Towards Semantics for Abstractions in Ontology-Driven Conceptual Modeling

Elena Romanenko[1(✉)], Oliver Kutz[1], Diego Calvanese[1,2], and Giancarlo Guizzardi[3]

[1] Free University of Bozen-Bolzano, 39100 Bolzano, Italy
{eromanenko,oliver.kutz,diego.calvanese}@unibz.it
[2] Umeå University, 90187 Umeå, Sweden
[3] University of Twente, 7500 Enschede, The Netherlands
g.guizzardi@utwente.nl

Abstract. Ontology-driven conceptual models are precise and semantically transparent domain descriptions that enable the development of information systems. As symbolic artefacts, such models are usually considered to be self-explanatory. However, the complexity of a system significantly correlates with the complexity of the conceptual model that describes it. Abstractions of both conceptual models and ontology-driven conceptual models are thus considered to be a promising way to improve the *understandability* and *comprehensibility* of those models. Although algorithms for providing abstractions of such models already exist, they still lack precisely formulated formal semantics. This paper aims to provide an approach towards the formalization of the abstraction process. We specify in first-order modal logic one of the graph-rewriting rules for ontology-driven conceptual model abstractions, in order to verify the correctness of the corresponding abstraction step. We also assess the entire network of abstractions of ontology-driven conceptual models and discuss existing drawbacks.

Keywords: Semantics for Abstractions · Abstractions of Ontology-Driven Conceptual Models · Networks of Model Abstractions

1 Introduction

Conceptual models (CMs) are high-level abstractions that are used to capture information about a domain of interest or a system that needs to be described. A special class of conceptual models that utilize foundational ontologies to ground modeling elements, modeling languages, and tools, is formed by *ontology-driven conceptual models*, ODCMs [22].

Although CMs are aimed at human comprehension [2] and are usually considered as self-explanatory artefacts, one of the most challenging problems is "to understand, comprehend, and work with very large conceptual schemas" [23]. The main reason for this is the complexity of the described domain or information system, which reveals itself in the complexity of the corresponding model.

T. P. Sales et al. (Eds.): ER 2023 Workshops, LNCS 14319, pp. 199–209, 2023.
https://doi.org/10.1007/978-3-031-47112-4_19

The problem of making (OD)CMs more comprehensible has been addressed in the literature for quite some time, and different complexity management techniques have been introduced, including clustering methods, relevance methods, and summarization methods [23]. This paper analyses algorithms from the last group, which are aimed to produce a reduced version of the original model (also called summarization or abstraction).

Most of the methods for complexity management of CMs are based on classic modeling notations, such as UML and ER, and rely on syntactic properties of the model [23], such as distance between model elements. However, in the case of ODCMs, there is the possibility to make good use of the built-in ontological semantics. The first version of an abstraction algorithm leveraging foundational ontological semantics was introduced in [12], followed by an enhanced one [18], which is based on 11 graph-rewriting rules.

Although there are some reports on testing the existing algorithm over the FAIR Catalog of OntoUML/UFO models [1] and with users (see [19]), the abstraction mechanism itself still lacks formal semantics. Hence, it is not always clear whether the existing abstraction patterns actually lead to more general models. The main goal of this paper is to propose an *approach for the formalization of this abstraction process*, which is illustrated for one of the existing graph-rewriting rules. This, in turn, enables further investigations of the properties of the process itself and of the resulting abstracted models.

The remainder of the paper is organized as follows. Section 2 reviews existing approaches to formal semantics for abstractions and provides some context for the analysed algorithm. Section 3 translates one of the graph-rewriting rules for producing ODCM's abstractions into first-order modal logic and investigates the relationship between the original and the abstracted model. Section 4 provides a preliminary analysis of networks of ODCM's abstractions that are generated by applying the specified rules. Section 5 elaborates on final considerations and future work.

2 Background and Related Work

2.1 Semantics for Abstractions

Before formulating the desired properties of the abstraction process, we first need to define what we mean by model abstraction. As outlined by Saitta & Zucker [20, p. 49], most existing theories identify abstracting with *a mapping from a ground (original) to an abstracted (intended) space*, and differ in the nature of spaces and the corresponding type of mapping.

The first works in this research field mostly dealt with abstraction at the level of syntax (see [15,17,21]). Giunchiglia & Walsh [10] extended those approaches and proposed a more general theory of the abstraction process. According to these authors, abstracting in problem-solving and theorem proving may be represented as a mapping f between the formal systems Σ_1 (representing the *ground space*) and Σ_2 (representing the *abstract space*). They suggested distinguishing between *Theorem-Decreasing* (TD), *Theorem-Constant* (TC), and *Theorem-*

Increasing (TI) abstractions depending on the changes in theorems of the formal system.

In TC abstractions, the abstract space has exactly the same theorems as the ground space reformulated in another language, so that all well-formed formulas that are theorems of Σ_1 map onto well-formed formulas that are theorems of Σ_2. In a TI abstraction, the abstract space has more theorems than the ground one, while the opposite happens for a TD abstraction. The authors argued that *"certain subclasses of TI abstractions are the appropriate formalization for abstraction"* [10]. Although for the authors, abstracting is a one-step process of mapping from one language to another, they noted that "the process of abstraction can be iterated to generate hierarchies of abstract spaces" [10].

Later Ghidini & Giunchiglia [8,9] proposed a model-theoretic formalization of the abstraction process based on the *Local Models Semantics* and the notion of *compatibility relation*. They claimed that "tuning of the compatibility relation allows for the definition of the many different kinds of abstraction" [8].

Therefore, we can assume that the rules suggested for developing the abstractions for ODCMs [12,18] should generate TI abstractions as defined by Giunchiglia & Walsh [10]. We come back to this discussion in Sect. 3.

2.2 The Unified Foundational Ontology and OntoUML

The *Unified Foundational Ontology* (UFO) is an axiomatic domain-independent formal theory that builds on contributions from analytic metaphysics, cognitive sciences, linguistics, and philosophical logic [14]. UFO addresses fundamental ontological notions via a set of micro-theories that represent types and taxonomic structures, part-whole relations, relations, and events among others. Also, it has been widely used as a foundational ontology in conceptual modeling [22].

The first distinction that UFO makes is highlighting the existence of both *endurants* and *perdurants*. Endurants are object-like individuals that persist in time and are able to qualitatively change while maintaining their identities [13]. Examples include ordinary objects, e.g., 'Car', and existentially dependent entities, e.g., 'Weight'. In contrast, perdurants are entities that unfold in time.

Endurant types in UFO are of different sorts, distinguished by formal meta-properties of *rigidity* and *sortality*. Sortality is defined via the notion of *principle of identity*. A type is *sortal* if all of its instances follow the same identity principle. A *non-sortal* type aggregates properties that are common to different sortals. An example is 'Artwork', which applies to paintings, music compositions, and statues. In a complementary manner, rigidity is a property that describes the dynamics of how the type may be instantiated. Rigid types classify their instances necessarily while anti-rigid types, including *roles* (e.g., 'Wife') and *phases* (e.g., 'Child'), classify their instances contingently. A rigid sortal type providing a uniform principle of identity for its instances is called a *kind* (e.g., 'Person').

One can continue to describe UFO's taxonomy further, but for more examples and formalization, we refer the reader to [5,13]. For the scope of this work, it is essential to mention that the presented taxonomy can be extended by introducing

new types and by instantiating at the level of individuals. Both options can be accomplished with the help of existing modeling tools.

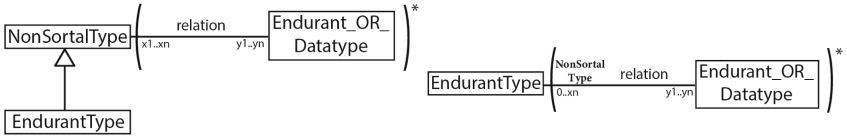
OntoUML is a language designed to extend UML with the concepts of UFO. OntoUML defines a set of constructs and semantically-motivated syntactical constraints tailored for ODCMs [3]. In other words, OntoUML shifts the inherent complexity of reality towards the language's definition in such a way that every syntactically valid model represents a sound ontology in terms of UFO [6]. With this in mind, patterns for ODCM's abstractions were initially developed in terms of OntoUML [12,18].

Although there are several tools that provide support in developing OntoUML models (e.g., OLED [11]), the models that are used as illustrations in this paper were developed using the OntoUML plugin[1] for Visual Paradigm[2].

3 Towards Formalization of Abstraction Rules

Reoccurring ontology modeling situations can be represented by means of *ontology design patterns* [7]. It has been shown, that OntoUML — which was designed to reflect the underlying ontological micro-theories of UFO — is an ontology pattern language [24]. In other words, models expressed in it are constructed by an exemplification of the provided patterns.

For this reason, the graph-rewriting rules for ODCM's abstractions in [12,18] were designed as patterns for models expressed in OntoUML. Thus, in order to apply them, one needs to substitute the matching model with the replacement in its exemplification. In the following, we illustrate the abstraction process with one of the rules, namely `Rule R2` [12].



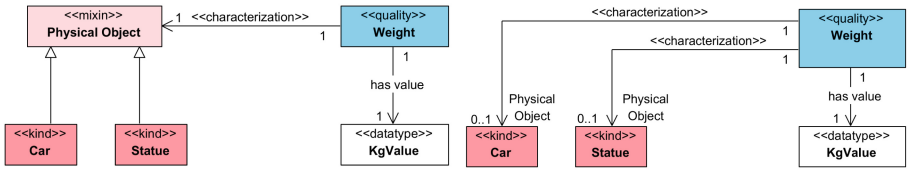
(a) Matching graph. (b) Replacing graph.

Fig. 1. `Rule R2` for abstracting a non-sortal type (from [12]).

`Rule R2` should be interpreted as part of the ruleset proposed in [12]. The rationale of that set in a nutshell is: (a) abstract all information in a model by transferring it to the *kinds* in that domain; (b) in order to do that, we need to move all information from lower-level sortals to their upper classes until reaching these kinds and, subsequently, eliminating from the model these lower-level sortal subtypes (`Rule R3`); (c) before performing Step *(b)*, we need to first move all information from non-sortal types to their sortal subtypes, subsequently, eliminating these non-sortal types. `Rule R2` captures exactly Step *(c)* as illustrated in Fig. 1.

[1] https://github.com/OntoUML/ontouml-vp-plugin.
[2] https://www.visual-paradigm.com/.

We provide an exemplification of the rule by replacing the placeholders with concrete classes. In Fig. 2b, you can see how this rule is applied to the ODCM given in Fig. 2a. In this example, the pattern in Fig. 1a can be found in the model twice, namely for the cases: (1) "`Car` as `Physical Object` has quality `Weight`", and (2) "`Statue` as `Physical Object` has quality `Weight`".



(a) Exemplification of the matching graph. (b) Exemplification of the replacing graph.

Fig. 2. Possible exemplification of `Rule R2`.

In order to understand the abstraction process, we need to distinguish the following levels (see Fig. 3): (1) the domain-independent level of UFO (on which the patterns are originally described), (2) the level of domain types that exemplify these patterns in a conceptual model, e.g., in OntoUML, and (3) the level of individuals on which the model can be instantiated.

When abstracting, we are changing the pattern exemplification by replacing the sub-model at the level of domain types only. Since the order of pattern applications is not fixed, one may also have another version of ODCM-2 produced by applying the rule first to `Statue`, instead of `Car` (for details see Sect. 4).

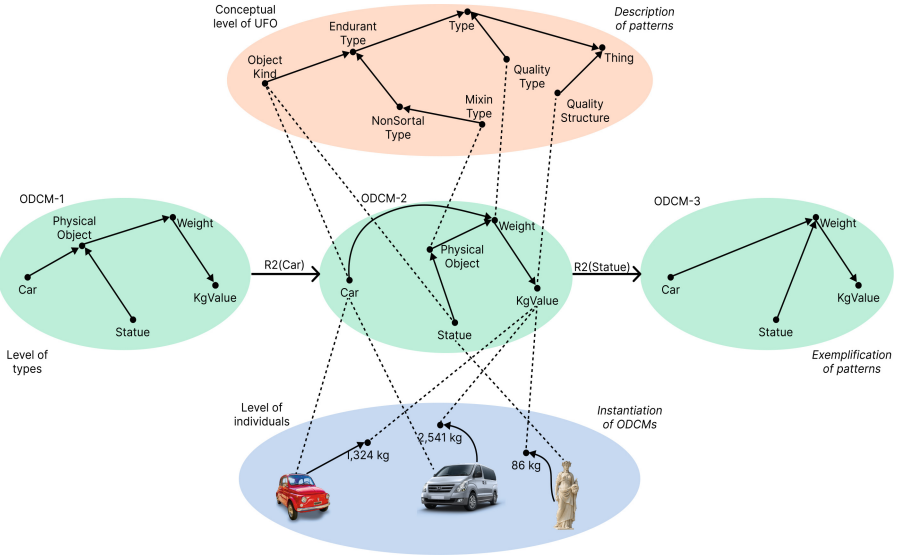


Fig. 3. ODCM abstractions produced by applying `Rule R2`.

To formalize the given pattern in first-order modal logic we reuse the formalization of UFO [14], which is partially reproduced here in order to ease understanding. Following it, we also use the first-order modal logic QS5 plus the Barcan formula and its converse [4,16]. In other words, we assume a fixed domain of entities for every possible world. The modal operators of necessity ($\Box$) and possibility ($\Diamond$) are used with their usual meaning, and specifically, since we are working in QS5, they quantify over all possible worlds, i.e., accessibility is interpreted over a universal relation.

First, since we want to stay in the first-order paradigm, we reify types into objects. Thus, we need to distinguish two kinds of first-order objects, namely *types*, which are possibly instantiated by something, and *individuals*, which are necessarily not instantiated by anything. For this purpose, the *instantiation relation* (::) is introduced with the possibility to describe first- and second-order types, i.e., respectively objects that are types instantiated by individuals, and objects that are types instantiated by types. In the following axioms, all free variables are implicitly universally quantified.

$$\mathrm{Type}(x) \leftrightarrow \Diamond(\exists y\,(y :: x)) \qquad x :: y \rightarrow (\mathrm{Type}(x) \vee \mathrm{Individual}(x))$$
$$\mathrm{Individual}(x) \leftrightarrow \Box(\neg\exists y\,(y :: x)) \qquad \neg\exists x, y, z\,(\mathrm{Type}(x) \wedge x :: y \wedge y :: z)$$

The *specialization relation* between types ($\sqsubseteq$) is defined in terms of necessary extensional inclusion, i.e., the inclusion of the instances. The specialization relation is quasi-reflexive and transitive. Also, whenever two types have a common instance, they must share a super-type or a sub-type for this instance.

$$x \sqsubseteq y \leftrightarrow \mathrm{Type}(x) \wedge \mathrm{Type}(y) \wedge \Box(\forall z\,(z :: x \rightarrow z :: y))$$
$$\forall x, y, z\,((z :: x \wedge z :: y \wedge \neg(x \sqsubseteq y) \wedge \neg(y \sqsubseteq x)) \rightarrow (\exists v\,(x \sqsubseteq v \wedge y \sqsubseteq v \wedge z :: v) \vee \exists v\,(v \sqsubseteq x \wedge v \sqsubseteq y \wedge z :: v)))$$
$$x \sqsubseteq y \rightarrow (x \sqsubseteq x \wedge y \sqsubseteq y)$$
$$x \sqsubseteq y \wedge y \sqsubseteq z \rightarrow x \sqsubseteq z$$

Considering Fig. 3, we can say that Athena :: Statue, ObjectKind(Car), and Car $\sqsubseteq$ PhysicalObject. Furthermore, in the formalization of the pattern, by *relation* we mean any relation that is specified in UFO, e.g., `componentOf`, `associatedWith`, and others [14].

Taking into account all the above mentioned, we can formalize the matching pattern from Fig. 1a as the following schema:

$$\bigwedge \begin{array}{l} \exists x, y\,(\mathrm{NonSortal}(x) \wedge \mathrm{EndurantType(y)} \wedge y \sqsubseteq x) \\ \Diamond\exists z\,((\mathrm{EndurantType}(z) \vee \mathrm{Set}(z)) \wedge \mathrm{Relation}(x, z)) \end{array} \tag{1}$$

Then the replacing pattern from Fig. 1b is the following:

$$\bigwedge \begin{array}{l} \exists y\,\mathrm{EndurantType}(y) \\ \Diamond\exists z\,((\mathrm{EndurantType}(z) \vee \mathrm{Set}(z)) \wedge \mathrm{Relation}(y, z)) \end{array} \tag{2}$$

Thus, the formalization of the exemplification given in Fig. 2 is shown in Formalizations 1 and 2. Also, here `KgValue` is the non-empty set of possible values that `Weight` can take, e.g., from Fig. 3:

$$1,324\mathrm{kg} \in \texttt{KgValue} \wedge 2,541\mathrm{kg} \in \texttt{KgValue} \wedge 86\mathrm{kg} \in \texttt{KgValue}$$

Formalization 1. Matching

ObjectKind(Car)
ObjectKind(Statue)
Mixin(PhysicalObject)
Car $\sqsubseteq$ PhysicalObject
Statue $\sqsubseteq$ PhysicalObject
QualityType(Weight)
QualityStructure(KgValue)
characterization(PhysicalObject, Weight)
associatedWith(Weight, KgValue)

Formalization 2. Replacing

ObjectKind(Car)
ObjectKind(Statue)
QualityType(Weight)
QualityStructure(KgValue)
characterization(Car, Weight)
characterization(Statue, Weight)
associatedWith(Weight, KgValue)

Proposition 1 (Rule R2). *Assuming the UFO axiomatisation, every exemplification of the matching graph of Rule R2 entails the corresponding exemplification of the replacing graph.*

Although for this particular case of `Rule R2`, the entailment of the replacing pattern is quite immediate, this is not always the case. In other rules, e.g., in `Rule R3` for abstracting sortal types (again, which proceeds in the opposite direction and moves the relation from lower-level sortal subtypes to kinds [12]), the entailment is possible from the replacing model to the matching model if a specialization of the general sortal type is given.

For example, taking into account the exemplification above we can extend it with a `RentalCar` concept (RentalCar $\sqsubseteq$ Car), the role that `Car` can play when it is rented for a specific `Price`: characterization(RentalCar, Price). Then, the replacing exemplification of the model according to `Rule R3` is: characterization(Car, Price), and if someone provides us with the specialization of the original `Car` type, we can entail the original model from its replacement.

Thus, in the existing rule system [12,18], generated abstractions are not always TI abstractions as suggested by Giunchiglia & Walsh [10], and a full characterisation of how different sequences of rule applications are theory increasing or theory decreasing should be undertaken.

4 Abstractions as Networks of Models

As we have seen, since the rules do not always bring us to TI abstractions, we consider ODCMs generated by them connected by a more general *compatibility relation*. Taking into account that there are different rules, we also consider this mapping process as *an iterative process, where modifications in an ODCM caused by each rule lead to the development of a hierarchy of abstractions*.

Indeed, even very small models allow several possible abstractions, which are not always fully equivalent. Figure 4 illustrates this case, where the concept name near an arrow specifies the type that has been eliminated at the given step. Namely, we can start by abstracting `Customer`, and then sequentially abstract from `Personal Customer` and `Corporate Customer` (or vice versa, the resulting model will be the same). But we may also start by eliminating `Personal Customer`, and the resulting model will contain at least one isolated class.

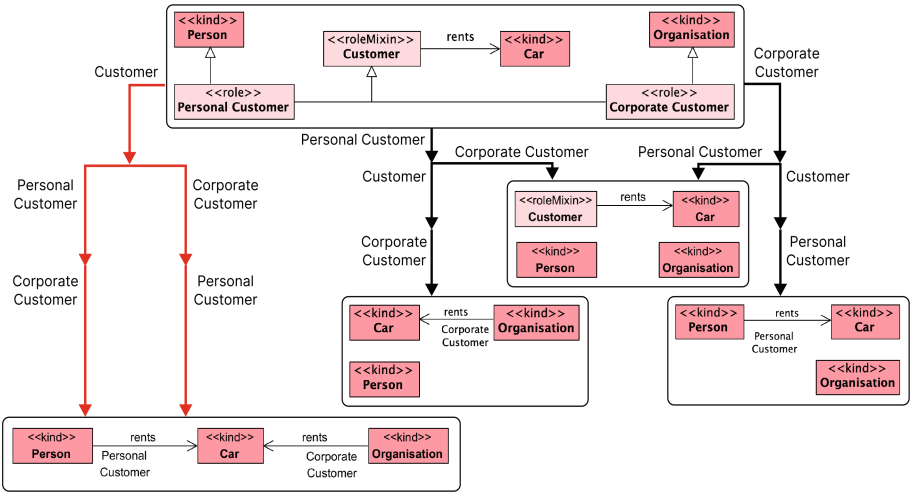


Fig. 4. Hierarchy of abstractions, where a *proper hierarchy* is shown in red (Color figure online).

Although there has been an attempt to specify the order for rule applications (see listings in [18]), those order constraints themselves were part of the methodology for rule application, but not part of the semantics of the rules themselves. As such, formally speaking and without such a constraint, every rule should be used until it is not applicable anymore.

Here, in order to distinguish different types of abstractions, we provide the following definitions.

Definition 1. *An* abstraction of an ODCM *is a model obtained from the given one by applying at least one abstraction rule.*

Definition 2. *A* complete *(or* full*)* abstraction *is a model to which no abstraction rule can be applied.*

Although in practice it is rare, some OntoUML models are complete abstractions of themselves.[3]

Definition 3. *In an ODCM, a* dependent concept *is either an existentially dependent entity or a concept connected to another one via the parthood or generalization relation.*

Definition 4. *A* directly applicable rule *is a rule that does not eliminate a dependent concept to which another rule could be applied.*

Definition 5. *A* primitive abstraction *is an ODCM obtained from a given one by applying a directly applicable rule.*

[3] Examples of such models can be found in the FAIR Catalog of OntoUML/UFO models [1] we mentioned earlier, e.g., `pereira2020ontotrans`.

Definition 6. *A* proper hierarchy of abstractions *is formed by primitive abstractions only.*

Most of the rules reduce the number of classes during their application, and in a nutshell, we do not want to remove those concepts until there are other rules that are applicable to them. In the original model from Fig. 4, `Personal Customer` and `Corporate Customer` are dependent concepts. Thus, only the left part of the tree (shown in red) is formed by primitive abstractions only.

The next question that arises is whether this process of applying rules is finite and to what kind of ODCMs it leads.

Theorem 1. *A complete abstraction of a domain ODCM is always reachable and not empty.*

The proof of the theorem is out of the scope of the paper. Unfortunately, in the worst case, as a complete abstraction, one can obtain a model with a single class.[4] Although tools for conceptual modeling, including Visual Paradigm, provide opportunities to modularise CMs or extract views, the majority of OntoUML models are developed as connected graphs. However, during the abstraction process, this connectivity can be lost.[5]

5 Conclusions

Ontology-driven conceptual models help in increasing the domain comprehensibility and appropriateness (including explainability) of information systems. However, they also reflect the complexity of the domain in which they are created. Thus, understanding and comprehension of such models can become a challenge without proper tool support. Abstraction — a process of providing a summary of a given CM whilst preserving the gist of conceptualization — is one of the approaches to adopt in this case.

Previously, [18] defined abstraction patterns via graph-rewriting rules. However, by just looking at those rules, and without considering formal semantics, it is not always clear whether they actually lead to more general models. Indeed, we have shown that for some rules the model could be entailed by the original one, and for some others, it cannot. A complete characterisation of how different sequences of rule applications change the theory described by an ODCM is the subject of future work.

The approach that we have specified in this paper is operational, where the abstraction process is a mapping from the original model to a modified, more abstract, version. This mapping defines a meta-theoretical relationship between ODCMs endowed with corresponding formal semantics. We have shown how hierarchies of abstraction spaces can be built and argued that this process is always finite and leads to a complete abstraction.

[4] An example of a model whose complete abstraction will include two classes only is `stock-broker2021`.

[5] E.g., for the model `silva2012itarchitecture`.

References

1. Barcelos, P.P., et al.: A FAIR model catalog for ontology-driven conceptual modeling research. In: Ralyté, J., Chakravarthy, S., Mohania, M., Jeusfeld, M.A., Karlapalem, K. (eds.) Conceptual Modeling. ER 2022. LNCS, vol. 13607, pp. 3–17. Springer. Cham (2022). https://doi.org/10.1007/978-3-031-17995-2_1
2. Bork, D.: Conceptual modeling and artificial intelligence: Challenges and opportunities for enterprise engineering. In: Aveiro, D., Proper, H.A., Guerreiro, S., de Vries, M. (eds.) Advances in Enterprise Engineering XV. EEWC 2021. LNCS, vol. 441, pp. 3–9 Springer. Cham (2022). https://doi.org/10.1007/978-3-031-11520-2_1
3. de Carvalho, V.A., Almeida, J.P.A., Guizzardi, G.: Using reference domain ontologies to define the real-world semantics of domain-specific languages. In: Jarke, M., et al. (eds.) CAiSE 2014. LNCS, vol. 8484, pp. 488–502. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-07881-6_33
4. Fitting, M., Mendelsohn, R.L.: First order modal logic. Synthese library 277, Kluwer Acad. Publ, Boston (1998)
5. Fonseca, C.M., Porello, D., Guizzardi, G., Almeida, J.P.A., Guarino, N.: Relations in ontology-driven conceptual modeling. In: Laender, A.H.F., Pernici, B., Lim, E.-P., de Oliveira, J.P.M. (eds.) ER 2019. LNCS, vol. 11788, pp. 28–42. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-33223-5_4
6. Fonseca, C.M. et al.: Ontology-driven conceptual modelling as a service. In: Proceedings of JOWO. CEUR, vol. 2969 (2021). www.ceur-ws.org/Vol-2969/paper29-FOMI.pdf
7. Gangemi, A., Presutti, V.: Ontology design patterns. In: Staab, S., Studer, R. (eds.) Handbook on Ontologies. IHIS, pp. 221–243. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-540-92673-3_10
8. Ghidini, C., Giunchiglia, F.: A semantics for abstraction. Technical report DIT-03-082, University of Trento (2003)
9. Ghidini, C., Giunchiglia, F.: What is local models semantics? In: Bouquet, P., Serafini, L., Thomason, R.H. (eds.) Perspectives on Contexts. CSLI (2008)
10. Giunchiglia, F., Walsh, T.: A theory of abstraction. Artif. Intell. **57**(2), 323–389 (1992)
11. Guerson, J., Sales, T.P., Guizzardi, G., Almeida, J.P.A.: OntoUML lightweight editor: a model-based environment to build, evaluate and implement reference ontologies. In: Kolb, J. (eds.) 2015 IEEE 19th International Enterprise Distributed Object Computing Workshop, EDOC Workshops 2015, Adelaide, SA, Australia, pp. 144–147. IEEE Computer Society (2015), https://doi.org/10.1109/EDOCW.2015.17
12. Guizzardi, G., Figueiredo, G., Hedblom, M.M., Poels, G.: Ontology-based model abstraction. In: Proceedings of the 13th International Conference on Research Challenges in Information Science (RCIS). pp. 1–13. IEEE (2019) https://doi.org/10.1109/RCIS.2019.8876971
13. Guizzardi, G., Fonseca, C.M., Almeida, J.P.A., Sales, T.P., Benevides, A.B., Porello, D.: Types and taxonomic structures in conceptual modeling: A novel ontological theory and engineering support. Data Knowl. Eng. **134**, 101891 (2021). ISSN 0169-023X. https://doi.org/10.1016/j.datak.2021.101891
14. Guizzardi, G., Botti Benevides, A., Fonseca, C.M., Porello, D., Almeida, J.P.A., Prince Sales, T.: UFO: unified foundational ontology. Appl. Ontol. **17**(1), 167–210 (2022). https://doi.org/10.3233/AO-210256

15. Hobbs, J.R.: Granularity. In: Proceedings of IJCAI, vol. 1. Morgan Kaufmann (1985)
16. Kracht, M., Kutz, O.: Logically possible worlds and counterpart semantics for modal logic. In: Handbook of the Philosophy of Science. Elsevier (2007)
17. Plaisted, D.A.: Theorem proving with abstraction. Artif. Intell. **16**(1), 47–108 (1981)
18. Romanenko, E., et al.: Abstracting ontology-driven conceptual models: Objects, aspects, events, and their parts. In: Guizzardi, R., Ralyté, J., Franch, X. (eds.) Research Challenges in Information Science. RCIS 2022. LNCS, vol. 446, pp. 372–388 Springer. Cham (2022). https://doi.org/10.1007/978-3-031-05760-1_22
19. Romanenlo, E., et al.: What do users think about abstractions of ontology-driven conceptual models? In: Nurcan, S., Opdahl, A.L., Mouratidis, H., Tsohou, A. (eds.) Research Challenges in Information Science: Information Science and the Connected World. RCIS 2023. LNBIP, vol. 476, pp. 53–68. Springer, Cham (2023). https://doi.org/10.1007/978-3-031-33080-3_4
20. Saitta, L., Zucker, J.D.: Abstraction in Artificial Intelligence and Complex Systems. Springer, New York (2013). https://doi.org/10.1007/978-1-4614-7052-6_3
21. Tenenberg, J.D.: Preserving consistency across abstraction mappings. In: IJCAI, pp. 1011–1014 (1987). www.ijcai.org/Proceedings/87-2/Papers/090.pdf
22. Verdonck, M., Gailly, F.: Insights on the use and application of ontology and conceptual modeling languages in ontology-driven conceptual modeling. In: Comyn-Wattiau, I., Tanaka, K., Song, I.-Y., Yamamoto, S., Saeki, M. (eds.) ER 2016. LNCS, vol. 9974, pp. 83–97. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46397-1_7
23. Villegas Niño, A.: A filtering engine for large conceptual schemas. Ph.D. thesis, Universitat Politècnica de Catalunya (2013)
24. Zambon, E., Guizzardi, G.: Formal definition of a general ontology pattern language using a graph grammar. In: Proceedings of the 2017 Federated Conference on Computer Science and Information Systems. IEEE (2017). https://doi.org/10.15439/2017f001