LETHE: Saturation-Based Reasoning for Non-Standard Reasoning Tasks

Patrick Koopmann, Renate A. Schmidt

The University of Manchester, UK
{koopmanp, schmidt}@cs.man.ac.uk

Abstract. We present the saturation-based reasoning system LETHE. LETHE is a tool that can be used for uniform interpolation, forgetting, TBox abduction and logical difference. To solve these problems, LETHE uses saturation-based reasoning to eliminate certain symbols from an ontology, such that entailments in the remaining vocabulary are preserved. This is known as forgetting or uniform interpolation. LETHE is an implementation of our forgetting methods for various expressive description logics, and can be used as a Java library and as a standalone tool for the mentioned reasoning tasks. We give a high level description of the calculi used by LETHE, describe the reasoning algorithm implemented in LETHE, and give an evaluation of the system on realistic ontologies.

1 Introduction

LETHE is a tool for non-standard reasoning tasks that provides functionality for forgetting, TBox abduction and logical difference, which have applications in ontology analysis, debugging and ontology change. These functionalities are implemented by saturation-based reasoning methods for forgetting. The name *Lethe* is taken from Greek mythology, and denotes the river of forgetfulness. Forgetting, also known as uniform interpolation, is the core functionality of LETHE. It reduces the vocabulary of an ontology in such a way that entailments in the restricted vocabulary are preserved. We give a formal definition. Let $sig(E)$ denote the concept and role symbols occurring in an ontology or axioms E.

Definition 1. *The result of forgetting a set of symbols Σ from an ontology $\mathcal{O}$ in a logic $\mathcal{L}$ is an ontology $\mathcal{O}^{-\Sigma}$ such that $sig(\mathcal{O}^{-\Sigma}) = sig(\mathcal{O}) \setminus \Sigma$ and for all $\mathcal{L}$ axioms α with $sig(\alpha) \cap \Sigma = \emptyset$ we have $\mathcal{O}^{-\Sigma} \models \alpha$ iff $\mathcal{O} \models \alpha$.*

Apart from TBox abduction and computing logical differences between ontology versions, forgetting has a lot of direct applications, and can be used to hide confidential information from an ontology, to analyse hidden relations between concepts and roles, to generate ontology summaries, and has many more applications, examples of which can found in [2, 13, 14].

Saturation-based reasoning has received increased interest in the last years by the description logic community due to its good performance for classification. Examples include the consequence-based reasoners used by ELK [3] and ConDOR [19, 20]. Saturation procedures work by adding new inferences performed

by a set of rules to the knowledge until no new inferences can be performed. This technique is especially useful if the aim is to compute all entailments of a specified form, such as atomic concept inclusions in the case of classification. For the same reason, saturation-based reasoning is well-suited for forgetting. Forgetting a concept or role symbol x is performed in LETHE by computing all entailments on that symbol, until axioms containing x can be removed. This can be done using goal-oriented saturation, provided that the underlying calculus enjoys the necessary completeness properties.

Common consequence-based reasoning methods do not enjoy these properties, since they are optimised towards classification. As example, take the following two axioms:

$$A_1 \sqsubseteq \forall r.B \qquad A_2 \sqsubseteq \forall r.\neg B$$

Forgetting B from these axioms should result in the single axiom $A_1 \sqcap A_2 \sqsubseteq \forall r.\bot$. However, this inference is not necessary if we are only interested in entailments of the form $A \sqsubseteq B$. Inferences between universal restrictions are usually not required for classification.

We developed a new family of saturation-based calculi for $\mathcal{ALC}$, $\mathcal{ALCH}$, $\mathcal{SHQ}$ and $\mathcal{SIF}$ ontologies [6, 5, 8, 10], as well as for $\mathcal{ALC}$ and $\mathcal{SHI}$ knowledge bases [9, 12, 11], that are *interpolation complete*. Interpolation completeness ensures that all inferences needed for forgetting are performed. LETHE implements the calculi for $\mathcal{ALCH}$ and $\mathcal{SHQ}$ ontologies, as well as for $\mathcal{ALC}$ knowledge bases.

Furthermore, LETHE implements optimised variations of the forgetting procedure to solve the related problems logical difference and TBox abduction. The aim of logical difference is to compute differing entailments between ontology versions with respect to some user-defined signature [4]. LETHE computes logical differences between two ontologies $\mathcal{O}_1$ and $\mathcal{O}_2$ by forgetting symbols from $\mathcal{O}_2$ that are not in the desired signature, and returning all resulting axioms that are not entailed by $\mathcal{O}_1$ (the same approach followed in [13]), and has some dedicated optimisations to make this feasible for large ontologies.

The form of abduction that is implemented in LETHE takes as input an ontology $\mathcal{O}$, a set of axioms Obs and a signature of abducibles Σ, and computes a logically weakest set of axioms H such that $sig(H) \subseteq \Sigma$ and $\mathcal{O} \cup H \models Obs$. TBox abduction can support ontology engineers in finding missing axioms of an ontology if expected entailments are not satisfied. This problem can be reduced to a form of forgetting by observing that $\mathcal{O} \cup H \models Obs$ iff $\mathcal{O} \cup \{\neg Obs\} \models \neg H$. The negation of a TBox axiom $C \sqsubseteq D$ can be encoded in the TBox axiom $\top \sqsubseteq \exists r^*.(C_i \sqcap \neg D_i)$, where r^* is fresh. Using the technique described in [7], we forget all symbols from $\neg Obs$ that are not in Σ , where $\mathcal{O}$ is used as background knowledge. If we negate the result again, we obtain the weakest set of axioms in Σ such that $\mathcal{O} \cup H \models Obs$.

2 The Calculi

Even though LETHE implements different calculi, these use similar techniques, which allows to implement a unified reasoning strategy. To give an idea, we

Resolution Rule:

$$\frac{C_1 \sqcup A \qquad C_2 \sqcup \neg A}{C_1 \sqcup C_2}$$

$\exists$-Elimination Rule:

$$\frac{C \sqcup \exists r.D \qquad \neg D}{C}$$

$\forall\mathsf{Q}$-Combination Rule:

$$\frac{C_1 \sqcup \forall R.D_1 \qquad C_2 \sqcup \mathsf{Q}S.D_2}{C_1 \sqcup C_2 \sqcup \mathsf{Q}S.D_{12}}$$

where $\mathsf{Q} \in \{\forall, \exists\}$ and D_{12} is a possibly new definer representing $D_1 \sqcap D_2$.

Fig. 1. Rules of the calculus $Res_{\mathcal{ALC}}$.

describe a simple calculus for $\mathcal{ALC}$, introduced in [6], that shares main ideas with these calculi.

The method requires the input to be normalised. Let N_c and N_r be the sets of *concept symbols* and *role symbols* that may be used in an ontology, and $N_d \subset N_c$ be a set of designated concept symbols called *definers*.

Definition 2. *An $\mathcal{ALC}$* literal *is a concept of the form A, $\neg A$, $\exists r.D$, $\forall r.D$, where $D \in N_d$. An $\mathcal{ALC}$ literal of the form $\neg D$, $D \in N_d$, is called* negative definer literal. *An $\mathcal{ALC}$* clause *is an axiom of the form $\top \sqsubseteq L_1 \sqcup \ldots \sqcup L_n$, where every L_i is an $\mathcal{ALC}$ literal. We usually omit the leading '$\top \sqsubseteq$' and assume that clauses are represented as sets, that is, they do not have duplicate literals and the order of the literals is not important. An ontology is in $\mathcal{ALC}$* normal form *iff every axiom is an $\mathcal{ALC}$ clause.*

Every $\mathcal{ALC}$ ontology can be transformed into $\mathcal{ALC}$ normal form using standard structural transformation and CNF transformation techniques. The resulting ontology shares all entailments modulo definers with the original ontology. If a normalised $\mathcal{ALC}$ ontology contains only clauses with at most one negative definer literal, similar transformations can be applied in the other direction in order to obtain an ontology without definers, which may use greatest fixpoint operators (see [6] for details).

Example 1. For the ontology given in the introduction, a normalised representation is $\mathcal{N} = \{\neg A_1 \sqcup \forall r.D_1, \neg D_1 \sqcup B, \neg A_2 \sqcup \forall r.D_2, \neg D_2 \sqcup \neg B\}$.

The calculus $Res_{\mathcal{ALC}}$ is shown in Figure 1. A major difference between our calculi and calculi for consequence-based reasoning [3, 19], or methods for direct resolution in modal logics [1, 15–17], is that symbols are not only introduced as part of the normalisation, but are also introduced dynamically during the saturation process. This allows to preserve the normal form without omitting required inferences. Specifically, the conclusions of the $\forall\mathsf{Q}$-combination rule use a definer D_{12} representing $D_1 \sqcap D_2$. Such a definer is introduced by adding the

two clauses $\neg D_{12} \sqcup D_1$ and $\neg D_{12} \sqcup D_2$ to the current clause set. New definers are only introduced if no corresponding definer already exists. This way, the number of introduced definers can be limited by 2^n, where n is the number of definers in the input ontology. As a result, we obtain that any saturated set of clauses is finite and contains at most 2^{2^n} clauses.

The calculi for $\mathcal{ALCH}$ ontologies, $\mathcal{SHQ}$ ontologies and $\mathcal{ALC}$ knowledge bases, as implemented in LETHE, use different rule sets, but the structure of these calculi is similar. They always have a resolution rule that allows to infer inferences on a specific symbol, and they have a set of combination rules, which lead to the introduction of new definers, and subsequently make further inferences possible.

In order to forget a concept symbol from the ontology, all inferences of the resolution rule for which the conclusion contains maximally one negative definer literal have to be performed. This may involve several applications of the combination rules, as these lead to the introduction of new definers and clauses. In the resulting clause set, all definers can be eliminated by applying the normalisation backwards, resulting in an ontology without definer symbols, which may use fixpoint operators. As described in [6], we can obtain a representation without fixpoints by leaving some definer symbols in the ontology or use the definer symbols as a guide to approximate the result. Both methods are implemented in LETHE.

3 Central Reasoning Algorithm Used in LETHE

The main challenge of implementing the calculi is to determine which applications of the combination rules are required in order to compute the forgetting result. If combination rules are applied unrestricted, we infer more clauses than necessary, and the implementation becomes impractical already for small ontologies. On the other hand, applications of the combination rules are necessary to make further inferences possible. LETHE uses an easy technique to ensure that all required inferences of combination rules are applied, while keeping the number of unnecessary inferences low. The idea is to use the resolution rule as guide to determine which definers should be introduced if possible. We can then determine how this definer can be introduced by using the combination rules.

Symbols to be forgotten are processed one after another, starting with the symbols with less occurrences in the ontology. For each symbol, we process only the portion of the ontology that uses this symbol. The main inference loop for forgetting a concept symbol A is then controlled by the following methods:

MAIN_LOOP. Process each negative occurrence of A one after another and perform all possible resolution steps. Pass the results to the method PROCESS_CLAUSE.

PROCESS_CLAUSE(C). Determine whether the clause C contains more than one negative definer literal. If not, pass it to ADD_CLAUSE. Otherwise, we have $C = \neg D_1 \sqcup \neg D_2 \sqcup C'$. If there is a definer D_{12} that represents $D_1 \sqcup D_2$, replace C by $\neg D_{12} \sqcup C'$ and pass it again to PROCESS_CLAUSE. (This step basically performs resolution on the clauses $\neg D_{12} \sqcup D_1$ and $\neg D_{12} \sqcup D_2$, which

are then also in the current clause set). If there is no such definer yet, look for occurrences of D_1 and D_2 under a role restriction and apply the corresponding role combination rules if possible, which may then introduce the definer D^{12}. Every clause that is inferred in this process, as well as the clause $\neg D_{12} \sqcup C'$ if it could be generated, is sent again to PROCESS_CLAUSE, since it may again contain more than one negative definer symbol.

ADD_CLAUSE(C). We apply simplification rules on the clause and check whether it is redundant. If the clause does not contain the symbol to be eliminated, we add it to the final output clause set.

Example 2. Assume we want to forget A from the clauses in the last example. MAIN_LOOP resolves on the clauses $\neg D_1 \sqcup A$ and $\neg D_2 \sqcup \neg A$, and passes $\neg D_1 \sqcup \neg D_2$ to PROCESS_CLAUSE. As this clause contains more than one negative definer literal, we do not have to add it to the result. Instead, we check whether we can apply combination rules on clauses in which D_1 and D_2 occur under a role restriction. This is possible on the clauses $\neg A_1 \sqcup \forall r.D_1$, $\neg A_2 \sqcup \forall r.D_2$, and we pass the resulting clauses $\neg A_1 \sqcup \neg A_2 \sqcup \forall r.D_{12}$, $\neg D_{12} \sqcup D_1$, $\neg D_{12} \sqcup D_2$ and $\neg D_{12}$ to the method ADD_CLAUSE. The final output contains only clauses with at most one negative definer literal, and no occurrences of B. The clause $\neg D_{12}$ corresponds to the axiom $D_{12} \sqsubseteq \bot$. We can therefore replace D_{12} by $\bot$. After some further syntactic adjustments we obtain the result $A_1 \sqcap A_2 \sqsubseteq \forall r.\bot$.

Several hash maps are used in the implementation to allow fast access of all occurrences of definers and the symbols that are currently forgotten. Throughout the computation, we ensure the invariant property that every stored clause contains maximally one negative definer literal. This is necessary to be able to eliminate introduced definer symbols in the end.

4 Evaluation

We evaluated the current version of LETHE on a set of 339 ontologies taken from the NCBO BioPortal repository [18], restricted to the different fragments supported by the different calculi. The ontologies correspond to the consistent ontologies of that repository that have less than 100,000 axioms. The average number of axioms of these ontologies is 4,760, and the 90th percentile is 13,045 axioms. For each ontology and each supported description logic, we generated 360 sets of symbols of size 50 and of size 100 and eliminated the symbols using LETHE. A timeout was set to 30 minutes. Table 1 shows the results of this evaluation. The tables show the success rate for each experiment, the fraction of results that could be represented without fixpoints, and the mean, median and 90th percentile of the duration of the computation. Note that for $\mathcal{SHQ}$ ontologies, LETHE only supports elimination of concept symbols.

In around 90–95% percent of cases, forgetting could be performed within 30 minutes. In most cases, the computation took just a few seconds, with the median of the duration being below 7 seconds in all cases. Except for the $\mathcal{ALC}$ forgetter with ABox support, the 90th percentile of the duration was always below 22 seconds.

$\mathcal{ALCH}$, forget 50 symbols	
Success Rate:	91.10%
Without Fixpoints:	95.29%
Duration Mean:	7.68 sec.
Duration Median:	2.74 sec.
Duration 90th percentile:	12.45 sec.

$\mathcal{ALCH}$, forget 100 symbols	
Success Rate:	88.10%
Without Fixpoints:	93.27%
Duration Mean:	18.03 sec.
Duration Median:	3.81 sec.
Duration 90th percentile:	21.17 sec.

$\mathcal{ALC}$ w. ABoxes, forget 50 symbols	
Success Rate:	94.79%
Without Fixpoints:	92.91%
Duration Mean:	23.94 sec.
Duration Median:	3.01 sec.
Duration 90th percentile:	29.00 sec.

$\mathcal{ALC}$ w. ABoxes, forget 100 symbols	
Success Rate:	91.37%
Fixpoints:	92.48%
Duration Mean:	57.87 sec.
Duration Median:	6.43 sec.
Duration 90th percentile:	99.26 sec.

$\mathcal{SHQ}$, forget 50 concept symbols	
Success Rate:	95.83%
Without Fixpoints:	93.40%
Duration Mean:	7.62 sec.
Duration Median:	1.04 sec.
Duration 90th percentile:	4.89 sec.

$\mathcal{SHQ}$, forget 100 concept symbols	
Success Rate:	90.77%
Fixpoints:	91.99%
Duration Mean:	13.51 sec.
Duration Median:	1.60 sec.
Duration 90th percentile:	11.65 sec.

Table 1. Forgetting 50 and 100 symbols using LETHE.

5 Outlook

LETHE currently supports forgetting of concept and role symbols from $\mathcal{ALCH}$ ontologies and $\mathcal{ALC}$ knowledge bases with ABoxes, as well as forgetting concept symbols from $\mathcal{SHQ}$ ontologies. Apart from providing functionality for forgetting, LETHE uses these forgetting procedures together with dedicated optimisations for TBox abduction and computing logical differences. The current version can be downloaded at `http://www.cs.man.ac.uk/~koopmanp/lethe`. It can be used as command line tool and as Java library. Furthermore, a simple GUI for computing and showing forgetting results on smaller ontologies is provided.

We currently have a prototypical implementation for $\mathcal{SHI}$ knowledge bases with ABoxes, which still has to be thoroughly tested and debugged. Even though our implementation of TBox abduction and logical differences has been thoroughly optimised, a proper evaluation on realistic use cases is still open.

A natural next step is to develop and implement a method for $\mathcal{SHIQ}$ knowledge bases, which would generalise all three currently implemented methods. Though currently only used for forgetting, the calculi can in theory also be used for classical reasoning tasks such as satisfiability checking, classification or realisation. An interesting open question is how LETHE would perform as a reasoner if it is implemented towards these reasoning tasks. Another open question is whether our saturation-based reasoning approach can be used for other non-classical problems, such as for example approximation or ABox abduction.

References

1. Enjalbert, P., Fariñas del Cerro, L.: Modal resolution in clausal form. Theoretical Computer Science 65(1), 1–33 (1989)
2. Gabbay, D.M., Schmidt, R.A., Szałas, A.: Second Order Quantifier Elimination: Foundations, Computational Aspects and Applications, Studies in Logic, vol. 12. College Publications (2008)
3. Kazakov, Y.: Consequence-Driven Reasoning for Horn $\mathcal{SHIQ}$ Ontologies. In: Boutilier, C. (ed.) Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-09). pp. 2040–2045. AAAI Press (2009)
4. Konev, B., Walther, D., Wolter, F.: The logical difference problem for description logic terminologies. In: Automated Reasoning, Lecture Notes of Computer Science, vol. 5195, pp. 259–274. Springer (2008)
5. Koopmann, P., Schmidt, R.A.: Forgetting concept and role symbols in $\mathcal{ALCH}$-ontologies. In: Logic for Programming, Artificial Intelligence and Reasoning. Lecture Notes in Computer Science, vol. 8312, pp. 552–567. Springer (2013)
6. Koopmann, P., Schmidt, R.A.: Uniform interpolation of $\mathcal{ALC}$-ontologies using fixpoints. In: Frontiers of Combining Systems. Lecture Notes in Computer Science, vol. 8152, pp. 87–102. Springer (2013)
7. Koopmann, P., Schmidt, R.A.: Computing uniform interpolants of $\mathcal{ALCH}$-ontologies with background knowledge. In: Proc. ARW-DT'14 (2014)
8. Koopmann, P., Schmidt, R.A.: Count and forget: Uniform interpolation of $\mathcal{SHQ}$-ontologies. In: Automated Reasoning. Lecture Notes in Computer Science, vol. 8562, pp. 434–448. Springer (2014)
9. Koopmann, P., Schmidt, R.A.: Forgetting and uniform interpolation for $\mathcal{ALC}$-ontologies with ABoxes. In: Proceedings of the 27th International Workshop of Description Logics (DL 2014). CEUR Workshop Proceedings, vol. 1193, pp. 245–257. CEUR-WS.org (2014)
10. Koopmann, P., Schmidt, R.A.: Saturation-based forgetting in the description logic $\mathcal{SIF}$. In: Proceedings of DL-15. CEUR Workshop Proceedings, vol. 1350, pp. 439–451. CEUR-WS.org (2015)
11. Koopmann, P., Schmidt, R.A.: Saturation-based reasoning for $\mathcal{SHI}$-knowledge bases with applications to forgetting and uniform interpolation. In: Proceedings of ARW-15. University of Birmingham (2015)
12. Koopmann, P., Schmidt, R.A.: Uniform interpolation and forgetting for $\mathcal{ALC}$ ontologies with ABoxes. In: Proceedings of AAAI-15. AAAI-Press (2015), to appear
13. Ludwig, M., Konev, B.: Practical uniform interpolation and forgetting for $\mathcal{ALC}$ TBoxes with applications to logical difference. In: Proc. KR'14. AAAI Press (2014)
14. Lutz, C., Wolter, F.: Foundations for uniform interpolation and forgetting in expressive description logics. In: Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-11). pp. 989–995. AAAI Press (2011)
15. Mints, G.: Gentzen-type systems and resolution rules part I: propositional logic. In: COLOG-88, Lecture Notes in Computer Science, vol. 417, pp. 198–231. Springer Berlin Heidelberg (1990)
16. Nalon, C., Dixon, C.: Clausal resolution for normal modal logics. Journal of Algorithms 62(34), 117 – 134 (2007)
17. Nalon, C., Marcos, J., Dixon, C.: Clausal resolution for modal logics of confluence. In: Automated Reasoning, Lecture Notes of Computer Science, vol. 8562, pp. 322–336. Springer (2014)

18. Noy, N.F., Shah, N.H., Whetzel, P.L., Dai, B., Dorf, M., Griffith, N., Jonquet, C., Rubin, D.L., Storey, M.A., Chute, C.G., Musen, M.A.: BioPortal: ontologies and integrated data resources at the click of a mouse. Nucleic Acids Research 37, 170–173 (2009)
19. Simancík, F., Kazakov, Y., Horrocks, I.: Consequence-based reasoning beyond Horn ontologies. In: Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence (IJCAI-11). vol. 22, pp. 1093–1098. AAAI Press (2011)
20. Simancík, F., Motik, B., Krötzsch, M.: Fixed parameter tractable reasoning in DLs via decomposition. In: Proceedings of the 24th International Workshop on Description Logics (DL 2011). CEUR Workshop Proceedings, vol. 745, pp. 400–410. CEUR-WS.org (2011)