Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence (IJCAI-24)
Sister Conferences Best Papers Track

Inferring Ontological Categories of OWL Classes Using Foundational Rules* (Extended Abstract)

Pedro Paulo F. Barcelos[1] , **Tiago Prince Sales**[1] , **Elena Romanenko**[2] ,
João Paulo A. Almeida[3] , **Gal Engelberg**[4] , **Dan Klein**[4] , **Giancarlo Guizzardi**[1]
[1]Semantics, Cybersecurity & Services (SCS), University of Twente, Enschede, The Netherlands
[2]KRDB Research Centre for Knowledge and Data, Free University of Bozen-Bolzano, Bolzano, Italy
[3]Ontology & Conceptual Modeling Research Group, Federal University of Espírito Santo, Vitória, Brazil
[4]Accenture Labs, Israel
{p.p.favatobarcelos, t.princesales}@utwente.nl, eromanenko@unibz.it, jpalmeida@inf.ufes.br,
{gal.engelberg, dan.klein}@accenture.com, g.guizzardi@utwente.nl

Abstract

Several efforts that leverage the tools of formal ontology have demonstrated the fruitfulness of considering key metaproperties of classes in ontology engineering. Despite that, it is still a common practice to apply representation schemes and approaches–such as OWL–that do not benefit from identifying ontological categories and simply treat all classes in the same manner. In the original study, we proposed an approach to support the automated classification of classes into the ontological categories underlying the (g)UFO foundational ontology. We proposed a set of inference rules derived from (g)UFO's axiomatization that, given an initial classification of the classes in an OWL ontology, supports the inference of the classification for the remaining classes in the ontology. We formalized these rules, implemented them in a tool, and assessed them against a catalog of ontologies.

1 Introduction

Ontologies have been promoted both as a reference model of consensus to support reuse and interoperability of domain conceptualizations **(purpose 1)** or as an explicit, declarative, and machine-processable artifact coding a domain model to enable automated reasoning **(purpose 2)**. This duality, however, points to different (and even conflicting) sets of quality criteria and requirements that these artifacts and the representation languages in which they are expressed should meet.

To serve **purpose 1**, an ontology should be constructed in a way that maximizes the expressivity in capturing fundamental aspects of the underlying domain and in making explicit the underlying ontological commitments. In contrast, to support **purpose 2**, it should be built in a way that supports decidable and computationally tractable automated reasoning. As it is well known, there is a trade-off between expressivity and maintaining these desirable computational properties [Levesque and Brachman, 1987]. In the context of the Semantic Web, the Web Ontology Language (OWL) was designed to maximize the latter side of this trade-off and, despite its limited expressivity (when compared to the First Order [Guizzardi *et al.*, 2021] or Higher Order Logics [Nicola, 2021] that have been used in ontology representation), OWL has been very successful in giving rise to a plethora of specifications in a large variety of domains.

These OWL ontologies, however, do not always live up to the promise of promoting interoperability and reuse in complex and critical domains [Guizzardi *et al.*, 2010; Bittner, 2009; Flügel *et al.*, 2022]. The reason for that goes beyond mere logical expressivity. As beautifully expressed by Varzi's dictum: *"No ontology without Ontology"* [Varzi, 2019], to support **purpose 1** ontologies should be constructed with a fuller consideration of (the discipline of) Ontology, i.e., with the explicit support of true ontological theories.

To offer that kind of foundational support for the construction of OWL ontologies, some of us have proposed gUFO [Almeida *et al.*, 2020], an OWL super-structure implementing the Unified Foundational Ontology (UFO) [Guizzardi *et al.*, 2022]. gUFO can support the systematic alignment of classes in OWL ontologies to its foundational categories, increasing the quality of these ontologies (in the sense of **purpose 1**) by design.

However, there is a vast amount of OWL ontologies that have been built without such support, and expecting the ontologies to be manually re-engineered is unrealistic. This is the problem we addressed in the original paper. We proposed a set of inference rules derived from UFO's and gUFO's axiomatization that, given an initial seeding, are triggered to infer the gUFO classification for the remaining classes. Further, we implemented these rules in a tool that operates in an automatic or semi-automatic manner. Finally, we tested this tool against a catalog of UFO-based ontologies that collects models designed by a variety of users and domains.

2 Unified Foundational Ontology

In UFO, endurants are object-like individuals. They endure in time and are able to qualitatively change while maintaining their identities [Guizzardi *et al.*, 2021]. Examples include

*Original paper published as *Barcelos et al., 2023*

ordinary objects of everyday life, such as a person; particularized relational properties, such as a marriage; and existentially dependent aspects of objects, such as the weight of a car. Endurant types are distinguished by metaproperties such as *rigidity* and *sortality*.

Rigidity describes the dynamics of how a type may be instantiated. Every type is either rigid, anti-rigid, or semi-rigid; no type is both rigid and anti-rigid, rigid and semi-rigid, and anti-rigid and semi-rigid; there is no rigid type that specializes an anti-rigid type; and there is no semi-rigid type that specializes an anti-rigid type [Guizzardi *et al.*, 2021].

Sortality defines the relation between a type and the principles of identity of its instances. A type is a sortal if all of its instances follow the same identity principle. Examples include the types "Person" and "Child". If a sortal type *supplies* an identity principle to its instances, it is a kind. Examples of kinds are "Person" and "Organization". As a complement, non-sortal types aggregate properties that are common to different sortals. An example of such a type is "Work of Art", which applies to paintings and statues.

Every individual must follow a unique identity principle and thus instantiate exactly one kind. No type is both a sortal and a non-sortal. Every sortal specializes a unique kind; a non-sortal cannot specialize a sortal; and non-sortals do not have direct instances, their instances are also instances of a sortal that either specializes the non-sortal, or that specializes a common non-sortal supertype [Guizzardi *et al.*, 2021]. Combining the sortality and rigidity aspects, we obtain the endurant types' leaf categories: **kind**, **subkind**, **phase**, **role**, **category**, **phase mixin**, **role mixin**, and **mixin**.

To be used on the Semantic Web, a lightweight implementation of UFO was created and named **gUFO** [Almeida *et al.*, 2020]. To reuse gUFO, one may instantiate and/or specialize the various classes, object properties, and data properties it provides. A key feature of gUFO is that it includes two class taxonomies: one with classes whose instances are individuals, such as `gufo:Endurant` and `gufo:Object`; and another with classes whose instances are types, such as `gufo:Kind` and `gufo:Category`. The following is an example (in Turtle notation) of how to apply gUFO.

```
:Person rdfs:subClassOf gufo:Object ;
        rdf:type gufo:Kind .
:Child rdfs:subClassOf :Person ;
       rdf:type gufo:Phase .
```

3 Inference Rules

Our objective in the original study was to develop a system to infer the UFO meta-categories of classes in an OWL ontology given an initial seeding. More precisely, consider ET as the set of leaf classes defined in gUFO's taxonomy of endurant types, namely `gufo:Kind`, `gufo:Subkind`, `gufo:Role`, `gufo:Phase`, `gufo:Category`, `gufo:RoleMixin`, `gufo:PhaseMixin`, and `gufo:Mixin`. Given an OWL ontology O containing a set of classes C and an initial mapping M (*seeding*) between classes in C and classes in ET using `rdf:type`, what additional `rdf:type` mappings between classes of C and ET can we infer?

This task would be facilitated if the complete axiomatization of UFO was implemented in gUFO. Since this is not the case, we proposed a set of rules based on UFO's and gUFO's axiomatizations. In the following, we specify inference rules in First Order Logic. Consider that all free variables are universally quantified, having all their occurring formulas as their scope. We assume that our domain of quantification only includes types. The symbol $\oplus$ is used here to represent an exclusive or operator and $\nexists$ as a shorthand for $\neg\exists$.

We start by declaring our $subClassOf$ relation, which is reflexive (R01) and transitive (R02), directly corresponding to the `rdfs:subClassOf` property and compatible with the UFO's *specialization* relation.

R01: $subClassOf(x, x)$

R02: $subClassOf(x, y) \land subClassOf(y, z) \rightarrow subClassOf(x, z)$

Rules R03–R16 were mapped from gUFO's taxonomy of endurant types. Capitalized unary predicates (e.g., $Sortal$, $Kind$, $Category$) directly correspond to gUFO's classes in its taxonomy of types, implications correspond to specialization relations, and equivalence and exclusive disjunctions are used to map disjoint unions.

R03: $EndurantType(x) \leftrightarrow RigidType(x) \oplus NonRigidType(x)$

R04: $NonRigidType(x) \leftrightarrow AntiRigidType(x) \oplus SemiRigidType(x)$

R05: $EndurantType(x) \leftrightarrow Sortal(x) \oplus NonSortal(x)$

R06: $Kind(x) \rightarrow RigidType(x) \land Sortal(x)$

R07: $SubKind(x) \rightarrow RigidType(x) \land Sortal(x)$

R08: $\nexists x(Kind(x) \land SubKind(x))$

R09: $Role(x) \rightarrow AntiRigidType(x) \land Sortal(x)$

R10: $Phase(x) \rightarrow AntiRigidType(x) \land Sortal(x)$

R11: $\nexists x(Phase(x) \land Role(x))$

R12: $Category(x) \rightarrow NonSortal(x) \land RigidType(x)$

R13: $RoleMixin(x) \rightarrow NonSortal(x) \land AntiRigidType(x)$

R14: $PhaseMixin(x) \rightarrow NonSortal(x) \land AntiRigidType(x)$

R15: $\nexists x(PhaseMixin(x) \land RoleMixin(x))$

R16: $Mixin(x) \rightarrow NonSortal(x) \land SemiRigidType(x)$

We need to complement these rules with R17–R21, which require endurant types to instantiate at least one of the leaf types in gUFO's taxonomy of types. In R19, we equate semi-rigid types with mixins, which are non-sortal semi-rigid types because gUFO does not allow semi-rigid sortals.

R17: $RigidType(x) \rightarrow Category(x) \lor Kind(x) \lor SubKind(x)$

R18: $AntiRigidType(x) \rightarrow Role(x) \lor Phase(x) \lor RoleMixin(x) \lor PhaseMixin(x)$

R19: $SemiRigidType(x) \rightarrow Mixin(x)$

R20: $Sortal(x) \rightarrow Kind(x) \lor Phase(x) \lor Role(x) \lor SubKind(x)$

R21: $NonSortal(x) \rightarrow Category(x) \lor PhaseMixin(x) \lor RoleMixin(x) \lor Mixin(x)$

As gUFO does not implement restrictions regarding the rigidity of an endurant type and of the types it specializes and generalizes, we implemented rules R22–R25. R22 and R23 state that no rigid or semi-rigid type can specialize an anti-rigid type. R24 states that for every anti-rigid sortal type x that specializes a category y, a rigid non-sortal type, there is at least a rigid sortal type z of which x is a subclass and which is a subclass of y. R25 states that for every mixin, there is at least one rigid type and one anti-rigid type that specialize it.

R22: $RigidType(x) \land subClassOf(x,y) \rightarrow \neg AntiRigidType(y)$

R23: $SemiRigidType(x) \land subClassOf(x,y) \rightarrow \neg AntiRigidType(y)$

R24: $subClassOf(x,y) \land AntiRigidType(x) \land Sortal(x) \land Category(y) \rightarrow \exists z(subClassOf(x,z) \land subClassOf(z,y) \land RigidType(z) \land Sortal(z))$

R25: $Mixin(x) \rightarrow \exists y,z(subClassOf(y,x) \land RigidType(y) \land subClassOf(z,x) \land AntiRigidType(z))$

We implemented rules R26–R28 regarding sortality constraints. R26 states that every type that a kind specializes (which is not itself) is a non-sortal. R27 states that non-sortal types can only specialize non-sortal types. R28 states that every sortal must specialize a unique kind.

R26: $x \neq y \land Kind(x) \land subClassOf(x,y) \rightarrow NonSortal(y)$

R27: $NonSortal(x) \land subClassOf(x,y) \rightarrow NonSortal(y)$

R28: $Sortal(x) \rightarrow \exists! y(subClassOf(x,y) \land Kind(y))$

To assert in our rule R31 that non-sortals do not have direct instances and classify individuals of at least two different kinds, we need a rule stating that every non-sortal must be a superclass (or a sibling) of at least two sortals that specialize different kinds. To do that, we first define the relations $shareKind$ and $shareSuperClass$. R29 states that the types x and y share a kind if and only if there is a single kind z such that both x and y specialize it. R30 states that the type x shares a same superclass with the type y if and only if there is at least one type that both x and y specialize.

R29: $shareKind(x,y) \leftrightarrow \exists! z(Kind(z) \land subClassOf(x,z) \land subClassOf(y,z))$

R30: $shareSuperClass(x,y) \leftrightarrow \exists z(subClassOf(x,z) \land subClassOf(y,z))$

R31: $NonSortal(x) \rightarrow \exists y,z(y \neq z \land Sortal(y) \land Sortal(z) \land \neg shareKind(y,z) \land (subClassOf(y,x) \lor shareSuperClass(x,y)) \land (subClassOf(z,x) \lor shareSuperClass(x,z)))$

Rules R32–R34 stem from the constraint that specializations of roles and role mixins always inhere their parent's *relational dependency*. So, phases cannot specialize roles and role mixins (R32), and phase mixins cannot specialize role mixins (R33). Further, whenever a role specializes a phase mixin, it does that by specializing a phase that specializes that phase mixin (R34).

R32: $Phase(x) \land subClassOf(x,y) \rightarrow \neg Role(y) \land \neg RoleMixin(y)$

R33: $PhaseMixin(x) \land subClassOf(x,y) \rightarrow \neg RoleMixin(y)$

R34: $Role(x) \land PhaseMixin(y) \land subClassOf(x,y) \rightarrow \exists z(Phase(z) \land subClassOf(x,z) \land subClassOf(z,y))$

Finally, rule R35 specifies that phases must always have at least one sibling class sharing a common kind, while rules R36 and R37 state that phase mixins must always have at least one sibling sharing a common category.

R35: $Phase(x) \rightarrow \exists y(Phase(y) \land shareKind(x,y) \land \neg subClassOf(x,y) \land \neg subClassOf(y,x))$

R36: $PhaseMixin(x) \rightarrow \exists y(Category(y) \land subClassOf(x,y))$

R37: $PhaseMixin(x) \land Category(y) \land subClassOf(x,y) \rightarrow \exists z(PhaseMixin(z) \land \neg subClassOf(x,z) \land \neg subClassOf(z,x) \land subClassOf(z,y))$

To verify the consistency and validate if the rules were sufficient for our inference needs, we implemented the specified rules in Alloy [Jackson, 2002]. The language is accompanied by a solver tool, which allowed us to generate instances that comply with our rule set and to seek counter-examples that would invalidate the expected theorems from it. The implementation of our inference rules in Alloy is available in a git repository (https://purl.org/scior/alloy).

4 Effectiveness Evaluation

To assess the effectiveness of our inference rules, we implemented them in an open-source command-line Python tool called **Scior**. The tool allows users to pick whether it should reason under a closed-world assumption (CWA) or an open-world assumption (OWA) [Hustadt, 1994]. When it reasons under CWA, it presumes that all classes and properties of an ontology have been declared. So, if a class has no subclass, this means that there is not one. Conversely, when it reasons under OWA, it presupposes that not all elements of an ontology have been declared. So, for any class in the ontology, there might be an undeclared class that specializes it (or which it specializes).

We evaluated Scior from two perspectives, correctness and effectiveness. The former is described in the original paper, while the latter is summarized below.

4.1 Materials

To conduct our evaluation, we needed a set of taxonomies for which the ontological categorization of its classes was known in advance. Since no such dataset existed, we built one from the OntoUML/UFO Catalog [Sales *et al.*, 2023], which contains models that could be directly translated to gUFO.

To assemble our test dataset, we first mapped each of the 140 OntoUML models that composed the catalog into an OWL ontology, keeping only their classes and generalizations. Each class in the source model was mapped into an `owl:Class` and each generalization was mapped into an `rdfs:subClassOf` statement. We also mapped OntoUML's stereotypes into gUFO types. Then, from the resulting OWL ontologies, we extracted each independent taxonomy. This left us with a set of 656 taxonomies.

We filtered out taxonomies with less than 10 classes or that violated UFO's axiomatization, which left us a dataset of 85 taxonomies (with 1624 classes). All of these could be used

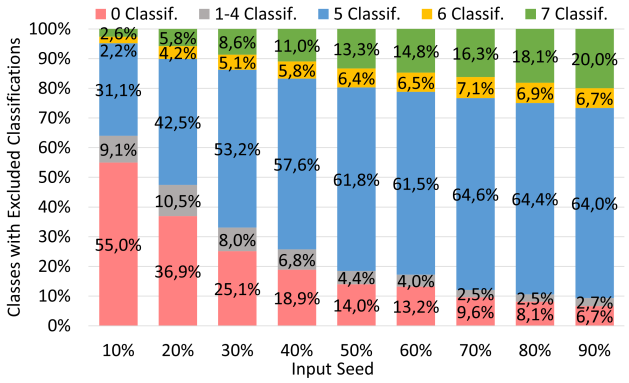

Figure 1: Results of the effectiveness evaluation under OWA.

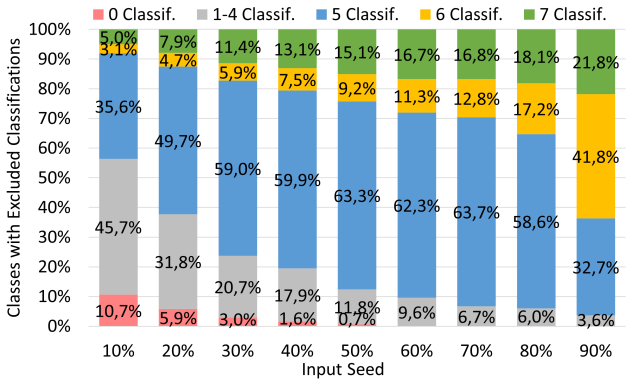

Figure 2: Results of the effectiveness evaluation under CWA.

to test Scior running under OWA, but because some models from the original catalog were not built assuming a CWA scenario, we had to further filter out taxonomies originating from these models. This left us with a second dataset of 67 taxonomies (with 1234 classes) to evaluate Scior under CWA.

4.2 Methods

We quantitatively evaluated the effectiveness of our inference rules by measuring the degree to which Scior could infer, given an initial seeding, the categories of the remaining classes in a taxonomy specified in OWL. We did that by running Scior on different taxonomies, with varying levels of seedings, and under both OWA and CWA modes.

More precisely, for each taxonomy, we repeatedly tested Scior by providing increasing levels of seeding, starting at 10% of the classes and making increments of 10% until we seeded 90% of the taxonomy. For each percentage level, we ran Scior 10 times, each time providing a different randomly selected subset of categorized classes as the seeding. We ran these tests first with Scior on OWA and then on CWA.

For each execution of Scior, we registered how many endurant types' leaf categories could be excluded from the list of classifications a class may receive. Hence, results vary from zero, indicating that no information could be inferred, to seven leaf categories, representing that Scior could determine the exact classification of the evaluated class.

4.3 Results and Discussion

Figure 1 and Figure 2 show, respectively, the results obtained for OWA and CWA models. For each level of initial seeding (from 10% to 90%, laid out horizontally), we show the portion of classes that did not have classifications excluded (in red), that had at least one and at most four classifications excluded (in gray), that had exactly five (in blue) and exactly six categories excluded (in yellow), and the classes for which Scior could determine the applicable leaf category (in green). Results are averages for 10 executions with a randomly selected subset of categorized classes for seeding.

The more classes excluded, the more effective is our proposed set of rules. Results in which no classification could be excluded indicate that Scior could not infer any ontological property for a class and that no new knowledge was added to it, while results that exclude seven categories determine the final classification of a class. Intermediate results, where at least one and at most six categories were excluded, contribute to the overall increase of knowledge as they reduce the number of possibilities users have when manually deciding the final classification for a class, leading to an easier and less error-prone decision-making process.

These results reveal a clear correlation between the amount of input provided and the higher number of excluded categories for both cases. The charts exhibit better classification results for CWA models than for OWA ones. Even though the total exclusion of possibilities is just slightly higher for the former, the higher percentages of exclusion of intermediate numbers of categories are noteworthy. Another interesting result is the expressive portion of cases that lead to the exclusion of five categories.

We evidenced the effectiveness of Scior for ontological categorization in both OWA and CWA. Scior performed better under CWA, always excluding some categories when more than 50% of classes were given as seed. Although OWA models had on average 20% of their classes fully classified for a seeding level of 90%, they still present a percentage of zero exclusions for this amount of input (6.7%). In both cases, the percentage of models with at least five categories excluded is higher than 50% for a seeding of 20%.

5 Final Remarks

We presented a set of inference rules that (semi)automatically support users in grounding their existing OWL ontologies in terms of the categories provided by the foundational ontology UFO as implemented in the gUFO OWL super-structure. We have tested these rules for consistency using the formal language Alloy and its model-finding computational support. The rules were then implemented in a free open-source software named Scior. We evidenced the efficiency of the inference rules and Scior in both open and closed-world assumption scenarios via an experiment that used diverse models from a catalog created by different modelers, for a wide range of domains and purposes, and having different quantities of classes. The results showed that, on average, Scior could automatically exclude at least five out of eight leaf classifications with only 20% of the classes provided as seeds.

References

[Almeida *et al.*, 2020] João Paulo A. Almeida, Ricardo A. Falbo, Giancarlo Guizzardi, and Tiago P. Sales. gUFO: A Lightweight Implementation of the Unified Foundational Ontology (UFO). Technical report, online: https://purl.org/nemo/doc/gufo, 2020. Accessed: 2024-04-25.

[Barcelos *et al.*, 2023] Pedro Paulo F. Barcelos, Tiago Prince Sales, Elena Romanenko, João Paulo A. Almeida, Gal Engelberg, Dan Klein, and Giancarlo Guizzardi. Inferring Ontological Categories of OWL Classes Using Foundational Rules. In *Formal Ontology in Information Systems - Proceedings of the 13th International Conference (FOIS 2023), Sherbrooke, Quebec, Canada, July 17-20, 2013 and Virtual Event, September 18-20, 2013*, volume 377 of *Frontiers in Artificial Intelligence and Applications*, pages 109–124. IOS Press, 2023.

[Bittner, 2009] Thomas Bittner. Logical properties of foundational mereogeometrical relations in bio-ontologies. *Applied Ontology*, 4:109–138, 2009. 2.

[Flügel *et al.*, 2022] Simon Flügel, Martin Glauer, Fabian Neuhaus, and Janna Hastings. When one Logic is Not Enough: Integrating First-order Annotations in OWL Ontologies, 2022.

[Guizzardi *et al.*, 2010] Giancarlo Guizzardi, F. Baião, M. Lopes, and R. Falbo. The Role of Foundational Ontologies for Domain Ontology Engineering: An Industrial Case Study in the Domain of Oil and Gas Exploration and Production. *International Journal of Information System Modeling and Design (IJISMD)*, 1(2):1–22, 2010.

[Guizzardi *et al.*, 2021] Giancarlo Guizzardi, Claudenir M. Fonseca, João Paulo A. Almeida, Tiago Prince Sales, Alessander Botti Benevides, and Daniele Porello. Types and taxonomic structures in conceptual modeling: A novel ontological theory and engineering support. *Data & Knowledge Engineering*, 134:101891, 2021.

[Guizzardi *et al.*, 2022] Giancarlo Guizzardi, Alessander Botti Benevides, Claudenir M. Fonseca, Daniele Porello, João Paulo A. Almeida, and Tiago Prince Sales. UFO: Unified Foundational Ontology. *Applied Ontology*, 17:167–210, 2022.

[Hustadt, 1994] Ullrich Hustadt. Do we need the closed world assumption in knowledge representation? In *Reasoning about Structured Objects: Knowledge Representation Meets Databases, Proceedings of 1st Workshop KRDB'94, Saarbrücken, Germany, September 20-22, 1994*, volume 1 of *CEUR Workshop Proceedings*. CEUR-WS.org, 1994.

[Jackson, 2002] Daniel Jackson. Alloy: a lightweight object modelling notation. *ACM Transactions on Software Engineering and Methodology*, 11(2):256–290, apr 2002.

[Levesque and Brachman, 1987] Hector J. Levesque and Ronald J. Brachman. Expressiveness and tractability in knowledge representation and reasoning. *Computational Intelligence*, 3(1):78–93, 1987.

[Nicola, 2021] João Rafael Moraes Nicola. *A formal analysis of Identity and Sortality in the Unified Foundational Ontology (UFO)*. Phd thesis, Federal University of Espírito Santo, 2021.

[Sales *et al.*, 2023] Tiago Prince Sales, Pedro Paulo F. Barcelos, Claudenir M. Fonseca, Isadora Valle Souza, Elena Romanenko, César Henrique Bernabé, Luiz Olavo Bonino da Silva Santos, Mattia Fumagalli, Joshua Kritz, João Paulo A. Almeida, and Giancarlo Guizzardi. A FAIR catalog of ontology-driven conceptual models. *Data & Knowledge Engineering*, 147:102210, 2023.

[Varzi, 2019] Achille C. Varzi. Carnapian Engineering. In *Ontology Makes Sense. Essays in Honor of Nicola Guarino*, pages 3–23. IOS Press, 2019.