Mathematical Logic for Life Science Ontologies

Carsten Lutz[1] and Frank Wolter[2]

[1] Department of Computer Science, University of Bremen, Germany
[2] Department of Computer Science, University of Liverpool, UK
clu@informatik.uni-bremen.de,frank@csc.liv.ac.uk

Abstract. We discuss how concepts and methods introduced in mathematical logic can be used to support the engineering and deployment of life science ontologies. The required applications of mathematical logic are not straighforward and we argue that such ontologies provide a new and rich family of logical theories that wait to be explored by logicians.

1 Introduction

In recent years, life sciences such as medicine and biology have experienced an abundant growth of terminology. This is mainly due to increasing interdisciplinarity and significant scientific advances such as the decoding of the human genome, which have established novel research areas such as pharmacogenomics (study of the effect of genes on drug response). Indeed, life science terminology has grown to a point where it information processing and exchange is seriously hampered as it can no longer be guaranteed that multiple parties interpret the data in the same way and using the same terminology. To address this problem, there is a strong trend to construct reference terminologies, which list a standard vocabulary to be used in information processing and data exchange, and which also fixes the meaning of each vocabulary item.

Reference terminologies, which are nowadays usually called *ontologies*, are often given in a logical language to obtain a formal semantics and make use of automated reasoning technology. More specifically, a logical signature is used to define the vocabulary and a (finitely axiomatized) logical theory defines the meaning of the vocabulary. Such ontologies are typically formulated in fragments of first-order logic and are thus amenable to the concepts and tools developed in mathematical logic. The aim of this paper is to provide a glimpse into the field of life science ontologies for readers with a mathematical logic background, to demonstrate that concepts and tools from mathematical logic can be fruitfully applied to engineering problems for ontologies, and to argue that, conversely, life science ontologies provide a rich landscape of logical theories that present interesting and novel challenges for logicians.

The paper is divided into two parts. In the first part, we introduce a typical and successful example of a medical ontology: SNOMED CT, the Systematized Nomenclature of Medicine, Clinical Terms. We give an idea of what SNOMED CT is, how it is developed and kept up to date, what its main applications are, and what kind of logical axioms are used to define the logical theory unterlying it. In

the second part of this paper, we show how concepts from mathematical logic can be applied to ontologies in general, and to SNOMED CT in particular. As concrete examples for successful such applications, we consider conservative extensions as a tool for achieving modularity of ontologies and uniform interpolants as a tool for obtaining an axiomation of a relevant fragment of an ontology. We close with some hints to other techiques from mathematical logic that potentially have interesting applications in the area of ontologies.

2 SNOMED CT

A large number of medical ontologies have emerged in recent years, including the National Cancer Institute (NCI) thesaurus [11], the Foundational Model of Anatomy (FMA) [1], Galen [26], and SNOMED CT [22, 28]. Among these ontologies, SNOMED CT plays a particularly prominent role as it is used to produce the standard healthcare terminology for a number of countries such as the US, Canada, the UK, and Sweden.

2.1 Overview

SNOMED CT is a comprehensive clinical healthcare terminology that comprises more than 400.000 vocabulary items and almost the same number of logical axioms. Since 2007, the intellectual property rights of SNOMED CT are held by a not-for-profit association called International Heath Terminology Standards Development Organisation (IHSTDO). Currently, IHSTDO is made of nine nations, including the ones listed above. The general goal is to develop SNOMED CT into *the* global clinical terminology, thus "enabling clinicians, researchers and patients to share and exchange healthcare and clinical data worldwide" [2].

Technically, SNOMED CT is a finite set of first-order predicate logic sentences and uses standard Tarski semantics. Medical terms are regarded as unary predicates such as

$$\mathsf{Disease}, \quad \mathsf{Appendicitis}, \quad \mathsf{Inflammation}, \quad \mathsf{Arthritis},$$

and binary predicates such as

$$\mathsf{Associated\_Morphology}, \quad \mathsf{Finding\_Site}, \quad \mathsf{Procedure\_Site}.$$

An example of a typical SNOMED CT axiom is as follows:

$$\forall x\,(\mathsf{Appendicitis}(x) \rightarrow \mathsf{Disease}(x) \wedge \exists y(\mathsf{Associated\_Morphology}(x,y) \wedge \mathsf{Inflammation}(y))).$$

Though easily translatable, the official syntax of SNOMED CT is not classical first-order syntax and does not contain variables and first-order quantifiers. Instead, the syntax is based on so-called description logics (DLs), a family of

knowledge representation formalisms often used as ontology languages [6]. In description logic, a theory generally consists of axioms of the form

$$\forall x\ (\varphi(x) \rightarrow \psi(x)) \quad \text{and} \quad \forall x\ (\varphi(x) \leftrightarrow \psi(x)),$$

where $\varphi(x)$ and $\psi(x)$ are first-order formulas with one free variable using unary and binary predicates (formulated in DL syntax).

The admissible form of $\varphi(x)$ and $\psi(x)$ depends on the description logic used. Without going into any details, we briefly give three examples of descriptions logics in increasing order of expressivity. The most inexpressive one is $\mathcal{EL}$ [5], where $\varphi(x)$ and $\psi(x)$ are composed using conjunction and guarded existential quantification. Although inexpressive, $\mathcal{EL}$ is a popular ontology language and is used, for example, for SNOMED CT. When $\mathcal{EL}$ is extended with disjunction, negation, and guarded universal quantification, we obtain the description logic $\mathcal{ALC}$. For example, the NCI thesaurus is formulated in this language. By further admitting guarded counting quantifiers and a number of other constructors, one obtains the description logic OWL DL (the Web Ontology Language) that, for example, is used for the medical ontology GALEN.

In many ontologies, the structure of axioms is restricted further by imposing *unfoldability*. In this case, $\varphi(x)$ has to be an atom $P(x)$, and thus axioms $\forall x\ (P(x) \rightarrow \psi(x))$ give necessary conditions for being a P and axioms $\forall x\ (P(x) \leftrightarrow \psi(x))$ give both necessary and sufficient conditions. In addition, no predicate name may be used on the left-hand side of more than one axiom and an acyclicity condition is enforced which basically says that ψ may not refer to P, neither directly nor indirectly. Both SNOMED CT and NCI are unfoldable terminologies, whereas GALEN is not.

2.2 Applications and Engineering

The main purpose of SNOMED CT is to produce a hierarchically organized catalogue of medical terms, with more general terms higher up in the hierarchy and more specialized ones further down. Each term is annotated with a natural language description of its meaning, a numerical code that uniquely identifies the term, and a logical axiom that defines the term's meaning (on a high level of abstraction).

The classical application of SNOMED CT is to simply use the catalogue and numerical codes in medical IT applications, without exploiting or having access to the logical axioms [3] (but see below). As an example for this use of SNOMED CT, consider the generation, processing, and storage of medical records. Whether working in a hospital or independently, physicians have to generate a detailed record of every patient visit, including details on findings, diagnosis, treatment, and medication. The purpose of the resulting medical records is manifold and ranges from archiving via accounting and billing to communication with external labs and hospitals. When a physician enters a medical record at his PC, he can browse medical terms by navigating along the SNOMED CT hierarchy and view the natural language descriptions when necessary. After selecting a relevant term,

the SNOMED CT numerical code can be automatically inserted into the record. In fact, there is a strong trend towards *electronic* medical records, and standard data formats such as the Health Level 7 Clinical Document Architecture (CDA) are already in wide-spread use. To make sure that medical records represented in such standard formats are not misinterpreted, it is important to use standardized medical terminology. To this effect, CDA is based on SNOMED CT numerical codes (among others). The general idea of SNOMED CT is that, if this standard is adopted by the healthcare system of a nation, then *all* medical data storage and exchange is in terms of SNOMED CT codes.

Why, then, is SNOMED CT a logical theory? We first note that other standardized medical terminologies such as ICD-10 are *not* based on logic. However, the designers of SNOMED CT find it useful to employ logic during the *design and maintenance* of their ontology. In fact, engineering a large terminology is far from trivial. To generate an ontology of the size of SNOMED CT that covers a range of specialized areas, many medical experts and IT experts have to work together and follow agreed-upon design patterns that guide design decisions. But even with a good design methodology, ensuring that the ontology is of consistent quality and free of mistakes is very difficult. It is here that logic comes into play: the designers of SNOMED CT use automated reasoning to verify their modelling and derive consequences that are only implicit in it. Traditionally, they concentrate on deciding implication in the description logic $\mathcal{EL}$; for example, a recently discovered mistake was the following entailed implication:

$$\text{SNOMED CT} \models \forall x\, (\mathsf{Amputation\_of\_arm}(x) \rightarrow \mathsf{Amputation\_of\_hand}(x)).$$

As we will discuss later, deciding entailment of implications is only the tip of the iceberg and there are many more opportunities for automated reasoning and logic techniques to support ontology design. Additional need for automated reasoning support is due to the fact that medical knowledge, national legislation, etc., are constantly changing, which requires frequent changes to the ontology. Finally, SNOMED CT is customized to national needs and translated into various (human) languages, which results in a large number of different, but closely related versions of the same ontology that need to be consistent with one another. From the perspective of mathematical logic, we thus have a huge and continuously developing first-order theory and a large number of slightly modified variants of this theory that are closely interrelated.

The design of large ontologies is not only difficult, but also time-consuming and expensive. For this reason and since IHSTDO is pleasantly liberal in giving out SNOMED CT for research purposes, SNOMED CT is increasingly being viewed as a valuable general-purpose tool and many novel applications are being proposed. Often, these novel applications involve logical reasoning and provide additional opportunities to apply techniques from logic. We mention only one such application as an example. Given that electronic medical records are gaining rapid popularity and that they use SNOMED CT codes for representing medical data, it is an intriguing idea to exploit not only the codes, but also the *logical definitions* when querying medical data, thus enabling more complete answers.

For example, when answering a query asking for patients with a liver disease, we may use SNOMED CT to deduce that hepatitis is a liver disease and thus include all patients suffering from hepatitis in the answer. In this way, query answering turns into logical deduction. From a logic perspective, an interesting new flavour of this application is scale: in addition to the already large SNOMED CT, the logical theory now also comprises a potentially huge amount of medical data (as ground facts), and new questions arise due to the special use of deduction in this application [23, 17].

3 Mathematical Logic

Today, it has become standard to apply automated reasoning and knowledge representation techniques during the design and deployment of ontologies. In particular, reasoning in languages such as $\mathcal{EL}$, $\mathcal{ALC}$, and OWL-DL has been investigated in depth and found to be decidable and PTIME-, EXPTIME-, and coNEXPTIME-complete, respectively [5, 6]. A large variety of automated reasoning systems are available and fruitfully employed by ontology engineers and users. In this section, we consider the potential role of mathematical logic, understood as the study of properties of logical theories and their interrelation. Specifically, we discuss how the notions of conservative extension and uniform interpolation can be employed for ontology engineering tasks. It is worth, though, to mention that these notions have previously been used in other areas of computer science. For example, uniform interpolation has been studied in AI under the name *forgetting* for more than a decade [27, 8] and modular program specification is another area of computer science where notions of conservativity and interpolation are of great interest [25, 21].

3.1 Conservative Extensions

As mentioned above, ontologies are not static objects, but are frequently corrected, customized, extended, and even merged with other ontologies. To understand and control the relationship between the resulting distinct versions of an ontology, one can directly employ notions from mathematical logic. The simplest such notion is probably that of a conservative extension. In mathematical logic, conservative extensions are used for relative consistency proofs and to decompose a theory into simpler subtheories. In ontology engineering, they can be used to give a rigorous definition of when the extension of an ontology by additional axioms interferes (in some possibly unintended way) with the original ontology, and to decompose an ontology into subtheories and modules [29]. To describe this application in more detail, recall that the notion of a conservative extension comes in two flavours. Let T and T' be two sets of axioms and $\mathcal{L}$ a logical language.

- *Model-theoretic*: $T \cup T'$ is a model-conservative extension of T if every model of T can be expanded to a model of T';

– *Language-dependent*: $T \cup T'$ is an $\mathcal{L}$-conservative extension of T if $T \cup T' \models \varphi$ implies $T \models \varphi$, for all $\mathcal{L}$-sentences φ with $\mathsf{sig}(\varphi) \subseteq \mathsf{sig}(T)$.

In mathematical logic, $\mathcal{L}$ is typically first-order logic (FO) or some language that extends FO. Every model-theoretic conservative extension is an FO-conservative extension, but the converse is well-known to fail. Both notions can, in principle, be used without modification to analyse the effect of adding axioms T' to a given ontology T. It tuns out, however, that the corresponding decision problem "decide whether $T \cup T'$ is a conservative extension of T" is undecidable for theories T and T' formulated in $\mathcal{EL}$, $\mathcal{ALC}$, and OWL-DL, both for model-conservative extensions and FO-conservative extensions [19, 18]. In the case of unfoldable theories, the situation changes: for $\mathcal{ALC}$ and OWL-DL, model and FO-conservativity are still undecidable whereas for $\mathcal{EL}$, model-conservativity is decidable in polynomial time [14].

The tractability result for unfoldable $\mathcal{EL}$-theories has been used to extract modules from SNOMED CT. The need for module extraction is due to the huge size of SNOMED CT and the fact that in many applications, only a small subset Σ of the 400.000 terms of SNOMED CT are required. In this case, it is useful to extract a minimal subset M of the set of SNOMED CT axioms such that Σ is included in the signature $\mathsf{sig}(M)$ of M and SNOMED CT is a conservative extension of M. The user can then work with M instead of SNOMED CT. By extending the algorithm that decides model-conservativity, one can extract such modules in polynomial time. A very encouraging experimental evaluation of this approach has been given in [14], see also the discussion of Figure 3.2 below. Interestingly, the modules extracted using such a logic-based approach are significantly smaller than modules extracted using heuristic approaches.

Under some natural conditions, model-conservativity becomes decidable even for non-unfoldable theories. For example, if T' and T share unary predicates only, then model-conservativity is coNEXPTIME$^{\text{NP}}$-complete for $\mathcal{ALC}$ [14]. Taken together, the stated negative and positive results show that a naive application of mathematical logic concepts to ontology engineering can fail because the associated algorithmic problems become unfeasible even for very weak fragments of FO, and thus the true challenge lies in adapting these notions to the needs of ontology engineers and users. Instead of resorting to unfoldable theories or unary predicates, there is another interesting direction to explore: the reason for undecidability of FO-conservativity in the case of non-unfoldable theories is due to the fact that we have considered *FO*-consequences, rather than consequences formulated at the same level of abstractness as ontologies. Indeed, an ontology engineer or user is typically not interested in arbitrary FO-consequences of an ontology, but in consequences relevant to her application. We now consider two notions of conservative extensions tailored towards ontology engineering.

Implication Conservativity. As illustrated by the "amputation of arm" example above, ontology engineers typically concentrate on implications formulated in the language $\mathcal{L}$ of the ontology. Therefore, it is natural to consider $\mathcal{L}$-conservativity with $\mathcal{L}$ the set of implications $\forall x(\varphi(x) \rightarrow \psi(x))$ formulated in $\mathcal{EL}$, $\mathcal{ALC}$, and OWL-DL, respectively. For simplicity, we simply speak of $\mathcal{EL}$-conservativity,

$\mathcal{ALC}$-conservativity, and OWL-conservativity. To analyze these syntactically defined notions of conservative extension, it is useful to first establish a model-theoretic characterization. Using techniques from modal logic, one can show that an $\mathcal{ALC}$-theory $T \cup T'$ is an $\mathcal{ALC}$-conservative extension of T if and only if for every model M of T, there exists a $\mathsf{sig}(T)$-bisimilar model M' of $T \cup T'$. Using this characterization, one can show that deciding $\mathcal{ALC}$-conservativity is decidable and 2ExpTime-complete [9]. A similar characterization, using simulations instead of bisimulations, can be used to characterize $\mathcal{EL}$-conservativity. In this case, the corresponding decision problem is ExpTime-complete [19]. Unfortunately, OWL-conservativity is undecidable [12]. We note, though, that there are extensions of $\mathcal{ALC}$ with guarded counting quantifiers for which conservativity is still decidable and 2ExpTime-complete [18].

Because of their high computational complexity, it remains to be seen in how far the decision problems associated with these notions of conservativity are useful in practice. Recall, however, that Snomed CT is unfoldable. To analyze $\mathcal{EL}$-conservativity for unfoldable $\mathcal{EL}$-theories, it is preferable to follow a proof-theoretic approach. Using the sequent calculus of [13], one can give a polynomial time algorithm that decides $\mathcal{EL}$-conservativity. The resulting algorithm has fruitfully been employed to ontology versioning [15].

Query Conservativity. At the end of Section 2, we have briefly discussed how Snomed CT can be used to query electronic medical records. In logic terms, one poses a query to a theory K that consists of an ontology T and a set A of ground facts that represent the data (called an *ABox* in description logic). Query languages of interest are *instance queries* of the form "output all constant symbols a such that $T \cup A \models P(a)$" and *conjunctive queries* in which P is a first-order formula built from conjunction and existential quantification. To compare theories K_1 and K_2, it is in principle possible to again apply standard notions of conservativity. However, ground facts change frequently and are typically unknown at the design time of the ontology. Thus, it is more useful to regard ground facts not as a part of the theory, but as an unknown "black-box". We say that $T \cup T'$ is a *query-conservative extension* of T if, and only if, for all sets A of ground facts and all queries q using symbols from T only, $T \cup T' \cup A \models q[a]$ iff $T \cup A \models q[a]$. The resulting notions of conservativity depend on the query language used and typically lie between implication conservativity and FO-conservativity. Again, model-theoretic and proof-theoretic methods can be applied to analyze it. For example, in [20] it is shown using model-theoretic methods that for $\mathcal{EL}$, query-conservativity (w.r.t. conjunctive queries) can be reduced to implication conservativity for an extension of $\mathcal{EL}$ with non-guarded existential quantification, and that the corresponding decision problem is decidable and ExpTime-complete.

3.2 Uniform Interpolation

As mentioned above, many applications of ontologies require only a rather small part of a large ontology, identified by a subvocabulary. Therefore, ontology users

are interested in generating small ontologies that "say the same" about the sub-vocabulary of interest as the original ontology. As discussed above, one possibility to obtain such an ontology is to extract a module, i.e., a subset of the original ontology of which the latter is a conservative extension. Another interesting way of generating such a small ontology is to compute a uniform interpolant. Let Σ be a signature, T a logical theory, and $\mathcal{L}$ a logical language. A finite set of $\mathcal{L}$-sentences T_Σ is called a *uniform interpolant* for T w.r.t. Σ and $\mathcal{L}$ if

- the signature $\mathsf{sig}(T_\Sigma)$ of T_Σ is contained in Σ;
- $T \models T_\Sigma$;
- for all $\varphi \in \mathcal{L}$: if $T \models \varphi$ and $\mathsf{sig}(\varphi) \cap T \subseteq \Sigma$, then $T_\Sigma \models \varphi$.

In other words, T_Σ provides an axiomatization of what T "says" about Σ in $\mathcal{L}$ without using symbols not in Σ. Apart from the motivation discussed above, there are various applications of uniform interpolants in ontology engineering. An example is ontology exploration: to avoid mistakes, an ontology engineer can identify a signature that captures a certain subject matter (such as amputations), generate a uniform interpolant that axiomatizes this subject matter, and then inspect it for problems. Another example is ontology re-use, where it may be more appropriate to import a uniform interpolant covering only the relevant terms than being forced to import additional terms as well.

For many logical languages, it is known that uniform interpolants do not always exist; this is true for FO, but also for various modal logics [10, 30]. Positive results are known for propositional intuitionistic logic [24] and the modal μ-calculus [7]. Thus, the applicability of uniform interpolants crucially depends on the ontology language used and the language $\mathcal{L}$ in which uniform interpolants are to be axiomatized. The following simple example shows that even for $\mathcal{EL}$, uniform interpolants do not always exist unless one admits second-order expressivity in $\mathcal{L}$: let $\Sigma = \{A, r\}$ and

$$T = \{\forall x\ (A(x) \rightarrow B(x)), \forall x\ (B(x) \rightarrow \exists y(r(x, y) \wedge B(y))\}.$$

The class of Σ-reducts of T-models coincides with the models satisfying

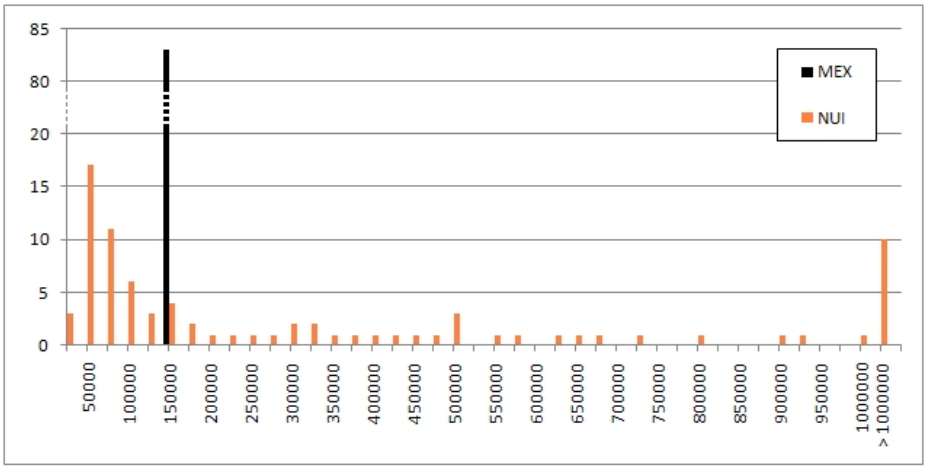
$$A(x) \rightarrow (\text{there exist } x_1, x_2, \ldots \text{ such that } r(x, x_1), r(x_1, x_2), \ldots)$$

and the first-order theory of this class of models is axiomatized by

$$\{A(x) \rightarrow \exists x_1 \cdots \exists x_n\ (r(x, x_1) \wedge \cdots \wedge r(x_{n-1}, x_n)) \mid n > 0\}$$

which is not finitely axiomatizable in FO. We are thus again faced with the problem that a naive application of tools from mathematical logic is only of limited use, and a more careful analysis of the situation is required. We explore three options.

First, there are useful cases in which uniform interpolants are guaranteed to exist: observe that, although rather simple, the theory T above is not unfoldable. Indeed, it turns out [15, 16] that for unfoldable $\mathcal{EL}$-theories, uniform interpolants w.r.t. $\mathcal{EL}$-implications always exist. These uniform interpolants can



be of exponential size in the worst case, and so their practical relevance can only be evaluated with the help of experiments. Figure 3.2, which is taken from [16], compares the size of minimal modules based on model-conservative extensions extracted from SNOMED CT using the prototype implementation MEX with the size of uniform interpolants computed using the prototype implementation NUI. For 83 randomly generated signatures consisting of 3000 unary and 20 binary predicates, it gives the number of modules and uniform interpolants (vertical axis) of a given size (horizontal axis), where the size is measured as as the number of symbols that occur in the module/interpolant. Thus, 10 uniform interpolants consist of more than 1 000 000 symbols. For all 83 signatures, the size of corresponding modules is between 125 000 and 150 000 symbols (which is about 3% of full SNOMED CT). Note that, in contrast to modules, the size of uniform interpolants depends a lot on the concrete signature. It is not clear yet, however, whether the very large uniform interpolants have a smaller axiomatization not found by the prototype implementation. Similar results hold for uniform interpolants formulated in languages $\mathcal{L}$ that correspond to query conservativity as discussed above. These first experiments suggest that uniform interpolants can become a useful tool for ontology engineering.

Second, even in setups where uniform interpolants are not guaranteed to exist, it might well be the case that for those ontologies that are actually used, uniform interpolants *do* usually exist. Then, it is of interest to develop algorithms that decide, given a signature Σ and an ontology T, whether a uniform interpolant exists. An example of such a result is the following: there do not always exist uniform interpolants for $\mathcal{ALC}$-theories (with uniform interpolants also formulated in $\mathcal{ALC}$), but it is decidable whether a uniform interpolant exists. Tight complexity bounds and experimental evaluation is still missing, though.

Third, one can move to a language $\mathcal{L}$ for the uniform interpolant that admits second-order expressivity. The modal μ-calculus has uniform interpolation, and

so a natural option is to consider as the language $\mathcal{L}$ extensions of $\mathcal{EL}$, $\mathcal{ALC}$, and OWL-DL with fixpoint operators.

4 Discussion

We have discussed how two notions introduced in mathematical logic, conservative extensions and uniform interpolation, can be applied to ontology engineering. Many other notions from mathematical logic remain to be explored. An example is *interpretations* between logical theories, which can be regarded as a generalization of conservative extensions. In many applications of ontologies such as data integration, mappings between the symbols of ontologies are of crucial importance. Investigating the relation between the mappings used by ontology engineers and theory interpretations as known from mathematical logic could be of great interest. Another research direction is abstract model theory for ontology languages: a main problem in the field of ontology languages is a lack of logic-based criteria to classify languages and understand their place within the landscape of all potential ontology languages. Currently, only concrete languages are investigated and compared, and methods from abstract model theory might well lead to a better understanding.

References

1. *Foundational Model Explorer*. http://fme.biostr.washington.edu:8089/FME/index.html.
2. http://www.ihtsdo.org/.
3. *KR-Med: Representing and sharing knowledge using SNOMED*, volume 410. CEUR Workshop Proceedings, 2008.
4. H. Andreka, J. Madarasz, and I. Nemeti. Logic of space-time and relativity theory. In *Handbook of Spatial logic*, pages 607–711. Springer, 2007.
5. F. Baader, S. Brandt, and C. Lutz. Pushing the $\mathcal{EL}$ envelope. In *Proc. IJCAI*. Morgan Kaufmann, 2005.
6. F. Baader, D. Calvanes, D. McGuiness, D. Nardi, and P. Patel-Schneider. *The Description Logic Handbook: Theory, implementation and applications*. Cambridge University Press, 2003.
7. G. D'Agostino and G. Lenzi. An axiomatization of bisimulation quantifiers via the μ-calculus. *Theoret. Comput. Sci.*, 338, 2005.
8. T. Eiter and K. Wang. Semantic forgetting in answer set programming. *Artif. Intell.*, 172(14):1644–1672, 2008.
9. S. Ghilardi, C. Lutz, and F. Wolter. Did I damage my ontology? A case for conservative extensions in description logic. In *Proc. KR*, pages 187–197, 2006.
10. S. Ghilardi and M. Zawadowski. Undefinability of propositional quantifiers in the modal system S4. *Studia Logica*, 55, 1995.
11. J. Golbeck, G. Fragaso, F. Hartel, J. Hendler, B. Parsia, and J. Oberhaler. The national cancer institute's thesaurus and ontology. *J. of Web Semantics*, 2003.
12. B. C. Grau, I. Horrocks, Y. Kazakov, and U. Sattler. Modular reuse of ontologies: Theory and practice. *Journal of Artificial Intelligence Research*, 31:273–318, 2008.
13. M. Hofmann. Proof-theoretic approach to description-logic. In *Proc. LICS*, pages 229–237, 2005.

14. B. Konev, C. Lutz, D. Walther, and F. Wolter. Semantic modularity and module extraction in description logic. In *Proc. ECAI*, pages 55–59, 2008.
15. B. Konev, D. Walther, and F. Wolter. The logical difference problem for description logic terminologies. In *Proc. IJCAR*, Lecture Notes in Computer Science, pages 259–274. Springer, 2008.
16. B. Konev, D. Walther, and F. Wolter. Forgetting and uniform interpolation in large-scale description logic terminologies. In *Proceedings of IJCAI*, 2009.
17. C. Lutz, D. Toman, and F. Wolter. Conjunctive query answering in el using a relational database system. In *Proceedings of IJCAI*, 2009.
18. C. Lutz, D. Walther, and F. Wolter. Conservative extensions in expressive description logics. In *Proc. IJCAI*, pages 453–458, 2007.
19. C. Lutz and F. Wolter. Conservative extensions in the lightweight description logic $\mathcal{EL}$. In *Proc. CADE*, Lecture Notes in Computer Science, pages 84–99. Springer, 2007.
20. C. Lutz and F. Wolter. Deciding inseparability and conservative extensios in the description logic EL. *Journal of Symbolic Computation*, 2009.
21. P. Mosses, editor. *CASL Reference Manual*. LNCS. Springer Verlag, 2004.
22. I. H. T. S. D. Organisation. *SNOMED CT User Guide*. 2008. Available from http://www.ihtsdo.org/snomed-ct/snomed-ct-publications/.
23. C. Patel, J. Cimino, J. Dolby, A. Fokoue, A. Kalyanpur, A. Kershenbaum, L. Ma, E. Schonburg, and K. Srinivas. Matching patient records to clinical trials using ontologies. In *Proceedings of International Semantic Web Conference*, volume 4825 of *Lecture Notes in Computer Science*, pages 816–829. Springer, 2007.
24. A. Pitts. On an interpretation of second-order quantification in first-order intuitionistic propositional logic. *J. Symbolic Logic*, 57, 1992.
25. J. G. R. Diaconescu and P. Stefaneas. Logical support for modularisation. In G. Huet and G. Plotkin, editors, *Logical Environments*, 1993.
26. A. L. Rector and J. Rogers. Ontological and practical issues in using a description logic to represent medical concept systems: Experience from galen. In *Reasoning Web*, pages 197–231, 2006.
27. R. Reiter and F. Lin. Forget it! In *Proceedings of AAAI Fall Symposium on Relevance*, pages 154–159, 1994.
28. K. Spackman. Managing clinical terminology hierarchies using algorithmic calculation of subsumption: Experience with SNOMED-RT. *J. of the Amer. Med. Informatics Ass.*, 2000. Fall Symposium Special Issue.
29. H. Stuckenschmidt, C. Parent, and S. Spaccapietra, editors. *Modular Ontologies*, volume 5445 of *Lecture Notes in Computer Science*. Springer, 2009.
30. A. Visser. Uniform interpolation and layered bisimulation. In *Gödel '96 (Brno, 1996)*, volume 6 of *Lecture Notes Logic*. Springer Verlag, 1996.