Efficient Computation of General Modules for $\mathcal{ALC}$ Ontologies (Extended Abstract)

Hui Yang[1], Patrick Koopmann[2], Yue Ma[1], Nicole Bidoit[1]

[1] LISN, CNRS, Université Paris-Saclay, Rue Raimond Castaing bâtiment 650, 91190 Gif-sur-Yvette, French
[2] Vrije Universiteit Amsterdam, De Boelelaan 1105, 1081 HV Amsterdam, The Netherlands

Keywords
Ontologies, General Module, Deductive Module, Uniform Interpolation

We present a method for extracting general modules for ontologies formulated in the description logic $\mathcal{ALC}$. Given an ontology $\mathcal{O}$ and a signature Σ of concept and role names, a *general module* is an ideally substantially smaller ontology that preserves all $\mathcal{ALC}$ axioms that are entailed by $\mathcal{O}$ and can be expressed using only the names in Σ. As such, it has applications such as ontology reuse and ontology analysis. In particular, if we want to reuse only a part of an ontology in a new application, rather than using the entire ontology, we may first want to extract a general module for the set of terms that are actually relevant for the application. General modules may also serve as a restricted view of the ontology, focused on a small set of terms of interest, which may make hidden relations between the terms visible. While *classical modules* have the additional requirement that they are a subset of the original ontology, general modules can also reformulate axioms from the input ontology, which can lead to smaller results that are more focused on the provided signature, and thus potentially better suited for the aforementioned applications. Another special case of general modules are *uniform interpolants*, which are general modules that are complete formulated using only names from the provided ontology. We believe that for application such as ontology reuse, this requirement is in fact too strict and can even be counter-productive. However, so far, general modules have only been investigated for lightweight description logics [1, 2]. Our main contributions are: 1) we present the first method dedicated to computing general modules in $\mathcal{ALC}$, 2) we provide a formal analysis of some properties of the general modules we compute, 3) based on our methods, we also obtain new methods for computing classical modules and uniform interpolants, and 4) using an evaluation on real-world ontologies we demonstrate the efficiency of our technique. This work has been accepted by IJCAI 2023. For detailed results and proofs, please refer to the extended version of the paper [3].

The main steps of our approach are shown in Figure 1. Essentially, our method works by performing uniform interpolation on a normalized version of the input ontology, inspired by the uniform interpolation method presented in [4]. Our normalization introduces fresh concept names, called *definers*, which are eliminated in the final step. However, different from [4], we put fewer constraints on the normal form and do not allow the introduction of

DL 2023: 36th International Workshop on Description Logics, September 2–4, 2023, Rhodes, Greece
yang@lisn.fr (H. Yang); p.k.koopmann@vu.nl (P. Koopmann); ma@lisn.fr (Y. Ma); nicole.bidoit@lisn.fr (N. Bidoit)

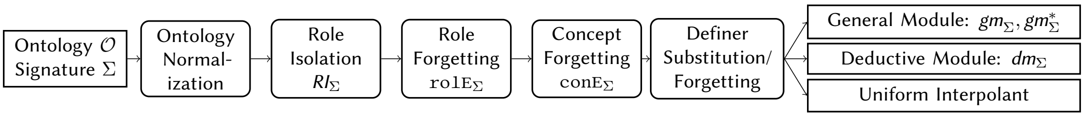

Figure 1: Overview of our unified method for computing general modules, deductive modules, and uniform interpolants

definers after normalization. As a result, our definer elimination step may reintroduce names eliminated during uniform interpolation. This is not a problem, since our aim is not to compute uniform interpolants of the input ontology. In contrast, eliminating definers as done in [4] can cause an exponential blowup, and introduce concepts with the non-standard *greatest fixpoint* constructor [5]. In the following, we give a short overview of our method.

Ontology Normalization An ontology $\mathcal{O}$ is in *normal form* if every axiom is of the following form:

$$\top \sqsubseteq L_1 \sqcup \ldots \sqcup L_n \qquad L_i ::= A \mid \neg A \mid \mathsf{Q}r.A, \;\; \mathsf{Q} \in \{\forall, \exists\}.$$

For simplicity, we omit the "$\top \sqsubseteq$" on the left-hand side of normalized axioms. As an example, the axiom $A_2 \sqsubseteq A_3 \sqcup \forall s.B_3$ is equivalent to $\neg A_2 \sqcup A_3 \sqcup \forall s.B_3$ in normal form.

As the first step, we normalize the input ontology $\mathcal{O}$ using standard transformations. In particular, we replace every concept C occurring under role restrictions by a so-called *definer* name D. For each definer D, we remember the concept C_D that was replaced by it.

Role Forgetting Next, we apply *role forgetting* to eliminate role names outside the given *signature* Σ. Existing methods to compute role forgetting either rely on an external reasoner [6, 4] or introduce the *universal role* ∇ [7, 8]. The former approach can be expensive, while the latter produces axioms outside of $\mathcal{ALC}$. Our normal form allows us to implement a more efficient solution within $\mathcal{ALC}$, which relies on an integrated reasoning procedure and an additional transformation step that produces so-called *role isolated ontologies* $RI_\Sigma(\mathcal{O})$. For role isolated ontologies, role forgetting is straightforward by the following result.

Theorem 1. *Let* $\mathtt{rolE}_\Sigma(\mathcal{O})$ *be the ontology obtained as follows:*

1. *apply the* r-Rule *in Figure 2 exhaustively for each* $r \in sig_\mathsf{R}(\mathcal{O}) \setminus \Sigma$,
2. *remove all axioms containing some* $r \in sig_\mathsf{R}(\mathcal{O}) \setminus \Sigma$.

If $\mathcal{O}$ *is role isolated for* Σ*, then* $\mathtt{rolE}_\Sigma(\mathcal{O})$ *is a role-forgetting for* $\mathcal{O}$ *and* Σ.

Concept Forgetting Inspired by [7, Theorem 1], we define a concept forgetting operator $\mathtt{conE}_\Sigma$ using *A-Rule* in Figure 2 in a similar way as defining $\mathtt{rolE}_\Sigma$. Applying the aforementioned procedure yields $\mathtt{conE}_\Sigma(\mathtt{rolE}_\Sigma(RI_\Sigma(\mathcal{O})))$, which contains only of names in the signature Σ or definers.

$$\underline{r\text{-}Rule}: \quad \frac{C_1 \sqcup \exists r.D_1,\ \bigcup_{j=2}^{n}\{C_j \sqcup \forall r.D_j\},\ K_D}{C_1 \sqcup \ldots \sqcup C_n}, \qquad K_D = \sqcup_{1\le i\le n}\neg D_i \text{ or } \sqcup_{2\le i\le n}\neg D_i$$

$$\underline{A\text{-}Rule}: \quad \frac{C_1 \sqcup A_1 \qquad \neg A_1 \sqcup C_2}{C_1 \sqcup C_2}$$

Figure 2: Inference rules used by our method.

Constructing the General Module To obtain our general modules $gm_\Sigma(\mathcal{O})$ for $\mathcal{O}$ and Σ, we eliminate the introduced definers D from $\texttt{conE}_\Sigma(\texttt{rolE}_\Sigma(RI_\Sigma(\mathcal{O})))$. For this, we replace each definer D by C_D, the concept replaced by D in the normalization step.

Eliminating definers in this way may reintroduce previously forgotten names, which is why our general modules are in general not uniform interpolants. This way, we avoid the triple exponential blow-up caused by uniform interpolation in the worst case [9]. However, a single exponential blow-up in the size of the input is still possible.

Proposition 1. *For any ontology $\mathcal{O}$ and signature Σ, we have $\|gm_\Sigma(\mathcal{O})\| \le 2^{O(\|cl(\mathcal{O})\|)}$. On the other hand, there exists a family of ontologies $\mathcal{O}_n$ and signatures Σ_n s.t. $\|\mathcal{O}_n\|$ is polynomial in $n \ge 1$ and $\|gm_{\Sigma_n}(\mathcal{O}_n)\| = n \cdot 2^{O(\|cl(\mathcal{O}_n)\|)}$.*

Optimized General Modules To obtain smaller general modules, we eliminate some definers before substituting them, using an operation inspired by [10]. The resulting *optimized general modules* are denoted by $gm^*_\Sigma(\mathcal{O})$.

Deductive Modules and Uniform Interpolants Our method can also be used to compute classical modules, which we do by tracing the inferences performed when computing the general module $gm^*_\Sigma(\mathcal{O})$. For applications that instead require uniform interpolants, such as logical difference [11], we change the definer elimination step, and eliminate definers using an existing uniform interpolation tool such as Lethe or Fame [4, 12].

Evaluation We evaluated our methods on 222 ontologies from the OWL Reasoner Evaluation (ORE) 2015 classification track [13], from which we removed axioms not expressible in $\mathcal{ALC}$. For each ontology, we randomly generated 50 signatures consisting of 100 concept and role names as in [8].

We implemented a prototype called GeMo[1] in Python 3.7.4. For each request $(\mathcal{O}, \Sigma)$, GeMo produced two types of general modules (gm and the optimized gm*), a classical module (dm), as well as a uniform interpolant (gmLethe). To show that our general modules can serve as a better alternative for ontology reuse and analysis, we compared them with the state-of-the-art tools implementing module extraction and uniform interpolation for $\mathcal{ALC}$: **(i)** $\top\bot^*$-*modules* [14] as implemented in the OWL API [15]; **(ii)** `minM` [8] that computes *minimal deductive modules* under

[1]The prototype can be downloaded at https://hub.docker.com/r/yh1997/demo_gemo

Table 1
Success rates. The first (resp. second) best results is highlighted in red (resp. blue).

$\top\bot^*$-module	minM	LETHE	FAME	GEMO	gmLethe
100%	84.34%	85.27%	91.25%	97.34%	96.17%

Table 2
Comparison of different methods (max. / avg. / med.).

Methods		Resulting ontology length	Time cost
minM		2,355 / 392.59 / 264	595.88 / 51.82 / 8.86
$\top\bot^*$-module		4,008 / 510.77 / 364	5.94 / 1.03 / 0.90
FAME		9,446,325 / 6,661.01 / 271	526.28 / 3.20 / 1.17
LETHE		131,886 / 609.30 / 196	598.20 / 49.21 / 13.57
GEMO	gm	179,999 / 2,335.05 / 195	17.50 / 2.44 / 1.63
	gm*	21,891 / 466.15 / 166	
	dm	2,789 / 366.36 / 249	
gmLethe		21,891 / 364.10 /162	513.15 / 3.08 / 1.68

$\mathcal{ALCH}^{\nabla}$-semantics; **(iii)** LETHE 0.6[2][4] and FAME 1.0[3] [12] that compute uniform interpolants. Some of results are shown below.

- *Success rate:* We say a method *succeeds* on a request if it outputs the expected results within 600s. Table 1 summarizes the success rate for the methods considered. After the $\top\bot^*$-modules, GEMO had the highest success rate.
- *Resulting ontology length and run time:* We used *ontology length*, which is the sum of the sizes of the axioms in the ontology, as metric. Table 2 shows the length and run time for the requests on which all methods were successful (78.45% of all requests). We observe that our optimization gm* was effective, and lead to almost the best length in median, only being improved by gmLethe, which however had longer run times. While our modules were generally smaller than what was computed by the state-of-the-art, run-times could almost compete with that of $\top\bot^*$-module-modules.

Conclusion The experiments validate the efficiency of our proposal and the quality of the computed general modules. In the future, we want to optimize the concept elimination step to obtain more concise general modules. Also, we would like to investigate how to generalize our ideas to more expressive description logics.

[2] https://lat.inf.tu-dresden.de/~koopmann/LETHE/
[3] http://www.cs.man.ac.uk/~schmidt/sf-fame/

References

[1] N. Nikitina, B. Glimm, Hitting the sweetspot: Economic rewriting of knowledge bases, in: P. Cudré-Mauroux, J. Heflin, E. Sirin, T. Tudorache, J. Euzenat, M. Hauswirth, J. X. Parreira, J. Hendler, G. Schreiber, A. Bernstein, E. Blomqvist (Eds.), The Semantic Web - ISWC 2012 - 11th International Semantic Web Conference, Proceedings, Part I, volume 7649 of *Lecture Notes in Computer Science*, Springer, 2012, pp. 394–409. URL: https://doi.org/10.1007/978-3-642-35176-1_25. doi:10.1007/978-3-642-35176-1_25.

[2] G. Alghamdi, R. A. Schmidt, W. Del-Pinto, Y. Gao, Upwardly abstracted definition-based subontologies, in: A. L. Gentile, R. Gonçalves (Eds.), K-CAP '21: Knowledge Capture Conference, ACM, 2021, pp. 209–216. URL: https://doi.org/10.1145/3460210.3493564. doi:10.1145/3460210.3493564.

[3] H. Yang, P. Koopmann, Y. Ma, N. Bidoit, Efficient computation of general modules for $\mathcal{ALC}$ ontologies (extended version), 2023. arXiv:2305.09503, https://arxiv.org/abs/2305.09503.

[4] P. Koopmann, LETHE: Forgetting and uniform interpolation for expressive description logics, KI—Künstliche Intelligenz 34 (2020) 381–387.

[5] D. Calvanese, G. De Giacomo, Expressive description logics, in: The description logic handbook: theory, implementation, and applications, 2003, pp. 178–218.

[6] Y. Zhao, G. Alghamdi, R. A. Schmidt, H. Feng, G. Stoilos, D. Juric, M. Khodadadi, Tracking logical difference in large-scale ontologies: A forgetting-based approach, in: The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, AAAI Press, 2019, pp. 3116–3124. URL: https://doi.org/10.1609/aaai.v33i01.33013116. doi:10.1609/aaai.v33i01.33013116.

[7] Y. Zhao, R. A. Schmidt, Role forgetting for $\mathcal{ALCOQH}(\nabla)$-ontologies using an Ackermann-based approach, in: C. Sierra (Ed.), Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, ijcai.org, 2017, pp. 1354–1361. URL: https://doi.org/10.24963/ijcai.2017/188. doi:10.24963/ijcai.2017/188.

[8] P. Koopmann, J. Chen, Deductive module extraction for expressive description logics, in: C. Bessiere (Ed.), Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, 2020, pp. 1636–1643. URL: https://doi.org/10.24963/ijcai.2020/227. doi:10.24963/ijcai.2020/227.

[9] C. Lutz, F. Wolter, Foundations for uniform interpolation and forgetting in expressive description logics, in: T. Walsh (Ed.), IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence, IJCAI/AAAI, 2011, pp. 989–995. URL: https://doi.org/10.5591/978-1-57735-516-8/IJCAI11-170. doi:10.5591/978-1-57735-516-8/IJCAI11-170.

[10] M. Sakr, R. A. Schmidt, Fine-grained forgetting for the description logic $\mathcal{ALC}$, in: O. Arieli, M. Homola, J. C. Jung, M. Mugnier (Eds.), Proceedings of the 35th International Workshop on Description Logics (DL 2022), volume 3263 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2022. URL: http://ceur-ws.org/Vol-3263/paper-17.pdf.

[11] M. Ludwig, B. Konev, Practical uniform interpolation and forgetting for $\mathcal{ALC}$ tboxes with applications to logical difference, in: C. Baral, G. D. Giacomo, T. Eiter (Eds.), Principles of Knowledge Representation and Reasoning: Proceedings of the Fourteenth International

Conference, KR 2014, AAAI Press, 2014. URL: http://www.aaai.org/ocs/index.php/KR/KR14/paper/view/7985.
[12] Y. Zhao, R. A. Schmidt, FAME: an automated tool for semantic forgetting in expressive description logics, in: International Joint Conference on Automated Reasoning, Springer, 2018, pp. 19–27.
[13] B. Parsia, N. Matentzoglu, R. S. Gonçalves, B. Glimm, A. Steigmiller, The OWL reasoner evaluation (ORE) 2015 competition report, J. Autom. Reason. 59 (2017) 455–482. URL: https://doi.org/10.1007/s10817-017-9406-8. doi:10.1007/s10817-017-9406-8.
[14] B. C. Grau, I. Horrocks, Y. Kazakov, U. Sattler, Modular reuse of ontologies: Theory and practice, Journal of Artificial Intelligence Research 31 (2008) 273–318.
[15] M. Horridge, S. Bechhofer, The OWL API: a java API for OWL ontologies, Semantic Web 2 (2011) 11–21. URL: https://doi.org/10.3233/SW-2011-0025. doi:10.3233/SW-2011-0025.