- ORIGINAL ARTICLE -

Evidence-based Lean Conceptual Data Modelling Languages

Lenguajes Austeros de Modelado Conceptual de Datos Basados en Evidencias

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

[1] *Departamento de Ciencias e Ingeniería de la Computación, Universidad Nacional del Sur, Bahía Blanca, Argentina*
prf@cs.uns.edu.ar

[2] *Comisión de Investigaciones Científicas de la Provincia de Buenos Aires*

[3] *Department of Computer Science, University of Cape Town, South Africa*
mkeet@cs.uct.ac.za

Abstract

Multiple logic-based reconstructions of UML class diagram, Entity Relationship diagrams, and Obect-Role Model diagrams exists. They mainly cover various fragments of these Conceptual Data Modelling Languages and none are formalised such that the logic applies simultaneously for the three language families as a 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 logic language design. In particular, a new phase of ontological analysis of language features is included, to apply to logic language design more generally and, in particular, by incorporating an ontological analysis of language features in the process. Second, we specify minimal logic profiles availing of this extended process, including the ontological commitments embedded in the languages, of evidence gathered of language feature usage, and of computational complexity insights from Description Logics (DL). The profiles characterise the essential logic structure needed to handle the semantics of conceptual models, therewith enabling the development of interoperability tools. No known DL language matches exactly the features of those profiles and the common core is in the tractable DL $\mathcal{ALNI}$. Although hardly any inconsistencies can be derived with the profiles, it is promising for scalable runtime use of conceptual data models.

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

Resumen

Existen varias reconstrucciones basadas en lógica de lenguajes de modelado conceptual como EER, diagramas de clases UML y ORM. Principalmente cubren fragmentos de estos lenguajes, y sus formalizaciones no están hechas para que se apliquen simultáneamente a estas tres familias de lenguajes como un mecanismo de unificación. Este hecho atenta contra el intercambio y la interoperabilidad de los modelos y el desarrollo de herramientas de soporte. Además, dada la falta de un proceso sistemático de diseño, ciertas decisiones ocultas en la representación lógica hacen que las formalizaciones sean incompatibles. En este trabajo nos proponemos atacar este problema, proponiendo primero un proceso de diseño lógico que puede ser aplicado en forma metodológica. Se generaliza y extiende el proceso DSL para que se pueda aplicar al diseño de lenguajes lógicos en general, incorporando análisis ontológico de las características del lenguaje. Segundo, se especifican perfiles lógicos minimales que sacan provecho de este proceso extendido, incluyendo los compromisos ontológicos asumidos, de evidencia de uso de las características del lenguaje, y de los propiedades computacionales de las Lógicas Descriptivas (DL, description logics). Estos perfiles caracterizan la estructura lógica esencial que se necesita para manejar la semántica de los modelos conceptuales, habilitando el desarrollo de herramientas automáticas de interoperabilidad. No existe correspondencia exacta directa entre estos perfiles y fragmentos conocidos de lenguajes DL, y el núcleo común es pequeño (la lógica tratable $\mathcal{ALNI}$). Aunque es muy poca la posibilidad de derivar inconsistencias dentro de estos perfiles, es prometedor su uso en modelos conceptuales dado su complejidad en tiempo escalable.

Palabras claves: Lenguajes de modelado conceptual, modelos conceptuales, perfiles de lenguajes, uso de lenguajes de modelado

1 Introduction

Conceptual data models were proposed in the 1970s as a vehicle to describe what has to be stored or processed in a prospective information system, aiming at separating those 'what' aspects from the 'how' to achieve them. Many conceptual data modelling languages (CDMLs) have been proposed over the past 40 years by several research communities, notably originating from relational databases and object-oriented software. A number of variants emerged with motivations that include aiming for simplicity and leanness vs. expressiveness, modelling aspect of an application domain (e.g., spatial entities in geographic information systems), and ontology-driven modelling language proposals. Such proposals focus typically on graphical syntax of the languages with, at best, a partial formalisation for a proposed extension; afterward, other researchers aim to specify a formal semantics for a larger fragment of the language to facilitate computational support. Both leave loose ends either as "semantic variation point" [1] or as 'unsupported' by that particular logic-based reconstruction, which in turn spurs research into those contentious aspects. Meanwhile, it hampers interoperability even at the syntax level; e.g., a GenMyModel[1] serialisation of a UML Class Diagram is different from that of draw.io[2] and other modelling tools, and they are not mutually readable by the respective tools. This stands in stark contrast to related artefacts such as ontologies, which are typically serialised in the same RDF/XML format that can be used across editors and where each element of the model has the same semantics everywhere, as specified in a standard like [2].

The number of CDML modelling features has increased over time toward higher precision; e.g., Unified Modelling Language (UML) has identifiers since v2.4.1 [3], Object Role Modelling (ORM) version 2 has more ring constraints than the original ORM (compare [4] and [5]), and Extended Entity-Relationship (EER) also supports entity type subsumption and disjointness compared to Entity-Relationship (ER) [6, 7]. Opinions vary about this feature richness and its relation to model quality [8] and fidelity of capturing all the constraints specified by the customer. Asking modellers and domain experts which features they think they use, actually use, and need showed discrepancies between them [9]. It has been shown that advanced features are being used somewhere by someone, albeit infrequently [10, 11].

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 can be used for, among others, automated reasoning over a model to improve its quality (e.g., [12, 13]) and other runtime usage, such as conceptual and visual query formulation [14, 15, 16] and optimisation of query compilation [17].

Logic-based reconstructions proposed over the years (and discussed below) can be grouped into either the Description Logics (DL)-based approach or the as-expressive-as-needed approach. While the former proposes logics from the computational complexity point of view, the latter prioritises the needs and usages of modellers, such as in the case of full first-order predicate logic. None of them have 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 hampers the actual use of logic-based conceptual data models in information systems and therewith risk sliding into disuse. These problems with the multitude of incompatible ad hoc formalisations raise the questions of:

i. How should one design a logic methodologically?
ii. What would be a compatible set of logics for CDMLs that do take into account model feature usage and ontological commitments?

We aim to address these problems and answer these questions in this paper, specifically for the structural fragment of the most widely used CDMLs, because this features most prominently as a core interoperability issue in system integration and needs to be resolved before harmonising any 'dynamic' components such as methods.

First, we adapt ontology-driven language design principles for ontologies languages [18] to the CDML setting, which is informed by Frank's [19] domain-specific language (DSL) design methodology regarding process as well. 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 [11]. These logic 'profiles' formalizes a subset of features covering about 99% of those appearing 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 a model in UML Class Diagram notation that can be fully reconstructed into the standard view core profile (more precisely: $\mathcal{DC}_s$) is included in Fig. 1[3] 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.

The main contributions presented here are: i) a methodological language design process; ii) a new

[1]`https://www.genmymodel.com/`

[2]`https://app.diagrams.net/`

[3]Note: The diagram is introduced with the sole purpose of illustrating supported elements and constraints. Whether this is a good model is a separate matter and not the topic of this paper. For instance, one may want to model the roles persons play in a different way or make the Affiliation's Address a class rather than an attribute; see [20] for sample patterns to assist redesign.

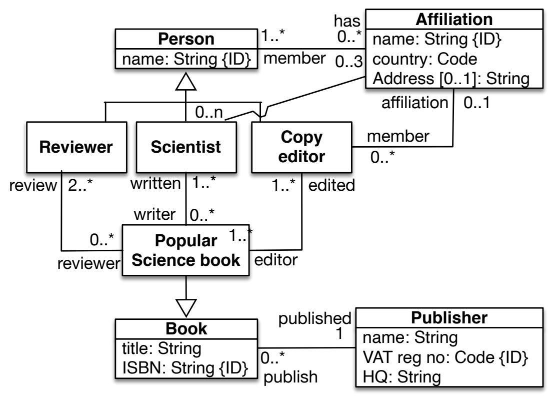

Figure 1: Sample UML Class Diagram containing all possible constraints of the Standard Core Profile, $\mathcal{DC}_s$, which emanated from the evidence-based profile specifications.

positionalist core profile; and iii) the profiles have been defined with a formal syntax and semantics. They are built upon a line of research reported in [21, 22, 23, 24, 11], which provided preliminary insights, such as the quantitative assessment of conceptual models on their features that inform the profile specification. 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 [3], EER [25, 6, 7] and its flavours such as Barker ER and IE notation, ORM [4, 5] and its flavours such as CogNIAM and FCO-IM, and others, such as MADS [26] and Telos [27]. Some of those are minor variants in notation, whereas others have a different number of features. Some 'families' of languages still run along the lines of the subfield from which they originally emerged: ER and EER originate from the relational database community, UML Class Diagrams from object-oriented programming, and ORM bears similarities with semantic networks, can be used for both relational and object-oriented 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 'craw's feet' icon vs ..n or ..* for 'many' multiplicity or 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., [28, 29] for this approach), and language design. The following sections highlight key aspects and are not to be assumed an exhaustive literature review.

2.1 Logic-based reconstructions of CDMLs

The two principal reasons for formalising conceptual models are: 1) to be more precise to improve a model's quality and 2) runtime usage of conceptual models. Most works are within the scope of the first motivation. Notably, various DLs have been used for giving the graphical elements a formal semantics and for automated reasoning over them [30, 12], although also other logics are being used, including OCL [13], CLIF [31], Alloy [32], and Z [33].

Zooming in on DLs, the $\mathcal{ALUNI}$ language has been used for a partial unification of CDMLs [28], whereas other DLs are used for particular modelling language formalisations, such as *DL-Lite* and $\mathcal{DLR}_{ifd}$ for ER [30] and UML [12], and OWL for ORM and UML [34]. These logic-based reconstructions are typically incomplete with respect to the CDML features they cover, such as omitting ER's identifiers ('keys') [28] or *n*-aries [30, 34], 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, [35, 4, 36, 37, 34], noting that full ORM and ORM2 (henceforth referred to inclusively as ORM2) 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 referred to inclusively as EER), multiple logic-based reconstructions exist from the modeller's viewpoints [25, 6, 7] and from the logician's vantage points with the $\mathcal{DLR}$ family [38, 39] and *DL-Lite* family [40] 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 [41, 10] and ontology-mediated query answering that involves, among others, user-oriented design and execution of queries [42, 14, 15, 16], querying databases during the stage of query compilation [17], and recent spatio-temporal stream queries that avail of ontology-based data access with conceptual models [43].

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

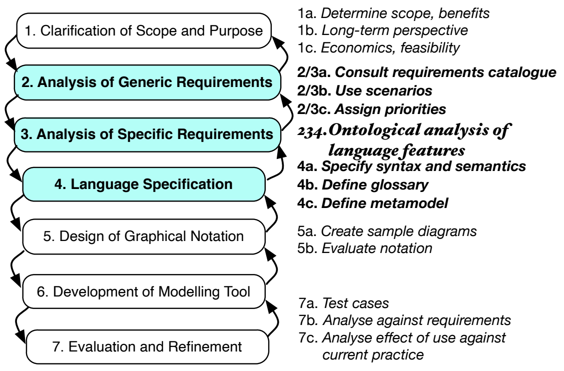
1. Clarification of Scope and Purpose
2. Analysis of Generic Requirements
3. Analysis of Specific Requirements
4. Language Specification
5. Design of Graphical Notation
6. Development of Modelling Tool
7. Evaluation and Refinement
1a. Determine scope, benefits
1b. Long-term perspective
1c. Economics, feasibility
2/3a. Consult requirements catalogue
2/3b. Use scenarios
2/3c. Assign priorities
234. Ontological analysis of language features
4a. Specify syntax and semantics
4b. Define glossary
4c. Define metamodel
5a. Create sample diagrams
5b. Evaluate notation
7a. Test cases
7b. Analyse against requirements
7c. Analyse effect of use against current practice

Figure 2: CDML language design process, adapted from [19, 18], where the focus of this paper is highlighted in bold and shaded (blue).

than for which features a modeller uses.

2.2 Language design

The design of a modelling or knowledge representation language is an engineering task, involving several steps and decision points among alternatives. A requirement might include n-ary relationships, no attributes, and a graphical syntax. Systematic approaches to language design have been developed, notably Frank's pipeline [19], as well as specifics for one step in the pipeline (e.g., requirements engineering [44]) or for one class of modelling languages (e.g., domain-specific languages [45]). In [18], we adapted Frank's waterfall model for domains-specific modelling languages (DSLs) [19] to the design of languages for ontologies, which we adapt here for CDLM design. Fig. 2 shows the adapted model with the steps we address in this paper highlighted. We will step through describing only these modified steps and with respect to applicability and related works on CDML design.

In order to identify requirements for Steps 2 and 3, there is no requirements catalogue for CDMLs (step 2/3a), but there is one for ontology languages [18] and there are several use cases (step 2/3b) (e.g., [46, 15]). Assigning priorities (step 2/3c) has been done for several languages, but mostly implicitly; e.g., prioritising scalability in the presence of large amounts of data, like with OWL 2 QL [47]. Assessment of sets of conflicting requirements are available, such as the pros and cons of several logics for formalising conceptual models [21]. It has yet to be decided how to assign priorities. One could survey industry [48], but it has been shown in at least one survey that modellers do not know the features well enough to be a reliable source [9]. Thus, existing works fall short on providing answers to steps 2 and 3.

Many papers describe a language specification (step 4), notably in a DL. Most of them do not have a metamodel, however. Regarding existing metamodels one may be able to reuse for the language specification: extant proposals in the conceptual modelling community span theoretical accounts, academic proof-of-concept implementations, and industry-level applications, such as in the Eclipse Modeling Framework[4]. The UML diagrams in the OWL and UML standards [2, 3] are essentially metamodels as well. To enable a comparison between CDMLs, a recent unified metamodel is required that covers all the language features, which reduces the choice to [46]. It covers all the static structural components in unifying UML Class Diagrams, ER and EER, and ORM and ORM2 at the metamodel layer and has both a glossary of elements and the constraints among them.

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. The case for, and benefits of, using insights from ontology to advance the modelling has been well-documented [49, 50], with ample detail about improvements on precision of representing the information; e.g., deploying the UFO foundational ontology to improve the UML 2.0 metamodel [51] and examining the nature of relationships [52, 53], and more general philosophical assessments about conceptual models, such as 3D versus 4D conceptual models [54]. The latter choice is primarily a metaphysical one, which is practically relevant in the data analysis stage. For instance, the Philips corporation evolved over its past 130 years of existence, acquiring companies into its conglomerate and spinning off others. If it is relevant for the domain of discourse to keep track of these changes, then a 4D perspective assists in the analysis.

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 usage

To the best of our knowledge, there are only two quantitative studies on CDML feature usage. In the first study, industry-grade ORM diagrams were examined [10] and in the second study, publicly available conceptual models in EER, UML, and ORM [11] were examined, whose results for ORM are similar to those reported in the former study. The diagrams of [11] were analysed using aforementioned unified metamodel [46], which facilitated cross-language comparisons and categorisation of entities in those languages into the harmonised terminology; a relevant selection of the terminology 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

[4]https://www.eclipse.org/modeling/emf/

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

Metamodel term	**UML Class Diagram**	**EER**	**ORM/FBM**
Relationship	Association	Relationship	Fact type
Role	Association/ member end	Component of a relationship	Role
Entity type	Classifier	–	Object type
Object type	Class	Entity type	Nonlexical object type/ Entity type
Attribute	Attribute	Attribute	–
Value type	–	–	Lexical object type/ Value type
Data type	Literal Specification	–	Data type

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, EER, and ORM2 models that were publicly available from online sources, published papers, and textbooks[5]. The average size of the diagram (vocabulary+subsumption) was found to be about 50 entities/diagram, totalling to 8036 entities, of which 5191 (i.e., 64%) were 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) [11]. The results are described and discussed in [11], where it is noted that while most features of each language family is typically used somewhere, their frequency varies; e.g., disjoint and covering constraints are used sparingly throughout the models, as are ring constraints in ORM and n-aries or association classes in UML. The obtained usage frequency for each entity, together with the design choices described in Section 3, sustain the logic profiles that will be introduced in Section 4.

3 Design choices for logic-based profiles

A formalisation based on the quantitative evidence is not as straight-forward as it may sound. Several design choices may result in a different logic, possibly be of a different computational complexity, use different reconstruction algorithms, and differ in tool support for the logic. This brings us to the "4" of the "234. Ontological analysis of language features" of Fig. 2. The "language specification" step concerns affordances and features of the logic, including the ability to represent the universe of discourse more or less precisely with more or less constraints and whether the representation language contributes to support, or even shape, the conceptualisation and one's data analysis for the conceptual data model or embeds certain philosophical assumptions and positions. Regarding the latter, we identified several decision points [18], which we adjusted to CDMLs, including, but not limited to:

1. Should the CDML be 'truly conceptual', ignoring 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, as being of a different kind);

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, with two notable observations: binaries vs. *n*-aries and plain relationship vs. also aggregation in the language. The latter was observed for UML vs. EER and ORM2 [11] and the former for UML [55]. The *n*-aries in UML class diagrams are hard to read due to the look-across notation [55] and it uses a different visual element than a binary association (diamond vs. line), and therefore used less frequently compared to EER and ORM2.

The interested reader is referred to [18] for a comprehensive explanations of the philosophical aspects.

[5]the models, their respective provenance, and raw data analysis are available from `http://www.meteck.org/swdsont.html`, which is not within the scope of this paper. That experiment design, results, and discussion are described in [11].

Here, we distill it into the main salient aspects that are interesting for comparing logics that are relatively popular for formalising CDMLs. This comparison is included in Table 2 and discussed in the remainder of this section. 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. Observe that the first section in the table suggests $\mathcal{DLR}_{ifd}$ and FOL are good candidates for logic-based reconstructions of conceptual data models; the second section has more positive evaluations for *DL-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.

A key difference across the different formalisations is whether to honour 'roles' in the language, which has been discussed at some length in [21, 52], because the main issue is that the CDMLs are all positionalist, yet most logics use predicates with the standard view. What does that mean? Take, e.g., a relationship teach that holds between Prof and Course. With the 'just predicates' decision, there is no teach relationship, but at least one predicate, teaches or taught by in which Professor and Course participate—in that specific order or in the reverse, respectively. Alternatively, the 'there are roles too'-option: Prof plays a role, e.g., [lecturer], in the relationship teach and Course plays the role [subject]; thus, role is an element in the language. This distinction between predicates-only and roles-too is called *standard view* vs. *positionalist*, respectively.

The notion of role as a component of an *n*-ary predicate is not an element of FOL (it just has predicates, functions, and constants), i.e., FOL adheres to the standard view by design, as do most DLs. In contrast, the $\mathcal{DLR}$ family [38] is positionalist, where the syntax does include a "DL role component" for the DL roles and it has a corresponding semantics. For instance, the teaching could then be specified as $\mathsf{teach} \sqsubseteq [\mathsf{lecturer}]\mathsf{Prof} \times [\mathsf{subject}]\mathsf{Course}$.

To address the impasse between CDMLs on the one hand and most logics on the other, there are several options. One could commit to a logic-based reconstruction of the models into a positionalist logic, such as those in the $\mathcal{DLR}$ family of DLs that have been used already for partial reconstructions of ER [38], UML class diagrams [12], and ORM [52], or include roles in the Z formalisation as proposed in [33], or devise a new logic for it. 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 [34], the original ORM formalisation was without roles in the language [4], and UML's association ends are sometimes ignored as well (e.g., [32, 56]), but not always [12].

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 ExpTime-complete under GCIs[57]. A notable exception is the OWL 2 EL profile that does not have inverse object properties [47].

4 Logic-based profiles for conceptual data modelling languages

We now proceed to define logics to characterise the model-theoretic semantics such that it is minimalist with respect to the most-used features for each of the three families of CDMLs. The language design takes into account the ontological considerations discussed in the previous section as well as the evidence from [11] 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 definitions and algorithms is shown in Fig. 3. These profiles constitute a theoretical backbone for an interoperability tool between conceptual models expressed in different graphical languages and with different philosophical assumptions. The main distinction is the positionalist or the standard view, resulting in profiles $\mathcal{DC}_p$ and $\mathcal{DC}_s$, which each formalise the most widely used language features. The standard view profile is then extended into three different profiles, one for each CDML, which serves as background knowledge to be exchanged between profiles. In order to interoperate from the positionalist and the standard view profiles some compromises must be taken, described mainly in Algorithm 1. Importing conceptual models into CDML may be carried out also with this theoretical structure, while exporting may be done by translating reasoner output into a suitable textual representation.

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 ORM and its precursor NIAM for the past 40 years [5], UML Class Diagram notation requires association ends as roles, and ER Models have relationship components [21]. On the other hand, First Order Logic

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; DOL aims to link logical theories represented in the same or different languages.

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
+ relevant reasoners	– no implementation	+ relevant reasoners	± few relevant reasoners
+ linking with ontologies doable	– no interoperability	+ linking with ontologies doable	– no interoperability with widely used infrastructures
+ compatibility with DOL	– no compatibility with DOL	+ compatibility with DOL	+ compatibility with DOL
+ modularity infrastructure	– modularity infrastructure	+ modularity infrastructure	– modularity infrastructure

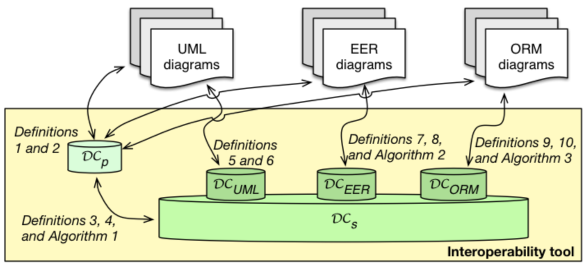

Figure 3: Sketch of the orchestration between the profiles and algorithms.

and most of its fragments, notably standard DLs [58], 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. As a side effect, the connection hold by the role name is lost, and two concepts that played the same role now play two completely independent roles. as the following example shows.

Example 1. *Consider de ER diagram shown in Fig. 4. In the* rent *relationship any person may rent any real estate property and then is assumed to occupy it somehow, whereas in the* mortgage *relationship any person living in a residential property may put a lien on it to obtain a loan from the bank. Both relationships involve the* occupant *role played by instances of the* Person *entity. This role name is relevant for querying, say, the real estate occupants in the database, so it is relevant for the model's intended meaning. Following the translation procedure described in [59] to the DL-Lite DL family, the role* occupant *in the* rent *relationship is formalised as*

$$\exists \mathsf{occupant} \sqsubseteq \mathsf{Person}$$
$$\exists \mathsf{occupant}^- \sqsubseteq \mathsf{RealEstateProperty}$$

The two formulas state the domain and the range of the role. Similarly, the role occupant *in the* mortgage *relationship is translated, including the functional constraint, as*

$$\exists \mathsf{occupant} \sqsubseteq \mathsf{Person}$$
$$\exists \mathsf{occupant}^- \sqsubseteq \mathsf{Residential}$$
$$\geq 2\, \mathsf{occupant} \sqsubseteq \bot$$

In the case that both formalisations are merged into the same conceptual model formalisation, then an unintended meaning may be obtained; e.g., that only houses may be rented, and that a person may rent only one property. The usual solution to such unintended consequences is to change the name one of the DL roles (i.e., the EER role bumped up to a binary relationship in the formalisation step), but then the connection between both roles is lost in the formalisation: the role is split, and therewith the intended meaning is weakened. This problem does not arise in the translation to the positionalist $\mathcal{DLR}_{\mathsf{ifd}}$ *following [12], since the roles are part of a different relationship and remain roles rather than be subject to element type recasting in the formalisation step.*

Therefore, we consider it relevant to design a positionalist core profile that preserves roles as first-class citizens among the DL vocabulary. In case reasoning

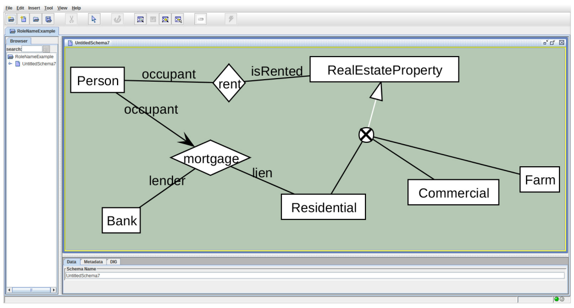

Figure 4: EER diagram showing multiple uses of the same role occupant.

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 further below.

4.1.1 Positionalist Core Profile

In this section we define the DL fragments that describes the positionalist core profile. We use the standard DL syntax and semantics, as given in [58, 12], and the terminology as listed in Table 1.

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 ranging over symbols A, B, ..., binary relationships P, datatypes T and attributes a in the model as the basic elements in the knowledge base.*
2. *for each binary relationship P formed by object types A and B, we add 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 [60], we add the assertion $A \sqsubseteq \exists a.T \sqcap \leq 1 a$.*
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 the overall complexity of the process (translation and then reasoning) 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 EER and ORM2 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, as 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 $\mathtt{from}, \mathtt{to} \in U$ fixed argument places for attributes such that $[\mathtt{from}]$ is the role played by the concept, and $[\mathtt{to}]$ the role played by the datatype. These names must be locally unique in each relationship/attribute.

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 $\{A \sqsubseteq \exists a.T \sqcap \leq 1 a \sqcap \forall a.T\}$ is not a $\mathcal{DC}_p$ knowledge base, it can't be obtained from the translation of any diagram.

Now we introduce the semantic characterisation.

Definition 2. *An $\mathcal{DC}_p$* interpretation *$\mathcal{I} = (\cdot^{\mathcal{I}}_{C}, \cdot^{\mathcal{I}}_{T}, \cdot^{\mathcal{I}})$ for a knowledge base in $\mathcal{DC}_p$ consists of a set of objects $\Delta^{\mathcal{I}}_{C}$, a set of datatype values $\Delta^{\mathcal{I}}_{T}$, 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 there exists T such that $C^{\mathcal{I}} \subseteq (\exists a.T \sqcap \leq 1 a)^{\mathcal{I}}$ (mandatory 1) and for all $v \in T^{\mathcal{I}}$ it holds that $\#\{c | c \in \Delta^{\mathcal{I}}_{C} \wedge (c, v) \in a^{\mathcal{I}}\} \leq 1$ (inverse functional).*

Example 2. *Let's consider the formalisation of the conceptual model in Fig. 4 in $\mathcal{DC}_p$, including some attributes and identification constraints not shown in* *the figure.*

$$
\begin{aligned}
\geq 1[\mathsf{occupant}]\mathsf{Rent} &\sqsubseteq \mathsf{Person}\\
\geq 1[\mathsf{isRented}]\mathsf{Rent} &\sqsubseteq \mathsf{RealEstateProperty}\\
\mathsf{Person} &\sqsubseteq \exists \mathsf{name}.\mathtt{String} \sqcap \leq 1\,\mathsf{name}\\
\mathsf{Person} &\sqsubseteq \exists \mathsf{idCard}.\mathtt{Integer}\\
&\quad \sqcap \leq 1\,\mathsf{idCard}\\
\mathsf{Person} &\sqsubseteq \exists \mathsf{phone}.\mathtt{Integer} \sqcap \leq 1\,\mathsf{phone}\\
\mathsf{Residential} &\sqsubseteq \mathsf{RealEstateProperty}\\
\mathsf{Commercial} &\sqsubseteq \mathsf{RealEstateProperty}\\
\mathsf{Farm} &\sqsubseteq \mathsf{RealEstateProperty}\\
&\mathtt{id}\ \mathsf{Person}\ \mathsf{idCard}
\end{aligned}
$$

The ternary relationship Mortgage *and the exclusiveness between the subconcepts of* RealEstateProperty *cannot be expressed in this profile. This allows the modeller to figure out which transformations are adequate to include in the formalisation (for example, an objectification of the relationship), and which semantics is missing. In this case, for simplicity, all the ternary relation is excluded from the formalisation.*

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

$$
\begin{gathered}
\top^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}_C\\
A^{\mathcal{I}} \subseteq \top^{\mathcal{I}}\\
\top^{\mathcal{I}}_2 = \top^{\mathcal{I}} \times \top^{\mathcal{I}}\\
P^{\mathcal{I}} \subseteq \top^{\mathcal{I}}_2\\
T^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}_T\\
a^{\mathcal{I}} \subseteq \top^{\mathcal{I}} \times \Delta^{\mathcal{I}}_T\\
(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}\\
(\leq k[i]R)^{\mathcal{I}} = \{c \in \Delta^{\mathcal{I}}_C | \#\{(d_1,d_2) \in R^{\mathcal{I}}.d_i = c\} \leq k\}\\
(\geq k[i]R)^{\mathcal{I}} = \{c \in \Delta^{\mathcal{I}}_C | \#\{d_1,d_2) \in R^{\mathcal{I}}.d_i = c\} \geq k\}\\
(\exists a.T)^{\mathcal{I}} = \{c \in \Delta^{\mathcal{I}}_C | \exists v \in \Delta^{\mathcal{I}}_T.(c,v) \in a^{\mathcal{I}} \wedge v \in T^{\mathcal{I}}\}\\
(\forall a.T)^{\mathcal{I}} = \{c \in \Delta^{\mathcal{I}}_C | \forall v \in \Delta^{\mathcal{I}}_T.(c,v) \in a^{\mathcal{I}} \rightarrow v \in T^{\mathcal{I}}\}\\
(\leq 1\, a)^{\mathcal{I}} = \{c \in \Delta^{\mathcal{I}}_C | \#\{(c,v) \in a^{\mathcal{I}}\} \leq 1\}\\
(i : C)^{\mathcal{I}} = \{(d_1,d_2) \in \top^{\mathcal{I}}_2 | d_i \in C^{\mathcal{I}}\}
\end{gathered}
$$

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 [39]) 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 [58, 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.

Despite all these limitations, no simpler positionalist DL has been introduced. 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.

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 $\exists Pe_1 \sqsubseteq \forall D_P$ and $\exists Pe_1^- \sqsubseteq R_P$
    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., $\exists Pe_i \sqsubseteq D_P$ and $\exists Pe_i^- \subseteq R_P$
        Declare inverses with $Pe_i \equiv Pe_2^-$
    **end if**
**end if**
```

4.1.2 Standard Core Profile

Considering formalisation choices such as the positionalism of the relationships [52, 61] and whether to use inverses or qualified cardinality constraints, a *standard core profile* has been specified [21]. In case the original context is a positionalist language, a translation into a standard (role-less) language is required. Algorithm 1 (adapted from [21]) does this work in linear time in the number of elements of the 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.*

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:

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

Definition 4. *A $\mathcal{DC}_s$* interpretation *for a knowledge base in $\mathcal{DC}_s$ is given by $\mathcal{I} = (\cdot^{\mathcal{I}}_C, \cdot^{\mathcal{I}}_T, \cdot^{\mathcal{I}})$, with $\Delta^{\mathcal{I}}_c$ the domain of interpretation for concepts, $\Delta^{\mathcal{I}}_T$ 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 $\mathcal{DC}_p$.*

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

$$\begin{gathered} \top^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}_C \\ A^{\mathcal{I}} \subseteq \top^{\mathcal{I}} \\ \top^{\mathcal{I}}_2 = \top^{\mathcal{I}} \times \top^{\mathcal{I}} \\ P^{\mathcal{I}} \subseteq \top^{\mathcal{I}}_2 \\ T^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}_T \\ a^{\mathcal{I}} \subseteq \top^{\mathcal{I}} \times \Delta^{\mathcal{I}}_T \\ (C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}} \end{gathered}$$

$$\begin{gathered} (R^-)^{\mathcal{I}} = \{(c_2, c_1) \in \Delta^{\mathcal{I}}_C \times \Delta^{\mathcal{I}}_C | (c_1, c_2) \in R^{\mathcal{I}}\} \\ (\forall R.A)^{\mathcal{I}} = \{c_1 \in \Delta^{\mathcal{I}}_C | \forall c_2.(c_1, c_2) \in R^{\mathcal{I}} \rightarrow c_2 \in A^{\mathcal{I}}\} \\ (\exists R.A)^{\mathcal{I}} = \{c_1 \in \Delta^{\mathcal{I}}_C | \exists c_2.(c_1, c_2) \in R^{\mathcal{I}} \wedge c_2 \in A^{\mathcal{I}}\} \\ (\leq k R)^{\mathcal{I}} = \{c_1 \in \Delta^{\mathcal{I}}_C | \#\{c_2 | (c_1, c_2) \in R^{\mathcal{I}}\} \leq k\} \\ (\geq k R)^{\mathcal{I}} = \{c_1 \in \Delta^{\mathcal{I}}_C | \#\{c_2 | (c_1, c_2) \in R^{\mathcal{I}}\} \geq k\} \\ (\forall a.T)^{\mathcal{I}} = \{c \in \Delta^{\mathcal{I}}_C | \forall v.(c, v) \in a^{\mathcal{I}} \rightarrow v \in T^{\mathcal{I}}\} \\ (\exists a.T)^{\mathcal{I}} = \{c \in \Delta^{\mathcal{I}}_C | \exists v.(c, v) \in a^{\mathcal{I}} \wedge v \in T^{\mathcal{I}}\} \\ (\leq 1\, a)^{\mathcal{I}} = \{c \in \Delta^{\mathcal{I}}_C | \#\{(c, v) \in a^{\mathcal{I}}\} | \leq 1\} \end{gathered}$$

Example 3. *We now show the formalisation of the same conceptual model as in Example 2, but then in this new standard view profile $\mathcal{DC}_s$. Recall that Algorithm 1 must be performed to get rid of roles, so the relationship* Rent *is renamed into* Rent1 *and* Rent2 *(we omit the subsumption axioms from this list, which are the same).*

$$\begin{aligned} \exists \mathsf{Rent1} &\sqsubseteq \forall \mathsf{Person} \\ \exists \mathsf{Rent1}^- &\sqsubseteq \forall \mathsf{RealEstateProperty} \\ \mathsf{Rent1} &\equiv \mathsf{Rent2}^- \\ \geq 1\, \mathsf{Rent1} &\sqsubseteq \mathsf{Person} \\ \geq 1\, \mathsf{Rent2} &\sqsubseteq \mathsf{RealEstateProperty} \end{aligned}$$

$$\begin{aligned} &\mathsf{Person} \sqsubseteq \leq 1\, \mathsf{name}.\mathtt{String} \\ &\mathsf{Person} \sqsubseteq \leq 1\, \mathsf{idCard}.\mathtt{Integer} \\ &\mathsf{Person} \sqsubseteq \exists \mathsf{phone}.\mathtt{Integer} \sqcap \leq 1\, \mathsf{phone} \\ &\qquad \mathtt{id}\, \mathsf{Person}\, \mathsf{idCard} \end{aligned}$$

It is possible to conclude from this example that the overall expressivity of the model, apart from the non positionalist view, is the same as in Example 2.

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 [62]) 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-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 the *DL-Lite*$_{core}$ fragment is also enough to include class disjointness in NLogSpace, and jumps to NP including disjoint covering [59].

4.1.3 UML Class diagram Profile

The profile for UML Class Diagrams strictly extends $\mathcal{DC}_s$. It was presented extensively in [21] and succinctly formally specified here.

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:

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

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 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 [12]); 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 $(\leq k\,a.T)^{\mathcal{I}} = \{c \in \Delta_C^{\mathcal{I}} | \#\{a \in T^{\mathcal{I}} | (c,a) \in a^{\mathcal{I}}\} \leq k\}$ and $(\geq k\,a.T)^{\mathcal{I}} = \{c \in \Delta_C^{\mathcal{I}} | \#\{a \in T^{\mathcal{I}} | (c,a) \in a^{\mathcal{I}}\} \geq k\}$.*

Example 4. *We now show the formalisation of the same conceptual model as before, but including the new constraints available in $\mathcal{DC}_{UML}$. We add the possibility for a* `Person` *to have two phone numbers.*

$$\begin{aligned}
\exists \mathtt{Rent1} &\sqsubseteq \forall \mathsf{Person}\\
\exists \mathsf{Rent1}^- &\sqsubseteq \forall \mathsf{RealEstateProperty}\\
\mathsf{Rent1} &\equiv \mathsf{Rent2}^-\\
\geq 1\,\mathsf{Rent1} &\sqsubseteq \mathsf{Person}\\
\geq 1\,\mathsf{Rent2} &\sqsubseteq \mathsf{RealEstateProperty}\\
\mathsf{Person} &\sqsubseteq \leq 1\,\mathsf{name}.\mathtt{String}\\
\mathsf{Person} &\sqsubseteq \leq 1\,\mathsf{idCard}.\mathtt{Integer}\\
\mathsf{Person} &\sqsubseteq \leq 2\,\mathsf{phone}.\mathtt{Integer}\\
\mathtt{id}\,\mathsf{Person}\,\mathsf{idCard}
\end{aligned}$$

The only new constraint here is a cardinality constraint on the attribute phone*.*

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-Lite*$_{core}^{\{\mathcal{HN}\}}$, which is NLogSpace for subsumption and AC^0 for data complexity [59]. 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 ExpTime-hard concept satisfiability [63].

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 $\mathcal{DC}_s$ 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, we add the assertion $\mathsf{fd}\,R\,i_1, i_2 \to i_3$, such that $1 \leq i, i_1, i_2 \leq 3$ and are all different.*
4. *associative object types are formalised by the reification of the association as a new DL concept with two binary relationships.*
5. *multi-attribute identification is formalised as a new composite attribute with single identification.*

This profile was presented extensively in [21] 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:

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

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 $\mathsf{fd}\,R\,i_1, i_2 \to i_3$ 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 $(\leq k\,a.T)^{\mathcal{I}} = \{c \in \Delta_C^{\mathcal{I}} | \#\{a \in T^{\mathcal{I}} | (c,a) \in a^{\mathcal{I}}\} \leq k\}$ and $(\geq k\,a.T)^{\mathcal{I}} = \{c \in \Delta_C^{\mathcal{I}} | \#\{a \in T^{\mathcal{I}} | (c,a) \in a^{\mathcal{I}}\} \geq k\}$.*

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 EER 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 EER 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 $\mathcal{DLR}_{ifd}$ [39] and its close relative $\mathcal{DLR}^+$ [64] or in $\mathcal{CFD}$ [65]. 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 $\mathcal{DLR}_{ifd}$. However, $\mathcal{DLR}_{ifd}$ also offers DL role components that are not strictly needed for EER, 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 EER Profile in the description logic $\textit{DL-Lite}^{\mathcal{N}}_{core}$ [59] 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 [30] for ER_{ref}, but it excludes composite attributes and weak object types.

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

```
$D_P$: 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 $\exists Re_i \sqsubseteq P'$ and $\exists Re_i^- \sqsubseteq R_P$, where $R_P$ is the player (EER entity type) in $n$
    Add $P' \sqsubseteq \exists Re_i.\top$ and $P' \sqsubseteq \leq 1\, Re_i.\top$
end for
Add external identifier $\top \sqsubseteq \leq 1\, (\sqcup_i Re_i)^-.P'$
```

Example 5. *In the formalization of Fig. 4 in this profile, we can now include the ternary relationship* Mortgage *that was absent from previous examples. By applying algorithm 1 we get three new roles which are labeled as* Lender, Occupant, *and* Lien. *Next we show only the new axioms for the second role, which it is more interesting since it has the uniqueness constraint. The other roles of the ternary relations are handled similarly, and the rest of the axioms as in previous examples.*

$$\mathsf{Mortgage} \sqsubseteq \top_3$$
$$\exists \mathsf{Occupant} \sqsubseteq \forall \mathsf{Person}$$
$$\exists \mathsf{Occupant}^- \sqsubseteq \forall \mathsf{Mortgage}$$
$$\geq 1\, \mathsf{Occupant} \sqsubseteq \mathsf{Person}$$
$$\geq 1\, \mathsf{Occupant}^- \sqsubseteq \mathsf{Mortgage}$$
$$\leq 1\, \mathsf{Occupant} \sqsubseteq \mathsf{Person}$$

$\top_3$ *is the universe of all ternary relationships in the discourse domain. To this formalisation we can further apply algorithm 2 if needed. Observe, as mentioned in example 1, the lost connection between the same named roles.*

4.1.5 ORM and ORM2 Profile

Unlike the case of the ER and EER profile, there is no suitable mechanism to avoid 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 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 *ORM2 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 ORM2 Diagrams was presented in [21], and a more detailed version including a text-based mapping as a restricted "ORM2$_{cfd}$" was developed in [23] using $\mathcal{CFDI}^{\forall -}_{nc}$ 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. *add pairwise disjoint axioms for each pair of relationships with different arity.*

4. *each subsumption between roles R, S is represented by the formula $R \sqsubseteq S$.*

5. *each subsumption between relationships R, S is represented by the subsumption between the reified concepts, $R' \sqsubseteq S'$, and the subsumption of each of the n components of the relationships, $R_{ei} \sqsubseteq S_{ei}, 1 \leq i \leq n$.*

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

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

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

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

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

11. *each external uniqueness constraint between roles $R_i, 1 < i \leq n$ not belonging to the same relationship is represented by* $\mathsf{id} A\, 1 R_1, \ldots, 1 R_n$*, 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.*

12. *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 compared to simple uniqueness).*

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

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

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. We show an example of this profile in next subsection.

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 ORM2 Profile, this is not clear either. The ExpTime-complete $\mathcal{DLR}_{ifd}$ 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 $\mathcal{CFDI}_{nc}^{\forall -}$ [66] 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. 5.

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:
$\top \sqsubseteq \forall \mathsf{hasMember.Affiliation} \sqcap$
$\forall \mathsf{hasMember}^-.\mathsf{Person}$
$\top \sqsubseteq \forall \mathsf{has.Person} \sqcap \forall \mathsf{has}^-.\mathsf{Affiliation}$
and 3) declare inverses, $\mathsf{hasMember} \equiv \mathsf{has}^-$. 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
$\mathsf{Person} \sqsubseteq \exists \mathsf{Name.String} \sqcap \leq 1\ \mathsf{Name}$
and likewise for the other attributes. Step 4 takes care of the subsumptions; among others
$\mathsf{Popular\_science\_book} \sqsubseteq \mathsf{Book}$
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: $\mathsf{Affiliation} \sqsubseteq \geq 1\ \mathsf{has\_member}$ (mandatory participation) whereas for, say, the scientist, it will be $\mathsf{Scientist} \sqsubseteq \leq 3\, \mathsf{has}$. Finally, any identifiers are processed, such as ISBN for Book, generating the addition of the $\mathsf{id\, Book\, ISBN}$ to the $\mathcal{DC}_s$ knowledge

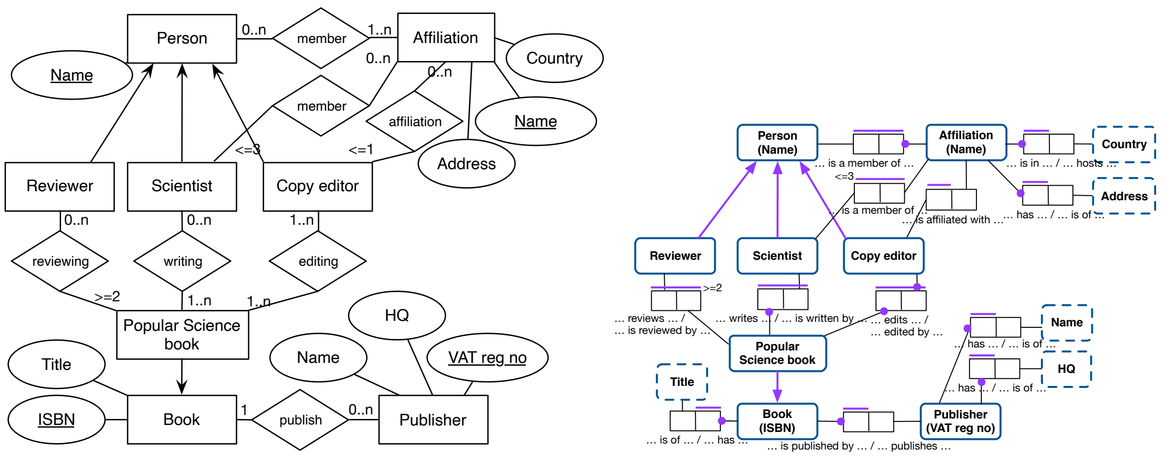

Figure 5: The sample diagram of Fig. 1 rendered in EER and ORM2 notation; the common $\mathcal{DC}_s$ logic-based construction is discussed in the text.

base.

The process for the EER diagram is the same except that the name of the relationship can be used directly instead of bumping up the role names to relationship names. The reconstruction into ORM has two permutations compared to 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 [60], 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 done automatically, except for the generation of labels. 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].

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 and computational complexity, and look ahead at applicability.

[6]It would have rules that render, e.g., a has_member into ... has member ... and a has_member$^-$ into ... member of ...

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 [19] 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 compared to Frank's domain-specific languages. An alternative option we considered beforehand was [45]'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. An interesting avenue for further research is transforming the proposed waterfall into actionable guidelines for CDML design.

Zooming in on the 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 design systematically. 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 ORM2, noting that it simply can be set to `xsd:anyType` and thus have no influence, which is the case for EER. Further, if the intended semantics of the aggregation association were to have been 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, 67] is not in widespread use and therewith did not meet the evidence-based threshold for inclusion.

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 that further work is necessary on the associated 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 as compared to EER's and ORM2's plain relationships *is* being used disproportionally more often than part-whole relations in EER and ORM2. 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 (instead of the logic and technology, as in, e.g., [68]), whilst still keeping it tractable. The current conceptual modelling tools that have a logic back-end are still sparse [32, 69, 70, 71], 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 [60, 72]. 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 [73] 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 Conclusion

A systematic logic design process was proposed that generalises and extends 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 for the three conceptual data modelling lan-

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	$DL\text{-}Lite_{core}^{\mathcal{HN}}$	NLogSpace
$\mathcal{DC}_{EER}$	ternary relationships, attribute cardinality, external keys	$DL\text{-}Lite_{core}^{\mathcal{N}}$	NLogSpace
		$\mathcal{CFD}$	P
$\mathcal{DC}_{ORM}$	entity type disjunction, relationships complement, relationship subsumption, complex identifiers (‘multi attribute keys’)	$\mathcal{DLR}_{ifd}$	ExpTime
		$\mathcal{CFDI}_{nc}^{\forall -}$	P

guage families 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 negation, hardly any inconsistencies can be derived within the profiles. Since $\neg A$ is not in the language, inconsistencies can only occur as side effects of incompatible cardinality constraints. This has as flip side that it promises scalable runtime usage 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.

Competing interests

The authors have declared that no competing interests exist.

Funding

This work was partially supported by the National Research Foundation of South Africa and the Argentinian Ministry of Science and Technology.

Authors’ contribution

PRF and CMK conceived the idea and developed the theory; CMK was the main contributor to sections 2 and 3; PRF and CMK both devised an example; PRF and CMK wrote the manuscript. All authors read and approved the final manuscript.

References

[1] Object Management Group, “OMG Unified Modeling Language (OMG UML).” online, December 2017. `http://www.omg.org/spec/UML/2.5.1`.

[2] B. Motik, P. F. Patel-Schneider, and B. Parsia, “OWL 2 web ontology language structural specification and functional-style syntax,” w3c recommendation, W3C, 27 Oct. 2009. http://www.w3.org/TR/owl2-syntax/.

[3] Object Management Group, “Superstructure specification,” Standard 2.4.1, Object Management Group, 2012. http://www.omg.org/spec/UML/2.4.1/.

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

[5] T. Halpin and T. Morgan, *Information modeling and relational databases*. Morgan Kaufmann, 2nd ed., 2008.

[6] I.-Y. Song and P. P. Chen, “Entity relationship model,” in *Encyclopedia of Database Systems* (L. Liu and M. T. Özsu, eds.), vol. 1, pp. 1003–1009, Springer, 2009.

[7] B. Thalheim, “Extended entity relationship model,” in *Encyclopedia of Database Systems* (L. Liu and M. T. Özsu, eds.), vol. 1, pp. 1083–1091, Springer, 2009.

[8] D. L. Moody, “Theoretical and practical issues in evaluating the quality of conceptual models: current state and future directions,” *Data & Knowledge Engineering*, vol. 55, pp. 243–276, 2005.

[9] 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, and M. Wessel, “Analysis of test results on usage scenarios,” deliverable TONES-D27 v1.0, TONES Project, Oct. 10 2008.

[10] Y. Smaragdakis, C. Csallner, and R. Subramanian, “Scalable satisfiability checking and test data generation from modeling diagrams,” *Automation in Software Engineering*, vol. 16, pp. 73–99, 2009.

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

[12] D. Berardi, D. Calvanese, and G. De Giacomo, “Reasoning on UML class diagrams,” *Artificial Intelligence*, vol. 168, no. 1-2, pp. 70–118, 2005.

[13] A. Queralt, A. Artale, D. Calvanese, and E. Teniente, “OCL-Lite: Finite reasoning on UML/OCL conceptual schemas,” *Data & Knowledge Engineering*, vol. 73, pp. 1–22, 2012.

[14] D. Calvanese, C. M. Keet, W. Nutt, M. Rodríguez-Muro, and G. Stefanoni, “Web-based graphical querying of databases through an ontology: the WONDER system,” in *Proceedings of ACM Symposium on Applied Computing (ACM SAC’10)* (S. Y. Shin, S. Ossowski, M. Schumacher, M. J. Palakal, and C.-C. Hung,

eds.), pp. 1389–1396, ACM, 2010. March 22-26 2010, Sierre, Switzerland.

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

[16] A. Soylu, E. Kharlamov, D. Zheleznyakov, E. J. Ruiz, M. Giese, M. G. Skjaeveland, D. Hovland, R. Schlatte, S. Brandt, H. Lie, and I. Horrocks, "OptiqueVQS: a visual query system over ontologies for industry," *Semantic Web Journal*, vol. 9, no. 5, pp. 627–660, 2018.

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

[18] P. R. Fillottrani and C. M. Keet, "An analysis of commitments in ontology language design," in *Proceedings of the 11th International Conference on Formal Ontology in Information Systems (FOIS'20)* (B. Brodaric and F. Neuhaus, eds.), vol. 330 of *Frontiers in Artificial Intelligence and Applications*, pp. 46–60, 2020.

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

[20] P. R. Fillottrani and C. M. Keet, "Patterns for heterogeneous tbox mappings to bridge different modelling decisions," in *Proceeding of the 14th Extended Semantic Web Conference (ESWC'17)* (E. Blomqvist *et al.*, eds.), vol. 10249 of *LNCS*, pp. 371–386, Springer, 2017. 30 May - 1 June 2017, Portoroz, Slovenia.

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

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

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

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

[25] P. P. Chen, "The entity-relationship model—toward a unified view of data," *ACM Transactions on Database Systems*, vol. 1, no. 1, pp. 9–36, 1976.

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

[27] J. Mylopoulos, A. Borgida, M. Jarke, and M. Koubarakis, "Telos: Representing knowledge about information systems," *ACM Transactions on Information Systems*, vol. 8, no. 4, pp. 325–362, 1990.

[28] D. Calvanese, M. Lenzerini, and D. Nardi, "Unifying class-based representation formalisms," *Journal of Artificial Intelligence Research*, vol. 11, pp. 199–240, 1999.

[29] C. M. Keet, "Ontology-driven formal conceptual data modeling for biological data analysis," in *Biological Knowledge Discovery Handbook: Preprocessing, Mining and Postprocessing of Biological Data* (M. Elloumi and A. Y. Zomaya, eds.), ch. 6, pp. 129–154, Wiley, 2013.

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

[31] W.-L. Pan and 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)*, vol. 2, pp. 104–109, IEEE Computer Society, 2010.

[32] B. F. B. Braga, J. P. A. Almeida, G. Guizzardi, and A. B. Benevides, "Transforming OntoUML into Alloy: towards conceptual model validation using a lightweight formal methods," *Innovations in Systems and Software Engineering*, vol. 6, no. 1-2, pp. 55–63, 2010.

[33] A. Jahangard Rafsanjani and S.-H. Mirian-Hosseinabadi, "A Z Approach to Formalization and Validation of ORM Models," in *Digital Enterprise and Information Systems* (E. Ariwa and E. El-Qawasmeh, eds.), vol. 194 of *CCIS*, pp. 513–526, Springer, 2011.

[34] H. M. Wagih, D. S. E. Zanfaly, and M. M. Kouta, "Mapping Object Role Modeling 2 schemes into $\mathcal{SROIQ}(d)$ description logic," *International Journal of Computer Theory and Engineering*, vol. 5, no. 2, pp. 232–237, 2013.

[35] E. Franconi, A. Mosca, and 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, March 2012.

[36] A. H. M. t. Hofstede and H. A. Proper, "How to formalize it? formalization principles for information systems development methods," *Information and Software Technology*, vol. 40, no. 10, pp. 519–540, 1998.

[37] C. M. Keet, "Mapping the Object-Role Modeling language ORM2 into Description Logic language $\mathcal{DLR}_{ifd}$," Tech. Rep. 0702089v2, KRDB Research Centre, Free University of Bozen-Bolzano, Italy, April 2009. arXiv:cs.LO/0702089v2.

[38] D. Calvanese, G. De Giacomo, and 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, 1998.

[39] D. Calvanese, G. De Giacomo, and M. Lenzerini, "Identification constraints and functional dependencies in description logics," in *Proc. of the 17th Int. Joint Conf. on Artificial Intelligence (IJCAI 2001)* (B. Nebel, ed.), pp. 155–160, Morgan Kaufmann, 2001. Seattle, Washington, USA, August 4-10, 2001.

[40] D. Calvanese, G. D. Giacomo, D. Lembo, M. Lenzerini, and R. Rosati, "Tractable reasoning and efficient query answering in description logics: The DL-Lite family," *Journal of Automated Reasoning*, vol. 39, no. 3, pp. 385–429, 2007.

[41] M. Nizol, L. K. Dillon, and 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)*, pp. 13–18, ACM Conference Proceedings, 2014. Hyderabad, India, June 02-03, 2014.

[42] A. C. Bloesch and T. A. Halpin, "Conceptual Queries using ConQuer-II," in *Proceedings of ER'97: 16th International Conference on Conceptual Modeling*, vol. 1331 of *LNCS*, pp. 113–126, Springer, 1997.

[43] E. G. Kalayci, G. Xiao, V. Ryzhikov, T. E. Kalayci, and D. Calvanese, "Ontop-temporal: A tool for ontology-based query answering over temporal data," in *Proceedings of the 27th ACM International Conference on Information and Knowledge Management, CIKM 2018, Torino, Italy, October 22-26, 2018*, pp. 1927–1930, 2018.

[44] S. de Kinderen and Q. Ma, "Requirements engineering for the design of conceptual modeling languages," *Applied Ontology*, vol. 10, no. 1, pp. 7–24, 2015.

[45] G. Karsai, H. Krahn, C. Pinkernell, B. Rumpe, M. Schindler, and S. Völkel, "Design guidelines for Domain Specific Languages," in *Proceedings of the 9th OOPSLA Workshop on Domain-Specific Modeling (DSM'09)*, 2009. Orlando, Florida, USA, October 2009.

[46] C. M. Keet and P. R. Fillottrani, "An ontology-driven unifying metamodel of UML Class Diagrams, EER, and ORM2," *Data & Knowledge Engineering*, vol. 98, pp. 30–53, 2015.

[47] B. Motik, B. C. Grau, I. Horrocks, Z. Wu, A. Fokoue, and C. Lutz, "OWL 2 Web Ontology Language Profiles," W3C recommendation, W3C, 27 Oct. 2009. http://www.w3.org/TR/owl2-profiles/.

[48] I. Malavolta, P. Lago, H. Muccini, P. Pelliccione, and A. Tang, "What industry needs from architectural languages: A survey," *IEEE Transactions on Software Engineering*, vol. 39, no. 6, pp. 869–891, 2013.

[49] N. Guarino, "The ontological level: Revisiting 30 years of knowledge representation," in *Mylopoulos Festschrift* (A. Borgida *et al.*, eds.), vol. 5600 of *LNCS*, pp. 52–67, Springer, 2009.

[50] N. Guarino and G. Guizzardi, "In the defense of ontological foundations for conceptual modeling," *Scandinavian Journal of Information Systems*, vol. 18, no. 1, pp. (debate forum, 9p), 2006.

[51] G. Guizzardi and G. Wagner, "Using the unified foundational ontology (UFO) as a foundation for general conceptual modeling languages," in *Theory and Applications of Ontology: Computer Applications*, pp. 175–196, Springer, 2010.

[52] C. M. Keet, "Positionalism of relations and its consequences for fact-oriented modelling," in *OTM Workshops, International Workshop on Fact-Oriented Modeling (ORM'09)* (R. Meersman, P. Herrero, and D. T., eds.), vol. 5872 of *LNCS*, pp. 735–744, Springer, 2009. Vilamoura, Portugal, November 4-6, 2009.

[53] G. Guizzardi and G. Wagner, "What's in a relationship: An ontological analysis," in *Proceedings of the 27th International Conference on Conceptual Modeling (ER'08)* (Q. Li, S. Spaccapietra, E. S. K. Yu, and A. Olivé, eds.), vol. 5231 of *LNCS*, pp. 83–97, Springer, 2008. Barcelona, Spain, October 20-24, 2008.

[54] M. West, C. Partridge, and M. Lycett, "Enterprise data modelling: Developing an ontology-based framework for the shell downstream business," in *Proceedings of Formal Ontologies Meet industry (FOMI'10)* (R. Cuel and R. Ferrario, eds.), pp. 71–84, 2010. 14-15 December 2010, Trento, Italy.

[55] P. Shoval and S. Shiran, "Entity-relationship and object-oriented data modeling—an experimental comparison of design quality," *Data and Knowledge Engineering*, vol. 21, pp. 297–315, 1997.

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

[57] S. Tobies, *Complexity Results and Practical Algorithms for Logics in Knowledge Representation*. PhD thesis, RWTH Aachen, 2001.

[58] F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider, eds., *The Description Logics Handbook – Theory and Applications*. Cambridge University Press, 2 ed., 2008.

[59] A. Artale, D. Calvanese, R. Kontchakov, and M. Zakharyaschev, "The DL-Lite family and relations," *Journal of Artificial Intelligence Research*, vol. 36, pp. 1–69, 2009.

[60] P. R. Fillottrani and C. M. Keet, "Conceptual model interoperability: a metamodel-driven approach," in *Proceedings of the 8th International Web Rule Symposium (RuleML'14)* (A. Bikakis *et al.*, eds.), vol. 8620 of *LNCS*, pp. 52–66, Springer, 2014. August 18-20, 2014, Prague, Czech Republic.

[61] J. Leo, "Modeling relations," *Journal of Philosophical Logic*, vol. 37, pp. 353–385, 2008.

[62] F. Donini, M. Lenzerini, D. Nardi, and W. Nutt, "Tractable concept languages.," in *Proc.of IJCAI'91*, vol. 91, pp. 458–463, 1991.

[63] F. Baader, S. Brandt, and C. Lutz, "Pushing the EL envelope," in *IJCAI-05, Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence, Edinburgh, Scotland, UK, July 30 - August 5, 2005* (L. P. Kaelbling and A. Saffiotti, eds.), pp. 364–369, Professional Book Center, 2005.

[64] A. Artale, E. Franconi, R. Peñaloza, and F. Sportelli, "A decidable very expressive description logic for databases," in *The Semantic Web – ISWC 2017: 16th International Semantic Web Conference* (C. d'Amato, M. Fernandez, V. Tamma, F. Lecue, P. Cudré-Mauroux, J. Sequeda, C. Lange, and J. Heflin, eds.), vol. 10587 of *LNCS*, (Cham), pp. 37–52, Springer, 2017. 21–25 October 2017, Vienna, Austria.

[65] D. Toman and 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*, pp. 948–954, AAAI Press, 2009.

[66] D. Toman and 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, 2014.

[67] G. Guizzardi, *Ontological Foundations for Structural Conceptual Models*. Phd thesis, University of Twente, The Netherlands. Telematica Instituut Fundamental Research Series No. 15, 2005.

[68] D. Calvanese, B. Cogrel, S. Komla-Ebri, R. Kontchakov, D. Lanti, M. Rezk, M. Rodriguez-Muro, and G. Xiao, "Ontop: Answering SPARQL queries over relational databases," *Semantic Web Journal*, vol. 8, no. 3, pp. 471–487, 2017.

[69] C. Farré, A. Queralt, G. Rull, E. Teniente, and T. Urpí, "Automated reasoning on UML conceptual schemas with derived information and queries," *Information and Software Technology*, vol. 55, no. 9, pp. 1529 – 1550, 2013.

[70] P. R. Fillottrani, E. Franconi, and S. Tessaris, "The ICOM 3.0 intelligent conceptual modelling tool and methodology," *Semantic Web Journal*, vol. 3, no. 3, pp. 293–306, 2012.

[71] G. A. Braun, C. Giménez, P. R. Fillottrani, and 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, 2017.

[72] Z. C. Khan, C. M. Keet, P. R. Fillottrani, and K. Cenci, "Experimentally motivated transformations for intermodel links between conceptual models," in *20th Conference on Advances in Databases and Information Systems (ADBIS'16)* (J. Pokorný *et al.*, eds.), vol. 9809 of *LNCS*, pp. 104–118, Springer, 2016. August 28-31, Prague, Czech Republic.

[73] V. Dimitrieski, M. Celikovic, S. Aleksic, S. Risti, A. Alargt, and 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*, vol. 44, no. Part C, pp. 299 – 318, 2015.

Citation: P.R.Fillottrani and C.M.Keet. *Evidence-based lean conceptual data modelling languages.* Journal of Computer Science & Technology, vol. 21, no. 2 , pp. 93-111, 2021.
DOI: 10.24215/16666038.21.e10
Received: January 18, 2021 **Accepted:** May 3, 2021.