $\mathcal{I}$* satisfies *the assertion $R \sqsubseteq S$ iff $(R \sqsubseteq S)^{\mathcal{I}} = R^{\mathcal{I}} \subseteq S^{\mathcal{I}}$, and the assertion $\mathsf{id}\, C\, R_1 \ldots R_n$ iff $C^{\mathcal{I}} \subseteq \cap_i (\exists R_i \sqcap \leq 1 R_i)^{\mathcal{I}}$ and for all objects $d_1, \ldots, d_n \in T^{\mathcal{I}}$ it holds that $\#\{c | c \in C^{\mathcal{I}} \wedge (c, d_i) \in R_i^{\mathcal{I}}, 1 \leq i \leq n\} \leq 1$.*

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.

The semantics, compared to $\mathcal{DC}_s$, is, like with the UML profile, extended in the interpretation for relationship subsumption. It also needs to be extended for the internal uniqueness, with the identification axioms for relationships. Concerning complexity of the ORM/2 Profile, this is not clear either. The EXPTIME-complete $\mathcal{DLR}_{\mathsf{ifd}}$ 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 $\mathcal{CFDI}_{nc}^{\forall -}$ [89] may be a better candidate if we admit a similar translation as the one given in Algorithm 2, but giving up arbitrary number restrictions and disjunctive mandatory on ORM roles.

4.2. Example application of the construction rules

Let us now return to the claim in the introduction about the sample UML Class Diagram in Fig. 1: that it has a logical underpinning in $\mathcal{DC}_s$ and therewith also has grounded equivalents in EER and ORM notation. The equivalents in EER and ORM are shown in Fig. 4.

The first step is to note that the $\mathcal{DC}_s$ reconstruction is obtained from $\mathcal{DC}_p$+ Algorithm 1 (by Definition 3). By the $\mathcal{DC}_p$ rules from Definition 1, we obtain the set of object types (fltr) {Person, Affiliation, ..., Publisher} and of data types {Name, ..., VAT reg no}. For the relationships, we need to use Algorithm 1, which we

illustrate here for the association between Person and Affiliation: 1) bump up the association end names, has member and has, to DL roles; 2) type the relationships with:

$$\top \sqsubseteq \forall \texttt{has\_member}.\texttt{Affiliation} \sqcap \forall \texttt{has\_member}^{-}.\texttt{Person}$$

$$\top \sqsubseteq \forall \texttt{has}.\texttt{Person} \sqcap \forall \texttt{has}^{-}.\texttt{Affiliation}$$

and 3) declare inverses, $\texttt{has\_member} \equiv \texttt{has}^{-}$. After doing this for each association in the diagram, we continue with step 3 of Definition 1, being the attributes. For instance, the Person's Name we obtain the axiom

$$\texttt{Person} \sqsubseteq \exists \texttt{Name}.\texttt{String} \sqcap \leq 1\ \texttt{name}$$

and likewise for the other attributes. Step 4 takes care of the subsumptions; among others $\texttt{Popular\_science\_book} \sqsubseteq \texttt{Book}$ is added to the $\mathcal{DC}_s$ knowledge base. Then cardinalities are processed in steps 5 and 6 (noting the algorithmic conversion from positionalist to standard view applies in this step), so that, for the membership association illustrated above, the following axioms are added to the knowledge base: $\texttt{Affiliation} \sqsubseteq \geq 1\ \texttt{has\_member}$ (mandatory participation) whereas for, say, the scientist, it will be $\texttt{Scientist} \sqsubseteq \leq 3\ \texttt{has}$. Finally, any identifiers are processed, such as ISBN for Book, generating the addition of the $\mathsf{id}\,\texttt{Book}\,\texttt{ISBN}$ to the $\mathcal{DC}_s$ knowledge base.

The process for the EER diagram is the same except that the name of the relationship can be used directly cf. bumping up the role names to relationship names. The reconstruction into ORM has two permutations cf. the UML one, which are covered by step 3 in Definition 1, being the conversion algorithm from ORM's value types to attributes as described in [83], and it passes through the second `else` statement of Algorithm 1 cf. the first `if` statement that we used for UML when going from positionalist to standard view.

Diagram construction rules, i.e., going in the direction from the logic-based profile to a graphical notation, can follow the same process in reverse. This can be achieved automatically, except where labels have to be generated. For instance, if one were to have a scenario on an interoperability tool of "UML diagram $\rightarrow$ $\mathcal{DC}_s$ $\rightarrow$ ORM diagram" and one wants to have the fact type readings, they will have to be added, which a user could write herself or it could be generated by one of the extant realisation engines for the controlled natural language[6], similar to OWL verbalisation [90] or SimpleNLG for natural language generation [91].

5. Discussion

The methodological approach proposed is expected to be of use for similar research to inform better the language design process and elucidate ontological commitments that are otherwise mostly hidden. The five profiles form an orchestrated network of compatible logics, which serve as the logic-based reconstructions of fragments of the three main CDMLs that include their most used features. In the remainder of the section, we discuss language design, computational complexity, and look ahead at applicability.

Language design. To the best of our knowledge, there is no 'cookbook process' for logic or conceptual data modelling language design. Frank's waterfall process [16] provided useful initial guidance for a methodological approach. In our experience in designing the profiles, we deemed our proposed extension with "Ontological analysis of language features" necessary for the conceptual modelling and knowledge representation languages setting cf. Frank's DSLs. An alternative option we considered beforehand was [48]'s list of 26 guidelines, but they are too specific to DSLs to be amenable to CDML design, such as the DSL's distinction between abstract and concrete syntax and their corresponding guidelines.

Zooming into that extra "Ontological analysis of language features" step, we had identified five decision points for language design with respect to ontology and several practical factors that are listed in Table 2 in Section 3. To the best of our knowledge, it is the first attempt to scope this component of language/logic

[6]It would have rules that render, e.g., a `has_member` into ... has member ... and a $\texttt{has\_member}^{-}$ into ... member of ...

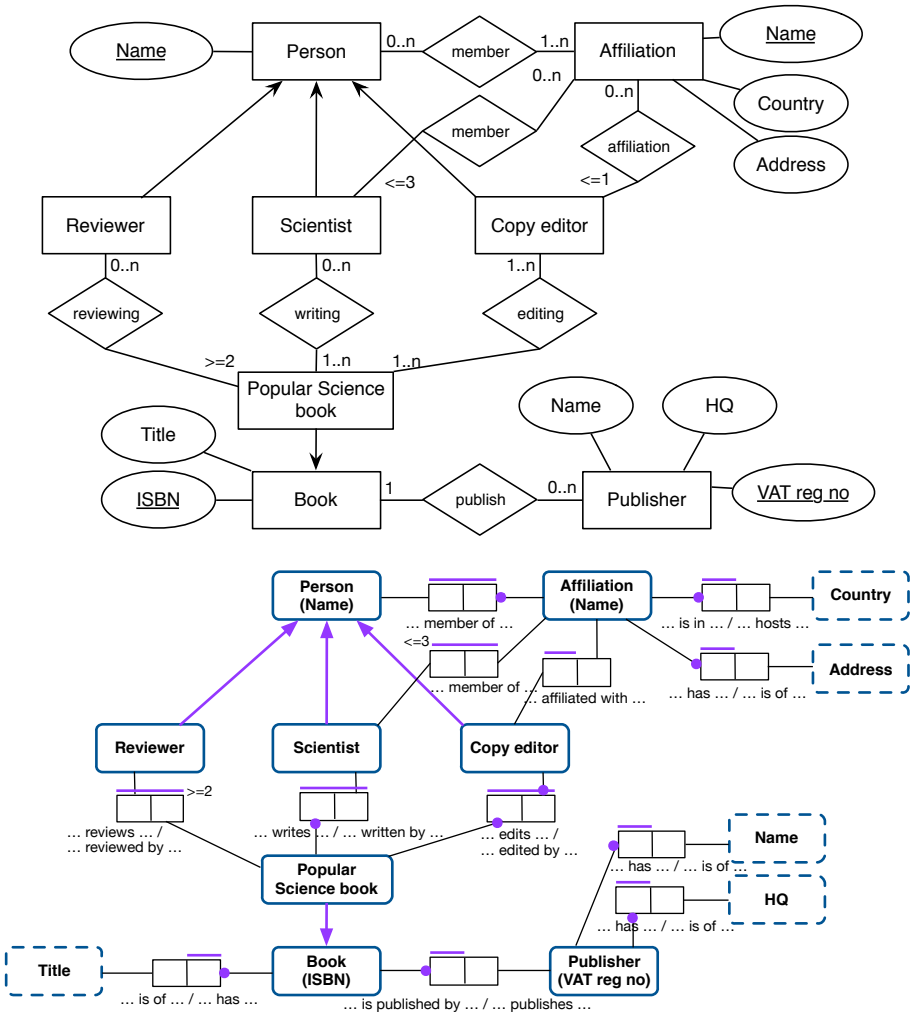

Figure 4: The sample diagram of Fig. 1 rendered in EER and ORM2 notation; the common $\mathcal{DC}_s$ logic-based construction is discussed in the text.

design systematically and it may spur further research into it. Our contribution in that regard should be seen as a starting point for a broader systematic investigation into this hitherto neglected aspect. In making choices, we had to accommodate alternative design choices and the need to achieve high coverage. This was addressed by designing two alternative cores—positionalist and standard view (item 2 in Section 3)—and, importantly, three algorithms to achieve that level of compatibility. More precisely, Algorithm 1 provides the conversion option for item 2—roles or not—in a generic way, Algorithm 2 takes case of the binaries vs n-aries (item 3a), and Algorithm 3 is a specific adaptation of Algorithm 2. All profiles have data types (item 1 in Section 3), for they are present in UML Class Diagrams and ORM/2, noting that it simply can be set to `xsd:anyType` and thus have no influence, which is the case for (E)ER. Further, if the intended semantics of the aggregation association was more specific in the UML standard, it would have merited inclusion in its profile (item 3b in Section 3), with then the onus on the DL community to find a way to add it as a primitive to a DL. If included, it would likely also be possible to design a conversion algorithm between the new primitive and a plain DL role with properties. Regarding adding more types of entity types to the language (item 3c), like sortal and phase: the one proposal [32, 72] is not in widespread use and therewith did not meet the evidence-based threshold for inclusion. It is also not clear how to represent in a decidable

Table 5: Profile comparison on language and complexity; "Approx. DL": the existing DL nearest to the profile defined.

Profile	**Main features**	**Approx. DL**	**Subsumption complexity**
$\mathcal{DC}_p$	positionalist, binary relationships, identifiers, cardinality constraints, attribute typing, mandatory attribute and its functionality	$\mathcal{DLR}$	EXPTIME
$\mathcal{DC}_s$	standard view, binary relationships, inverses	$\mathcal{ALNI}$	P
$\mathcal{DC}_{UML}$	relationship subsumption, attribute cardinality	$DL\text{-}Lite^{\mathcal{HN}}_{core}$	NLOGSPACE
$\mathcal{DC}_{EER}$	ternary relationships, attribute cardinality, external keys	$DL\text{-}Lite^{\mathcal{N}}_{core}$	NLOGSPACE
		$\mathcal{CFD}$	P
$\mathcal{DC}_{ORM}$	entity type disjunction, relationships complement, relationship	$\mathcal{DLR}_{ifd}$	EXPTIME
	subsumption, complex identifiers ('multi attribute keys')	$\mathcal{CFDI}^{\forall-}_{nc}$	P

language such notions that are essentially based on OntoClean [74] that requires modality and higher-order predicates, nor how an equivalence-preserving algorithm would look like, if possible at all.

Complexity considerations for the profiles. Traditionally, the DL research community has strived for identifying more and more expressive DLs for which reasoning is still decidable. The introduced profiles show that high expressivity is not necessary for representing most of the semantics of conceptual models, independently of the chosen modelling language. They thus are 'lean', evidence-based, profiles that, while not covering all corners of modelling issues, do have those features that are used most in practice. We summarise the complexity of each profile by immersion into a DL language in Table 5. The "Approximate DL" column is not an exact match for each profile, and often involves some extra assumptions that explains the different complexities. Low complexities are achievable by the standard profiles (i.e., those that give up on positionalism), due to the existence of a more accurate matching logic. Recall that $\mathcal{DC}_s$ is included in $\mathcal{DC}_{UML}$, $\mathcal{DC}_{EER}$, and $\mathcal{DC}_{ORM}$. The biggest gap between the profiles and the matching DLs is given in $\mathcal{DC}_p$ showing more work on positionalist DLs is necessary, especially with respect to reasoning algorithms.

An outstanding issue is whether object types in the diagrams are by default disjoint when not in a hierarchy, or not. Some research are convinced they are, and some are not; most formalisations and tools do not include it. Because of the lack of agreement, we have not included it. Note further that if this assumption were to be added, i.e., full negation in the profiles, it would affect the computational complexity of the profiles negatively.

It is also interesting to analyse at which point increasing expressiveness by adding new features to the language is worthwhile from the point of view of the modeller. If the feature is present, at least one modeller will use it, though mostly only occasionally. It is not clear if this is due to them being corner cases, a lack of experience on representing advanced constraints by modellers, tooling, or another reason. On the other hand UML's aggregation as 'extra' feature cf. (E)ER's and ORM/2's plain relationships *is* being used disproportionally more often than part-whole relations in (E)ER and ORM/2. It remains to be investigated why exactly this is the case.

Toward applicability. The presented profiles may be applied as the back-end of CASE tools using the compatible profiles as unifying logics and orchestration of corresponding optimised reasoners for, say, Ontology-Based Data Access such that it focusses on the perceived language needs of the modellers (cf. the logic and technology, as in, e.g., [92, 46]), whilst still keeping it tractable. The current conceptual modelling tools that have a logic back-end are still sparse [32, 93, 94, 95], and allow a modeller to model in only one language, rather than being allowed to switch between language families.

Using the common core for model interoperability by mapping each graphical element into a construct in $\mathcal{DC}_s$ is an option. However, one also would want to be precise and therefore use more language features than those in the common core, and when linking models, 'mismatch' links would still need to be managed, and wrong ones discarded. To solve this, an interoperability approach with equivalence, transformation, and approximation rules that is guided by the metamodel is possible [83, 96]. There, one can have two models with an intermodel assertion; e.g., between a UML association and an ORM fact type. The entities are first classified/mapped into entities of the metamodel, any relevant rules are executed, and out comes the result, being either a valid or an invalid link. The 'any relevant rules are executed' is coordinated by the metamodel; e.g., the metamodel states that each Relationship has to have two or more Roles, which, in turn, have to have attached to it either an Object Type or Value Type, so those mapping and transformation rules are called as well during the checking of the link. The MIST EER tool [97] has a similar goal, though currently it supports only EER and its translation to SQL and therewith is complementary to our work presented here.

The formal foundation presented here would enable such an interface were either multiple graphical rendering in different modelling language families could be generated, or link models represented in different languages in a system integration scenario.

6. Conclusions

A systematic logic design process was proposed that generalising and extending the DSL design process to be more broadly applicable by incorporating an ontological analysis of language features in the process. This first compilation of ontological commitments embedded in a logic design process includes, among others, the ontology of relations, the conceptual vs design features trade-off, and 3-dimensionalist vs. 4-dimensionalist commitments.

Based on this extended process with explicit ontological distinctions and the evidence of the prevalence of the features in the models, different characteristic profiles were specified into a suitable Description Logic, which also brought with it insights into their computational complexity. The common core profile is of relatively low computational complexity, being in the tractable $\mathcal{ALNI}$. Without the negation, hardly any inconsistencies can be derived with the profiles, with as flip side that it is promising for scalable runtime use of conceptual data models.

We are looking into several avenues for future work, including ongoing tool development and more precise complexity results for the profiles so that it would allow special, conceptual data model-optimised, reasoners.

Acknowledgements

This work was partially supported by the National Research Foundation of South Africa and the Argentinian Ministry of Science and Technology. Any opinion, findings and conclusions or recommendations expressed in this material are those of the author and therefore the NRF does not accept any liability in regard thereto.

References

[1] Object Management Group, Superstructure Specification, Standard 2.4.1, Object Management Group, 2012. Http://www.omg.org/spec/UML/2.4.1/.
[2] T. Halpin, Information Modeling and Relational Databases, San Francisco: Morgan Kaufmann Publishers, 2001.
[3] T. Halpin, T. Morgan, Information modeling and relational databases, Morgan Kaufmann, 2nd edition, 2008.
[4] I.-Y. Song, P. P. Chen, Entity relationship model, in: L. Liu, M. T. Özsu (Eds.), Encyclopedia of Database Systems, volume 1, Springer, 2009, pp. 1003–1009.
[5] B. Thalheim, Extended entity relationship model, in: L. Liu, M. T. Özsu (Eds.), Encyclopedia of Database Systems, volume 1, Springer, 2009, pp. 1083–1091.
[6] D. L. Moody, Theoretical and practical issues in evaluating the quality of conceptual models: current state and future directions, Data & Knowledge Engineering 55 (2005) 243–276.

[7] R. Alberts, D. Calvanese, G. D. Giacomo, A. Gerber, M. Horridge, A. Kaplunova, C. M. Keet, D. Lembo, M. Lenzerini, M. Milicic, R. Möller, M. Rodríguez-Muro, R. Rosati, U. Sattler, B. Suntisrivaraporn, G. Stefanoni, A.-Y. Turhan, S. Wandelt, M. Wessel, Analysis of Test Results on Usage Scenarios, Deliverable TONES-D27 v1.0, TONES Project, 2008.

[8] Y. Smaragdakis, C. Csallner, R. Subramanian, Scalable satisfiability checking and test data generation from modeling diagrams, Automation in Software Engineering 16 (2009) 73–99.

[9] C. M. Keet, P. R. Fillottrani, An analysis and characterisation of publicly available conceptual models, in: P. Johannesson, M. L. Lee, S. Liddle, A. L. Opdahl, O. Pastor López (Eds.), Proceedings of the 34th International Conference on Conceptual Modeling (ER'15), volume 9381 of *LNCS*, Springer, 2015, pp. 585–593. 19-22 Oct, Stockholm, Sweden.

[10] D. Berardi, D. Calvanese, G. De Giacomo, Reasoning on UML class diagrams, Artificial Intelligence 168 (2005) 70–118.

[11] A. Queralt, A. Artale, D. Calvanese, E. Teniente, OCL-Lite: Finite reasoning on UML/OCL conceptual schemas, Data & Knowledge Engineering 73 (2012) 1–22.

[12] D. Calvanese, C. M. Keet, W. Nutt, M. Rodríguez-Muro, G. Stefanoni, Web-based graphical querying of databases through an ontology: the WONDER system, in: S. Y. Shin, S. Ossowski, M. Schumacher, M. J. Palakal, C.-C. Hung (Eds.), Proceedings of ACM Symposium on Applied Computing (ACM SAC'10), ACM, 2010, pp. 1389–1396. March 22-26 2010, Sierre, Switzerland.

[13] D. Calvanese, P. Liuzzo, A. Mosca, J. Remesal, M. Rezk, G. Rull, Ontology-based data integration in epnet: Production and distribution of food during the roman empire, Engineering Applications of Artificial Intelligence 51 (2016) 212–229.

[14] A. Soylu, E. Kharlamov, D. Zheleznyakov, E. J. Ruiz, M. Giese, M. G. Skjaeveland, D. Hovland, R. Schlatte, S. Brandt, H. Lie, I. Horrocks, OptiqueVQS: a visual query system over ontologies for industry, Semantic Web Journal (2017) in press.

[15] D. Toman, G. E. Weddell, Fundamentals of Physical Design and Query Compilation, Synthesis Lectures on Data Management, Morgan & Claypool, 2011.

[16] U. Frank, Domain-specific modeling languages - requirements analysis and design guidelines, in: I. Reinhartz-Berger, A. Sturm, T. Clark, J. Bettin, S. Cohen (Eds.), Domain Engineering: Product Lines, Conceptual Models, and Languages, Springer, 2013, pp. 133–157.

[17] P. R. Fillottrani, C. M. Keet, Evidence-based languages for conceptual data modelling profiles, in: T. Morzy, et al. (Eds.), 19th Conference on Advances in Databases and Information Systems (ADBIS'15), volume 9282 of *LNCS*, Springer, 2015, pp. 215–229. 8-11 Sept, 2015, Poitiers, France.

[18] P. R. Fillottrani, C. M. Keet, A design for coordinated and logics-mediated conceptual modelling, in: R. Peñaloza, M. Lenzerini (Eds.), Proceedings of the 29th International Workshop on Description Logics (DL'16), volume 1577 of *CEUR-WS*. 22-25 April, 2016, Cape Town, South Africa.

[19] P. R. Fillottrani, C. M. Keet, D. Toman, Polynomial encoding of orm conceptual models in $\mathcal{CFDI}_{nc}^{\forall-}$, in: D. Calvanese, B. Konev (Eds.), Proceedings of the 28th International Workshop on Description Logics (DL'15), volume 1350 of *CEUR-WS*, pp. 401–414. 7-10 June 2015, Athens, Greece.

[20] C. M. Keet, T. Chirema, A model for verbalising relations with roles in multiple languages, in: E. Blomqvist, P. Ciancarini, F. Poggi, F. Vitali (Eds.), Proceedings of the 20th International Conference on Knowledge Engineering and Knowledge Management (EKAW'16), volume 10024 of *LNAI*, Springer, 2016, pp. 384–399. 19-23 November 2016, Bologna, Italy.

[21] P. P. Chen, The entity-relationship model—toward a unified view of data, ACM Transactions on Database Systems 1 (1976) 9–36.

[22] T. Halpin, A logical analysis of information systems: static aspects of the data-oriented perspective, Ph.D. thesis, University of Queensland, Australia, 1989.

[23] C. Parent, S. Spaccapietra, E. Zimányi, Conceptual modeling for traditional and spatio-temporal applications—the MADS approach, Berlin Heidelberg: Springer Verlag, 2006.

[24] J. Mylopoulos, A. Borgida, M. Jarke, M. Koubarakis, Telos: Representing knowledge about information systems, ACM Transactions on Information Systems 8 (1990) 325–362.

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

[26] C. M. Keet, Ontology-driven formal conceptual data modeling for biological data analysis, in: M. Elloumi, A. Y. Zomaya (Eds.), Biological Knowledge Discovery Handbook: Preprocessing, Mining and Postprocessing of Biological Data, Wiley, 2013, pp. 129–154.

[27] A. Artale, D. Calvanese, R. Kontchakov, V. Ryzhikov, M. Zakharyaschev, Reasoning over extended ER models, in: C. Parent, K.-D. Schewe, V. C. Storey, B. Thalheim (Eds.), Proceedings of the 26th International Conference on Conceptual Modeling (ER'07), volume 4801 of *LNCS*, Springer, 2007, pp. 277–292. Auckland, New Zealand, November 5-9, 2007.

[28] A. H. M. t. Hofstede, H. A. Proper, How to formalize it? formalization principles for information systems development methods, Information and Software Technology 40 (1998) 519–540.

[29] K. Kaneiwa, K. Satoh, Consistency checking algorithms for restricted UML class diagrams., in: Proceedings of the 4th International Symposium on Foundations of Information and Knowledge Systems (FoIKS'06), Springer Verlag, 2006.

[30] A. Queralt, E. Teniente, Decidable reasoning in UML schemas with constraints, in: Z. Bellahsene, M. Léonard (Eds.), Proceedings of the 20th International Conference on Advanced Information Systems Engineering (CAiSE'08), volume 5074 of *LNCS*, Springer, 2008, pp. 281–295. Montpellier, France, June 16-20, 2008.

[31] W.-L. Pan, D.-x. Liu, Mapping object role modeling into common logic interchange format, in: Proceedings of the 3rd International Conference on Advanced Computer Theory and Engineering (ICACTE'10), volume 2, IEEE Computer Society, 2010, pp. 104–109.

[32] B. F. B. Braga, J. P. A. Almeida, G. Guizzardi, A. B. Benevides, Transforming OntoUML into Alloy: towards conceptual

model validation using a lightweight formal methods, Innovations in Systems and Software Engineering 6 (2010) 55–63.
[33] A. Jahangard Rafsanjani, S.-H. Mirian-Hosseinabadi, A Z Approach to Formalization and Validation of ORM Models, in: E. Ariwa, E. El-Qawasmeh (Eds.), Digital Enterprise and Information Systems, volume 194 of *CCIS*, Springer, 2011, pp. 513–526.
[34] M. Cadoli, D. Calvanese, G. De Giacomo, T. Mancini, Finite model reasoning on UML class diagrams via constraint programming, in: Proc. of AI*IA 2007, volume 4733 of *LNAI*, Springer, 2007, pp. 36–47.
[35] J. Cabot, R. Clarisó, D. Riera, Verification of UML/OCL class diagrams using constraint programming, in: Model Driven Engineering, Verification, and Validation: Integrating Verification and Validation in MDE (MoDeVVA 2008).
[36] H. M. Wagih, D. S. E. Zanfaly, M. M. Kouta, Mapping Object Role Modeling 2 schemes into $\mathcal{SROIQ}(d)$ description logic, International Journal of Computer Theory and Engineering 5 (2013) 232–237.
[37] E. Franconi, A. Mosca, D. Solomakhin, The formalisation of ORM2 and its encoding in OWL2, KRDB Research Centre Technical Report KRDB12-2, Faculty of Computer Science, Free University of Bozen-Bolzano, Italy, 2012.
[38] C. M. Keet, Mapping the Object-Role Modeling language ORM2 into Description Logic language $\mathcal{DLR}_{ifd}$, Technical Report 0702089v2, KRDB Research Centre, Free University of Bozen-Bolzano, Italy, 2009. ArXiv:cs.LO/0702089v2.
[39] D. Calvanese, G. De Giacomo, M. Lenzerini, On the decidability of query containment under constraints, in: Proc. of the 17th ACM SIGACT SIGMOD SIGART Sym. on Principles of Database Systems (PODS'98), pp. 149–158.
[40] D. Calvanese, G. De Giacomo, M. Lenzerini, Reasoning in expressive description logics with fixpoints based on automata on infinite trees, in: Proc. of the 16th Int. Joint Conf. on Artificial Intelligence (IJCAI'99), pp. 84–89.
[41] D. Calvanese, G. De Giacomo, M. Lenzerini, Identification constraints and functional dependencies in description logics, in: B. Nebel (Ed.), Proc. of the 17th Int. Joint Conf. on Artificial Intelligence (IJCAI 2001), Morgan Kaufmann, 2001, pp. 155–160. Seattle, Washington, USA, August 4-10, 2001.
[42] D. Calvanese, G. D. Giacomo, D. Lembo, M. Lenzerini, R. Rosati, Tractable reasoning and efficient query answering in description logics: The DL-Lite family, Journal of Automated Reasoning 39 (2007) 385–429.
[43] M. Nizol, L. K. Dillon, R. E. K. Stirewalt, Toward tractable instantiation of conceptual data models using non-semantics-preserving model transformations, in: Proceedings of the 6th International Workshop on Modeling in Software Engineering (MiSE'14), ACM Conference Proceedings, 2014, pp. 13–18. Hyderabad, India, June 02-03, 2014.
[44] A. C. Bloesch, T. A. Halpin, Conceptual Queries using ConQuer-II, in: Proceedings of ER'97: 16th International Conference on Conceptual Modeling, volume 1331 of *LNCS*, Springer, 1997, pp. 113–126.
[45] T. Eiter, J. X. Parreira, P. Schneider, Spatial ontology-mediated query answering over mobility streams, in: E. Blomqvist, et al. (Eds.), Proceedings of the 13th Extended Semantic Web Conference (ESWC'17), volume 10249 of *LNCS*, Springer, 2017, pp. 219–237. 30 May - 1 June 2017, Portoroz, Slovenia.
[46] Ö. L. Özçep, R. Möller, C. Neuenstadt, Stream-query compilation with ontologies, in: B. Pfahringer, J. Renz (Eds.), Proceedings of the 28th Australasian Joint Conference on Advances in Artificial Intelligence (AI'15), volume 9457 of *LNCS*, Springer, 2015, pp. 457–463. Canberra, ACT, Australia, November 30 – December 4, 2015.
[47] S. de Kinderen, Q. Ma, Requirements engineering for the design of conceptual modeling languages, Applied Ontology 10 (2015) 7–24.
[48] G. Karsai, H. Krahn, C. Pinkernell, B. Rumpe, M. Schindler, S. Völkel, Design guidelines for Domain Specific Languages, in: Proceedings of the 9th OOPSLA Workshop on Domain-Specific Modeling (DSM'09). Orlando, Florida, USA, October 2009.
[49] C. M. Keet, P. R. Fillottrani, An ontology-driven unifying metamodel of UML Class Diagrams, EER, and ORM2, Data & Knowledge Engineering 98 (2015) 30–53.
[50] B. Motik, B. C. Grau, I. Horrocks, Z. Wu, A. Fokoue, C. Lutz, OWL 2 Web Ontology Language Profiles, W3C Recommendation, W3C, 2009. Http://www.w3.org/TR/owl2-profiles/.
[51] I. Horrocks, O. Kutz, U. Sattler, The even more irresistible $\mathcal{SROIQ}$, Proceedings of KR-2006 (2006) 452–457.
[52] C. M. Keet, F. C. Fernández-Reyes, A. Morales-González, Representing mereotopological relations in OWL ontologies with ONTOPARTS, in: E. Simperl, et al. (Eds.), Proceedings of the 9th Extended Semantic Web Conference (ESWC'12), volume 7295 of *LNCS*, Springer, 2012, pp. 240–254. 29-31 May 2012, Heraklion, Crete, Greece.
[53] I. Malavolta, P. Lago, H. Muccini, P. Pelliccione, A. Tang, What industry needs from architectural languages: A survey, IEEE Transactions on Software Engineering 39 (2013) 869–891.
[54] T. A. Halpin, Advanced Topics in Database Research, volume 3, Idea Publishing Group, Hershey PA, USA, pp. 23–44.
[55] P. Atzeni, P. Cappellari, R. Torlone, P. A. Bernstein, G. Gianforme, Model-independent schema translation, VLDB Journal 17 (2008) 1347–1370.
[56] M. Boyd, P. McBrien, Comparing and transforming between data models via an intermediate hypergraph data model, Journal on Data Semantics IV (2005) 69–109.
[57] J. Venable, J. Grundy, Integrating and supporting Entity Relationship and Object Role Models, in: M. P. Papazoglou (Ed.), Proceedings of the 14th International Conference on Object-Oriented and Entity-Relationship Modelling (ER'95), volume 1021 of *LNCS*, Springer, 1995, pp. 318–328. Gold Coast, Australia, December 12-15, 1995.
[58] B. Motik, P. F. Patel-Schneider, B. Parsia, OWL 2 Web Ontology Language Structural Specification and Functional-Style Syntax, W3C Recommendation, W3C, 2009. Http://www.w3.org/TR/owl2-syntax/.
[59] P. R. Fillottrani, C. M. Keet, KF metamodel formalization, Technical Report 1078634, 2014. Arxiv.org, 21p.
[60] N. Guarino, The ontological level: Revisiting 30 years of knowledge representation, in: A. Borgida, et al. (Eds.), Mylopoulos Festschrift, volume 5600 of *LNCS*, Springer, 2009, pp. 52–67.
[61] N. Guarino, G. Guizzardi, In the defense of ontological foundations for conceptual modeling, Scandinavian Journal of Information Systems 18 (2006) (debate forum, 9p).
[62] G. Guizzardi, G. Wagner, Using the unified foundational ontology (UFO) as a foundation for general conceptual modeling

languages, in: Theory and Applications of Ontology: Computer Applications, Springer, 2010, pp. 175–196.
[63] C. M. Keet, Positionalism of relations and its consequences for fact-oriented modelling, in: R. Meersman, P. Herrero, D. T. (Eds.), OTM Workshops, International Workshop on Fact-Oriented Modeling (ORM'09), volume 5872 of *LNCS*, Springer, 2009, pp. 735–744. Vilamoura, Portugal, November 4-6, 2009.
[64] G. Guizzardi, G. Wagner, What's in a relationship: An ontological analysis, in: Q. Li, S. Spaccapietra, E. S. K. Yu, A. Olivé (Eds.), Proceedings of the 27th International Conference on Conceptual Modeling (ER'08), volume 5231 of *LNCS*, Springer, 2008, pp. 83–97. Barcelona, Spain, October 20-24, 2008.
[65] C. Partridge, C. Gonzalez-Perez, B. Henderson-Sellers, Are conceptual models concept models?, in: W. Ng, V. C. Storey, J. Trujillo (Eds.), 32nd International Conference on Conceptual Modeling (ER'13), volume 8217 of *LNCS*, Springer, 2013, pp. 96–105. 11-13 November, 2013, Hong Kong.
[66] M. West, C. Partridge, M. Lycett, Enterprise data modelling: Developing an ontology-based framework for the shell downstream business, in: R. Cuel, R. Ferrario (Eds.), Proceedings of Formal Ontologies Meet industry (FOMI'10), pp. 71–84. 14-15 December 2010, Trento, Italy.
[67] P. Shoval, S. Shiran, Entity-relationship and object-oriented data modeling—an experimental comparison of design quality, Data and Knowledge Engineering 21 (1997) 297–315.
[68] K. A.-M. Donnelly, A short communication - meta data and semantics the industry interface: what does the food industry think are necessary elements for exchange?, in: Metadata and Semantic Research: 4th International Conference, MTSR 2010.
[69] M. Solanki, C. Brewster, OntoPedigree: modelling pedigrees for traceability in supply chains, Semantic Web Journal 7 (2016) 483–491.
[70] G. Guizzardi, On the representation of quantities and their parts in conceptual modeling, in: Proceedings of 6th International conference on Formal Ontology in Information Systems (FOIS'10), IOS Press, 2010. 11th-14th May 2010, Toronto, Canada.
[71] C. M. Keet, Relating some stuff to other stuff, in: E. Blomqvist, P. Ciancarini, F. Poggi, F. Vitali (Eds.), Proceedings of the 20th International Conference on Knowledge Engineering and Knowledge Management (EKAW'16), volume 10024 of *LNAI*, Springer, 2016, pp. 368–383. 19-23 November 2016, Bologna, Italy.
[72] G. Guizzardi, Ontological Foundations for Structural Conceptual Models, Phd thesis, University of Twente, The Netherlands. Telematica Instituut Fundamental Research Series No. 15, 2005.
[73] Z. Khan, C. M. Keet, The foundational ontology library ROMULUS, in: A. Cuzzocrea, S. Maabout (Eds.), Proceedings of the 3rd International Conference on Model & Data Engineering (MEDI'13), volume 8216 of *LNCS*, Springer, 2013, pp. 200–211. September 25-27, 2013, Amantea, Calabria, Italy.
[74] N. Guarino, C. Welty, An overview of OntoClean, in: S. Staab, R. Studer (Eds.), Handbook on Ontologies, Springer Verlag, 2009, pp. 201–220.
[75] K. Fine, Neutral relations, The Philosophical Review 109 (2000) 1–33.
[76] J. Leo, Modeling relations, Journal of Philosophical Logic 37 (2008) 353–385.
[77] S. Tobies, Complexity Results and Practical Algorithms for Logics in Knowledge Representation, Ph.D. thesis, RWTH Aachen, 2001.
[78] A. Artale, C. Parent, S. Spaccapietra, Evolving objects in temporal information systems, Annals of Mathematics and Artificial Intelligence 50 (2007) 5–38.
[79] C. M. Keet, S. Berman, Determining the preferred representation of temporal constraints in conceptual models., in: H. Mayr, et al. (Eds.), 36th International Conference on Conceptual Modeling (ER'17), volume 10650 of *LNCS*, Springer, 2017, pp. 437–450. 6-9 Nov 2017, Valencia, Spain.
[80] S. Batsakis, E. Petrakis, I. Tachmazidis, G. Antoniou, Temporal representation and reasoning in OWL 2, Semantic Web Journal 8 (2017) 981–1000.
[81] P. Buitelaar, P. Cimiano (Eds.), Towards the Multilingual Semantic Web: Principles, Methods and Applications, Springer, 2014.
[82] F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, P. F. Patel-Schneider (Eds.), The Description Logics Handbook – Theory and Applications, Cambridge University Press, 2 edition, 2008.
[83] P. R. Fillottrani, C. M. Keet, Conceptual model interoperability: a metamodel-driven approach, in: A. Bikakis, et al. (Eds.), Proceedings of the 8th International Web Rule Symposium (RuleML'14), volume 8620 of *LNCS*, Springer, 2014, pp. 52–66. August 18-20, 2014, Prague, Czech Republic.
[84] F. Donini, M. Lenzerini, D. Nardi, W. Nutt, Tractable concept languages., in: Proc.of IJCAI'91, volume 91, pp. 458–463.
[85] A. Artale, D. Calvanese, R. Kontchakov, M. Zakharyaschev, The DL-Lite family and relations, Journal of Artificial Intelligence Research 36 (2009) 1–69.
[86] F. Baader, S. Brandt, C. Lutz, Pushing the EL envelope, in: L. P. Kaelbling, A. Saffiotti (Eds.), IJCAI-05, Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence, Edinburgh, Scotland, UK, July 30 - August 5, 2005, Professional Book Center, 2005, pp. 364–369.
[87] A. Artale, E. Franconi, R. Peñaloza, F. Sportelli, A decidable very expressive description logic for databases, in: C. d'Amato, M. Fernandez, V. Tamma, F. Lecue, P. Cudré-Mauroux, J. Sequeda, C. Lange, J. Heflin (Eds.), The Semantic Web – ISWC 2017: 16th International Semantic Web Conference, volume 10587 of *LNCS*, Springer, Cham, 2017, pp. 37–52. 21–25 October 2017, Vienna, Austria.
[88] D. Toman, G. E. Weddell, Applications and extensions of PTIME Description Logics with functional constraints, in: Proceedings of the 21st International Joint Conference on Artificial Intelligence IJCAI'09, AAAI Press, 2009, pp. 948–954.
[89] D. Toman, G. E. Weddell, On adding inverse features to the description logic $CFD^{\forall}_{\mathrm{nc}}$, in: PRICAI 2014: Trends in Artificial Intelligence - 13th Pacific Rim International Conference on Artificial Intelligence, Gold Coast, QLD, Australia,

December 1-5, 2014., pp. 587–599.
[90] H. Safwat, B. Davis, CNLs for the semantic web: a state of the art, Language Resources & Evaluation 51 (2017) 191–220.
[91] A. Gatt, E. Reiter, Simplenlg: A realisation engine for practical applications, in: E. Krahmer, M. Theune (Eds.), Proceedings of the 12th European Workshop on Natural Language Generation (ENLG'09), ACL, 2009, pp. 90–93. March 30-31, 2009, Athens, Greece.
[92] D. Calvanese, B. Cogrel, S. Komla-Ebri, R. Kontchakov, D. Lanti, M. Rezk, M. Rodriguez-Muro, G. Xiao, Ontop: Answering SPARQL queries over relational databases, Semantic Web Journal 8 (2017) 471–487.
[93] C. Farré, A. Queralt, G. Rull, E. Teniente, T. Urpí, Automated reasoning on UML conceptual schemas with derived information and queries, Information and Software Technology 55 (2013) 1529 – 1550.
[94] P. R. Fillottrani, E. Franconi, S. Tessaris, The ICOM 3.0 intelligent conceptual modelling tool and methodology, Semantic Web Journal 3 (2012) 293–306.
[95] G. A. Braun, C. Giménez, P. R. Fillottrani, L. A. Cecchi, Towards conceptual modelling interoperability in a web tool for ontology engineering, in: Proceedings of the 3rd Argentine Symposium on Ontologies and their Applications co-located with 46 Jornadas Argentinas de Informática (46JAIIO), pp. 25–38.
[96] Z. C. Khan, C. M. Keet, P. R. Fillottrani, K. Cenci, Experimentally motivated transformations for intermodel links between conceptual models, in: J. Pokorný, et al. (Eds.), 20th Conference on Advances in Databases and Information Systems (ADBIS'16), volume 9809 of *LNCS*, Springer, 2016, pp. 104–118. August 28-31, Prague, Czech Republic.
[97] V. Dimitrieski, M. Celikovic, S. Aleksic, S. Risti, A. Alargt, I. Lukovic, Concepts and evaluation of the extended entity-relationship approach to database design in a multi-paradigm information system modeling tool, Computer Languages, Systems & Structures 44 (2015) 299 – 318.