UI-FAME: A High-Performance Forgetting System for Creating Views of Ontologies

Xuan Wu
Nanjing University
Nanjing, China

Wenxing Deng
Beijing University of Technology
Beijing, China

Chang Lu
Nanjing University
Nanjing, China

Peiqi Wei
Beijing University of Chem. Tech.
Beijing, China

Yizheng Zhao*
Nanjing University
Nanjing, China

Hao Feng
Meituan-Dianping Group
Beijing, China

ABSTRACT

This paper describes a Java-based forgetting system UI-FAME for creating views of ontologies. In relational databases, a view is a subset of the database, while in ontologies, a view is more than a subset; it contains not only axioms that are contained in the original ontology, but also newly-derived axioms that are entailed by the original ontology. Forgetting is a form of non-standard reasoning that can be used to create views of ontologies by eliminating from the ontologies a set of concept and role names, namely the forgetting signature, while keeping all logical consequences over the names in the remaining signature.

We compared UI-FAME with publicly accessible forgetting systems, namely LETHE and an early prototypical version of UI-FAME, over the $\mathcal{ALC}$-TBox fragment of 494 ontologies taken from the Oxford Ontology Library. The results showed that UI-FAME had better success rates than LETHE and its prototypical version, and outsped LETHE by a large margin. UI-FAME has been integrated as back-end support in Babylon Health's (a leading digital healthcare company based in London) knowledge base (formed by the merger of healthcare ontologies from different sources) interfaces for their main concerns of ontology analysis and keeping track of the changes in different versions of each merged ontology.

1 INTRODUCTION

In the context of Information Science and Artificial Intelligence, an *ontology* defines a set of representational primitives with which to model a domain of discourse. These representational primitives include classes (*concepts*), class members (*individuals*), and properties (*attributes* of concepts and individuals and *relationships* to other concepts and individuals). The definitions of the representational primitives include information about their meanings and constraints on their logically consistent application. In the context of database systems, an ontology can be viewed as a level of abstraction of data models, analogous to hierarchical and relational models, but intended for modeling knowledge about concepts, individuals, and their properties. Ontologies, serving as a new data model for representation of domain knowledge, have been widely deployed across numerous industry sectors, including energy, laws, biology, medical, and healthcare sectors.

With the growing usage of ontologies in real-world application scenarios, not only has the number of available ontologies increased considerably, but also they are becoming large in size, complex in structure, and thus more difficult to manage. Moreover, capturing domain knowledge in the form of ontologies is labor-intensive work from the engineering perspective. There is therefore a strong demand from both academia and industry for techniques and automated tools for re-engineering with ontologies, so that existing ontologies can be reused to their full potential — new ontologies can be generated from existing ones, and are not necessarily scratch-developed, which is costly and error-prone.

Creating views of ontologies is one of the ontology engineering operations that seeks to generate new ontologies from existing ones. The *signature* (*vocabulary*) of an ontology $\mathcal{O}$, denoted by $\mathsf{sig}(\mathcal{O})$, is the set of the concept and role names in $\mathcal{O}$. A *view* $\mathcal{V}$ of $\mathcal{O}$ is a new ontology obtained from $\mathcal{O}$ using only part of $\mathcal{O}$'s signature, namely the *target signature*, while preserving the original meanings of the terms in the target signature. Creating views of ontologies is useful for many ontology-based knowledge management tasks such as ontology debugging and repair [14], merging and alignment [10], and related tasks such as information hiding [4], logical difference computation [9], and ontology-based query answering [11].

Forgetting is a form of (non-standard) reasoning concerned with eliminating from an ontology a set of concept and role names in its signature, namely the *forgetting signature*, in such a way that all logical consequences are preserved up to the remaining signature. Forgetting in this sense can be used as an effective means for the task of computing views of ontologies: the ontology obtained from forgetting, namely the *forgetting solution*, is a view of the original ontology for *the target signature*, which amounts to the remaining signature in forgetting.

Forgetting is an inherently difficult problem; it is much harder than standard reasoning (satisfiability testing), and very few logics are known to be complete for forgetting.[1] E Existing forgetting methods include LETHE [8], NUI [7], the method developed by [9], and UI-FAME [16]. In particular, LETHE is based on a generalization of *resolution* [2, 3]; it can eliminate concept and role names from $\mathcal{ALCH}$-TBoxes and can eliminate concept names from $\mathcal{SHQ}$-TBoxes. NUI and the one of [9] are based on resolution as well; it can eliminate concept and role names from TBoxes specified in $\mathcal{ELH}$, a light-weight DL less expressive than $\mathcal{ALC}$, and the method of [9] can eliminate concept names from $\mathcal{ALC}$-TBoxes. UI-FAME is a hybrid approach based on both resolution and a monotonicity property called Ackermann's Lemma [1], and it can eliminate concept and role names from $\mathcal{ALC}$-TBoxes. All these

*Yizheng Zhao is the corresponding author of this paper.

[1] When we say a logic $\mathcal{L}$ is complete for forgetting, we mean that for any forgetting problem specified in $\mathcal{L}$, there always exists a forgetting solution in $\mathcal{L}$.

methods have been implemented as standalone systems, among which, LETHE is the only publicly accessible one at present.[2]

In this paper we describe UI-FAME. The forgetting algorithm [16] used by UI-FAME is a refinement of the algorithm presented in [15]. We compared UI-FAME with publicly accessible forgetting systems, namely LETHE and an early prototypical version of UI-FAME, over the $\mathcal{ALC}$-TBox fragment of 494 ontologies taken from the Oxford Ontology Library. The comparison considered success rates, speed, memory consumption as the principal indicators to assess the performance of the systems. Our experimental results showed that UI-FAME had better success rates than LETHE and its prototypical version, and outsped LETHE by a large margin. UI-FAME is being used as a back-end support in Babylon Health's (a digital healthcare company based in London) knowledge base (formed by the merger of healthcare ontologies from different sources) interfaces for their main concerns of ontology analysis and keeping track of the changes in different versions of the merged ontologies.

2 FORGETTING FOR $\mathcal{ALC}$-TBOXES

We assume familiarity with the description logic $\mathcal{ALC}$. By $\mathsf{sig_C}(X)$ and $\mathsf{sig_R}(X)$ we denote respectively the sets of the concept names and role names occurring in X, where X ranges over concepts, axioms, and ontologies. We define $\mathsf{sig}(X) = \mathsf{sig_C}(X) \cup \mathsf{sig_R}(X)$.

Definition 1 (**Forgetting for $\mathcal{ALC}$-TBoxes**). *Let $\mathcal{O}$ be an $\mathcal{ALC}$-TBox and let $\mathcal{F} \subseteq \mathsf{sig}(\mathcal{O})$ be a set of concept and role names. An $\mathcal{ALC}$-TBox $\mathcal{V}$ is a* solution of deductively forgetting $\mathcal{F}$ from $\mathcal{O}$ *iff the following conditions hold: (i) $\mathsf{sig}(\mathcal{V}) \subseteq \mathsf{sig}(\mathcal{O})\backslash\mathcal{F}$, and (ii) for any axiom α with $\mathsf{sig}(\alpha) \subseteq \mathsf{sig}(\mathcal{O})\backslash\mathcal{F}$, $\mathcal{V} \models \alpha$ iff $\mathcal{O} \models \alpha$.*

Definition 1 says that $\mathcal{V}$ (the forgetting solution) has the same logical consequences with $\mathcal{O}$ (the original ontology) in the remaining signature $\mathsf{sig}(\mathcal{O})\backslash\mathcal{F}$. $\mathcal{F}$ is called the *forgetting signature*, i.e., the set of concept and role names to be eliminated. $\mathcal{V}$ can be regarded as a *view* of $\mathcal{O}$ w.r.t. the remaining signature $\mathsf{sig}(\mathcal{O})\backslash\mathcal{F}$ in the sense that it gives the same answers as $\mathcal{O}$ to the queries formulated using the names in $\mathsf{sig}(\mathcal{O})\backslash\mathcal{F}$. In relational databases, a view is a subset of the database, whereas in ontologies, a view is more than a subset; it contains not only axioms that are contained in the original ontology, but also newly derived axioms that are entailed by the original ontology (implicitly contained in the original ontology). Such axioms can be derived during the forgetting process. The remaining signature in forgetting corresponds to the *target signature* when considering the problem as computing views of ontologies.

3 IMPLEMENTATION OF UI-FAME

UI-FAME is implemented in Java using the OWL API,[3] which is a Java API for creating, parsing, manipulating, and serializing OWL ontologies, and is released under the open source licenses LGPL[4] and Apache[5]. UI-FAME employs the OWL API Version 3.4.7 for the tasks of loading, parsing, and saving OWL ontologies.

Figure 1 depicts the general design of UI-FAME. The input to UI-FAME are an $\mathcal{ALC}$-TBox $\mathcal{O}$, a set $\mathcal{F}_\mathsf{C} \subseteq \mathsf{sig_C}(\mathcal{O})$ of concept names to be forgotten, and a set of $\mathcal{F}_\mathsf{R} \subseteq \mathsf{sig_R}(\mathcal{O})$ of role names to be forgotten. $\mathcal{F}_\mathsf{C}$ and $\mathcal{F}_\mathsf{R}$ make up the forgetting signature $\mathcal{F}$. The input ontology must be given as a text file in XML, OWL, RDF, or TURTLE format, or a URL pointing to the file. UI-FAME takes only $\mathcal{ALC}$-axioms; axioms not expressible in $\mathcal{ALC}$ are omitted.

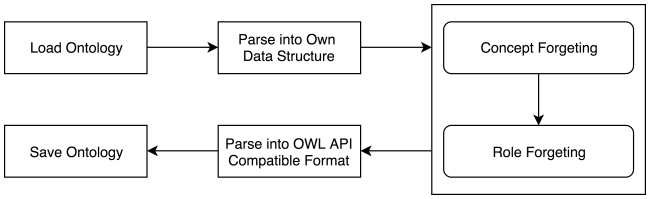

Figure 1: General Design of UI-FAME

UI-FAME computes a solution of forgetting $\mathcal{F}$ from $\mathcal{O}$ by eliminating single concept and role names in $\mathcal{F}$ using the forgetting algorithm presented in [16].[6] The algorithm is a refinement of the one presented in [15], which underpins the initial version of UI-FAME. The refined algorithm allows more kinds of forgetting problems to be solved. For details of the forgetting algorithm we refer the reader to a long version of this paper at https://github.com/anonymous-ai-researcher/uifame, where the source code and an executable .jar file of UI-FAME can also be found. Another way to try out UI-FAME is via the online platform http://www.forgettingshow.info/.

4 COMPARISON OF UI-FAME WITH EXISTING FORGETTING SYSTEMS

In order to find the best forgetting systems, we compared UI-FAME with LETHE, and with the original prototype of UI-FAME based on an unrefined version of the forgetting algorithm.[7] Here "best" means the system has better overall performance considering success rate, speed, memory consumption, etc.

4.1 Test Data

The ontologies used for the comparison were taken from the Oxford Ontology Repository (Oxford-ISG).[8] Oxford-ISG contained a large number of ontologies collected from multiple sources. In particular, Oxford-ISG contained 797 ontologies, and we took 494 of them with the number $|TBox|$ of TBox axioms in the ontology not exceeding 10000. We further split the entire corpus of 494 ontologies into three groups: Corpus I with $10 \leq |TBox| \leq 1000$, containing 356 ontologies, Corpus II with $1000 \leq |TBox| \leq 5000$, containing 108 ontologies, and Corpus III with $5000 \leq |TBox| \leq 10000$, containing 26 ontologies. This gave a clearer insight into how LETHE and UI-FAME performed forgetting for ontologies of different sizes. Table 1 shows statistical information about the selected ontologies, where $|N_C|$ and $|N_R|$ denote the average numbers of the concept names and role names in the ontologies.

[2] http://www.cs.man.ac.uk/ koopmanp/lethe/index.html
[3] http://owlcs.github.io/owlapi/
[4] https://www.gnu.org/licenses/lgpl-3.0.html
[5] https://www.apache.org/licenses/LICENSE-2.0

[6] The algorithm of [16] fully supports forgetting for $\mathcal{ALCO}$-ontologies, but UI-FAME implements only part of the algorithm that handles exactly $\mathcal{ALC}$-TBoxes.
[7] We did not take NUI into the comparison because NUI accommodates only $\mathcal{ELH}$-TBoxes, or the implementation of the forgetting method of [9] because the method can only eliminate concept names, and moreover, it is not publicly accessible
[8] https://www.cs.ox.ac.uk/isg/ontologies/

Table 1: Statistics of three Oxford-ISG snapshots

		Min	Max	Medium	Mean	90th Percentile
I	$\|N_C\|$	0	1582	86	191	545
	$\|N_R\|$	0	332	10	29	80
	$\|TBox\|$	0	990	162	262	658
II	$\|N_C\|$	200	5877	1665	1769	2801
	$\|N_R\|$	0	887	11	34	61
	$\|TBox\|$	1008	4976	2282	2416	3937
III	$\|N_C\|$	1162	9809	4042	5067	8758
	$\|N_R\|$	1	158	4	23	158
	$\|TBox\|$	5112	9783	7277	7195	9179

4.2 Settings

To fit for real-world application scenarios, the standard of forgetting success was set as: (1) forgetting all the terms in the forgetting signature $\mathcal{F}$; (2) without introducing any extra expressivity outside of $\mathcal{ALC}$ in the forgetting solutions; (3) finished in the given timeout; (4) finished in the given space limit. The experiments were run on a laptop with an Intel Core i7-9750H processor, 6 cores running at up to 2.60 GHz, and 16 GB of DDR4-1330 MHz memory. We limited the timeout to 20 minutes and heap space to 8GB.

Both UI-FAME and LETHE were tested over the three Oxford-ISG snapshots. For each snapshot, we considered forgetting respectively 10% and 30% of the terms from the signature of each ontology. With a randomly generated forgetting signature, the test was repeated three times for UI-FAME and LETHE.

4.3 Comparison Results & Analysis

Table 2: Results (Mem: memory consumption of successful cases, S-Rate: success rate, TO-Rate: timeout rate, MO-Rate: out of memory rate, Extra: extra expressivity)

			Duration(s)	Mem(MB)	S-Rate	TO-Rate	MO-Rate	Extra
Refined UI-FAME	0.1	I	6.2	43	89.7	2.9	0.0	7.4
		II	13.3	349	81.7	12.5	0.0	5.7
		III	3.6	306	81.5	16.9	0.0	1.5
	0.3	I	4.8	106	85.4	13.3	0.0	1.3
		II	19.3	301	57.5	42.5	0.0	0.0
		III	26.9	424	68.6	31.4	0.0	0.0
Original UI-FAME	0.1	I	7.0	37	87.2	2.8	4.6	5.4
		II	13.5	86	64.8	22.2	11.8	1.2
		III	17.5	537	73.1	19.2	6.4	1.3
	0.3	I	8.4	51	78.4	15.2	3.7	2.7
		II	37.2	107	43.2	52.5	3.4	0.9
		III	15.0	610	59.0	25.6	11.6	3.8
LETHE	0.1	I	9.7	199	85.4	7.1	0.0	7.3
		II	10.2	1101	66.7	31.8	0.0	1.5
		III	48.1	1881	65.4	25.6	0.0	5.1
	0.3	I	21.7	322	73.9	18.2	0.0	7.7
		II	41.4	868	51.2	47.5	0.0	1.2
		III	69.7	1083	50.0	46.2	0.0	5.0

The results are shown in Table 2, where "Duration" denotes the average time consumption of the successful cases, and "Extra" denotes the percentage of cases introducing extra expressivity outside of $\mathcal{ALC}$. Such expressivity is not desired to be in forgetting solutions; if not, then the forgetting fails.

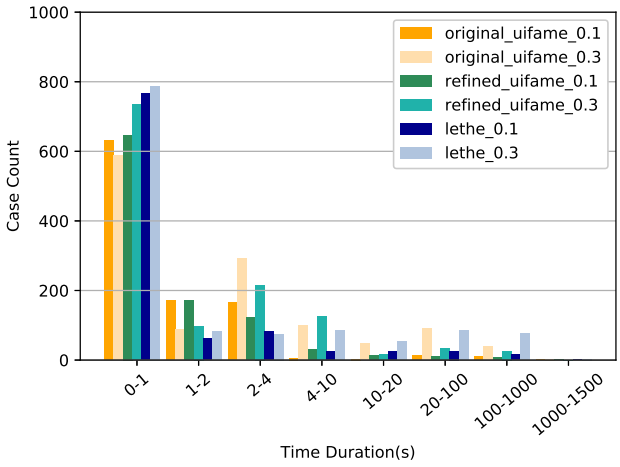

Figure 2: Statistical graph of samples in each time period

Compared to prototypical version of UI-FAME and LETHE, the refined UI-FAME had better success rates; see the "S-Rate" column. Most failures of the refined UI-FAME were due to the timeout (TO-Rate), and others due to extra expressivity being introduced in the resulting ontologies. Quite a few failures of the original UI-FAME were due to run-out of memory. The results also show that UI-FAME had lower memory consumption during the forgetting process; see the "Mem" column. In particular, LETHE's memory consumption was about four times of UI-FAME.

Figure 2 depicts the cach time distribution for all the successful cases, where it can be seen that most successful cases were finished within 1 second for both LETHE and UI-FAME.

5 APPLICATION SCENARIO

As one of the leading companies in the digital healthcare sector, Babylon Health[9], based in London, is seeking to make healthcare affordable and accessible for everyone through an AI doctor system, which relies on a huge medical knowledge base formed by merging and aligning ontologies from different sources. These medical ontologies are in a continuous state of evolving, being refined, being improved, and being adapted. Thus, one natural demand from Babylon Health would be to gain a better understanding of the differences between medical terms in different releases of ontologies.

Detecting the differences that occur in the meanings of the terms between two versions of an ontology can be done based on the notion of logical difference. The *logical difference* between two ontologies are the axioms entailed by one ontology but not by the other, reflecting the information gained and information lost between the ontologies [5, 6]. In the setting of $\mathcal{ALC}$, the notion of logical difference can be formalized as follows as a means of capturing the difference in the meaning of terms that is independent of the representation of ontologies.

Definition 2 (**Logical Difference for $\mathcal{ALC}$-TBoxes**). *Let O_1 and O_2 be $\mathcal{ALC}$-TBoxes. Let $\Sigma = sig(O_1) \cap sig(O_2)$ be the common signature of O_1 and O_2. The* logical difference *between*

[9] https://www.babylonhealth.com/

O_1 and O_2 is the set $\mathsf{UIDiff}(O_1, O_2)$ *of all $\mathcal{ALC}$-axioms α such that (i)* $\mathsf{sig}(\alpha) \subseteq \Sigma$*, (ii)* $O_2 \models \alpha$*, but (iii)* $O_1 \not\models \alpha$*. An axiom α that satisfies these conditions is a* witness *of a change in O_2 with respect to O_1.*

We notice that $\mathsf{UIDiff}(O_1, O_2)$ computes *information gain* from O_1 to O_2, and *information loss* from O_2 to O_1.

$\mathsf{UIDiff}(O_1, O_2)$ can however be computed using a forgetting-based approach: $\mathsf{UIDiff}(O_1, O_2) = \emptyset$ iff $O_1 \models \mathcal{V}_2$, where $\mathcal{V}_2$ is a view of O_2 for $\Sigma = \mathsf{sig}(O_1) \cap \mathsf{sig}(O_2)$. If $O_1 \not\models \mathcal{V}_2$, this means that $\mathsf{UIDiff}(O_1, O_2)$ is non-empty. Then every axiom $\alpha \in \mathcal{V}_2$ with $O_1 \not\models \alpha$ is a witness of $\mathsf{UIDiff}(O_1, O_2)$. From these considerations that $\mathsf{UIDiff}(O_1, O_2)$ can be computed by this two step algorithm:

Step (1): compute the view $\mathcal{V}_2$ of O_2 for $\Sigma = \mathsf{sig}(O_1) \cap \mathsf{sig}(O_2)$,
Step (2): collect the axioms $\alpha \in \mathcal{V}_2$ not entailed by O_1.

The first step can be done using a forgetting tool, and the second step can be done using a DL reasoner.

Computing a view $\mathcal{V}$ of an ontology O for a signature Σ amounts to capturing the information in O that involves only the terms in Σ. This is realized by *forgetting* from O the terms that do not belong to Σ [9]. The problem is therefore referred to as *forgetting* the terms in $\mathcal{F} = \mathsf{sig}(O) \backslash \Sigma$. In the case of computing $\mathsf{UI\text{-}Diff}(O_1, O_2)$, the forgetting signature is the set $\mathcal{F} = \mathsf{sig}(O_2) \backslash \mathsf{sig}(O_1)$.

UI-FAME has been integrated as back-end support into Babylon Health's knowledge base interface since May 2019 for their main concerns of ontology analysis and keeping track of the semantic difference in large-scale medical ontologies. The framework of the interface is shown in Figure 3.

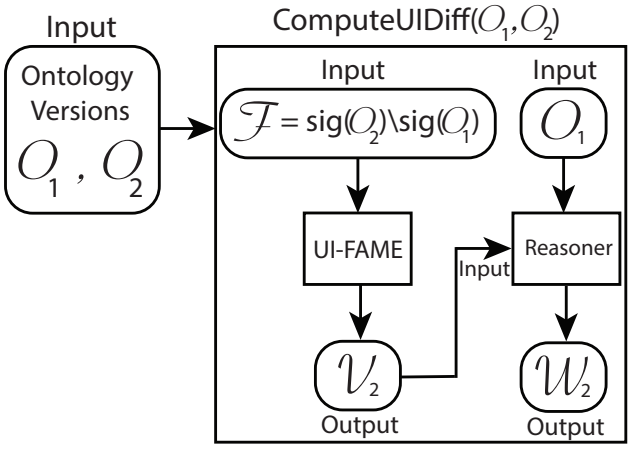

Figure 3: Framework of Babylon Health's Interface

The *input* to the interface are two $\mathcal{ALC}$-TBoxes O_1 and O_2 to be compared and a file path specifying the location where the output returned by the system is going to be saved. The input is order sensitive. Given O_1 as the first ontology and O_2 as the second one, the system computes a set $\mathsf{UI\text{-}Diff}(O_1, O_2)$ of UI-witnesses, which is the *information gain* from O_1 to O_2, and *information loss* from O_2 to O_1. If one wants to compute $\mathsf{UI\text{-}Diff}(O_2, O_1)$, O_2 must be specified as the first ontology, and O_1 as the second one. The *output* returned by the system is a set $\mathcal{W}_2$ of witnesses. The witness sets are saved as OWL/XML files which can be used for in-depth analysis or further processing.

SNOMED CT [13] is the most comprehensive, multilingual clinical healthcare terminology in the world, and is the kernel part of Babylon Health's knowledge base. One regular daunting task at Babylon Health is to identify the changes of the meaning of the terms between an International Edition and a subsequent release of SNOMED CT, and an International Edition and many of its country extensions. Previously this was done manually by ontology curators in the company. The task was fully automated since UI-FAME took over in May 2019. As of May 2020, UI-FAME has been successfully serving for a full year at Babylon Health, leading to several notable findings. For example, UI-FAME found that the Australia extension had many term definitions of the base edition changed in the extension; there were 42920 witnesses found in UI-Diff(Intl., Australia). In this case, the involvement of a domain expert may be useful to validate the changes.

The availability of advanced reasoning technologies like UI-FAME allows high level of automation in the process of creation and maintenance of Babylon's knowledge base and reduces the time spent on these tasks by 20%. This translates to direct saving for the company in staff cost. There is no direct translation to revenue as the knowledge base is not a product (it supports of client side products). The bigger gain comes via the competitive advantage via early adoption of the technology and faster route to market. Concerning market size, Babylon Health is currently working with NHS[10] to provide services to registered UK population of about 50m.

REFERENCES

[1] W. Ackermann. Untersuchungen über das Eliminationsproblem der mathematischen Logik. *Mathematische Annalen*, 110(1):390–413, 1935.
[2] L. Bachmair, H. Ganzinger, D. A. McAllester, and C. Lynch. Resolution theorem proving. In Robinson and Voronkov [12], pages 19–99.
[3] C. G. Fermüller, A. Leitsch, U. Hustadt, and T. Tammet. Resolution decision procedures. In Robinson and Voronkov [12], pages 1791–1849.
[4] B. C. Grau and B. Motik. Reasoning over ontologies with hidden content: The import-by-query approach. *J. Artif. Intell. Res.*, 45:197–255, 2012.
[5] B. Konev, M. Ludwig, D. Walther, and F. Wolter. The Logical Difference for the Lightweight Description Logic $\mathcal{EL}$. *J. Artif. Intell. Res.*, 44:633–708, 2012.
[6] B. Konev, D. Walther, and F. Wolter. The Logical Difference Problem for Description Logic Terminologies. In *IJCAR*, volume 5195 of *Lecture Notes in Computer Science*, pages 259–274. Springer, 2008.
[7] B. Konev, D. Walther, and F. Wolter. Forgetting and Uniform Interpolation in Large-Scale Description Logic Terminologies. In *Proc. IJCAI'09*, pages 830–835. IJCAI/AAAI Press, 2009.
[8] P. Koopmann. *Practical Uniform Interpolation for Expressive Description Logics*. PhD thesis, The University of Manchester, UK, 2015.
[9] M. Ludwig and B. Konev. Practical Uniform Interpolation and Forgetting for $\mathcal{ALC}$ TBoxes with Applications to Logical Difference. In *Proc. KR'14*. AAAI Press, 2014.
[10] N. F. Noy and M. A. Musen. PROMPT: Algorithm and Tool for Automated Ontology Merging and Alignment. In *Proc. AAAI/IAAI'00*, pages 450–455. AAAI Press/The MIT Press, 2000.
[11] M. Ortiz. Ontology based query answering: The story so far. In *Proc. AMW'13*, volume 1087 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2013.
[12] J. A. Robinson and A. Voronkov, editors. *Handbook of Automated Reasoning (in 2 volumes)*. Elsevier and MIT Press, 2001.
[13] K. A. Spackman. SNOMED RT and SNOMED CT. promise of an international clinical ontology. *M.D. Computing 17*, 2000.
[14] N. Troquard, R. Confalonieri, P. Galliani, R. Peñaloza, D. Porello, and O. Kutz. Repairing Ontologies via Axiom Weakening. In *Proc. AAAI'18*, pages 1981–1988. AAAI Press, 2018.
[15] Y. Zhao, G. Alghamdi, R. A. Schmidt, H. Feng, G. Stoilos, D. Juric, and M. Khodadadi. Tracking Logical Difference in Large-Scale Ontologies: A Forgetting-Based Approach. In *Proc. AAAI'19*, pages 3116–3124. AAAI Press, 2019.
[16] Y. Zhao, R. A. Schmidt, Y. Wang, X. Zhang, and H. Feng. A practical approach to forgetting in description logics with nominals. In *Proc. AAAI'20*, pages 3073–3079. AAAI Press, 2020.

[10] https://www.nhs.uk/