FAME: An Automated Tool for Semantic Forgetting in Expressive Description Logics

Yizheng Zhao and Renate A. Schmidt

School of Computer Science, The University of Manchester, UK

Abstract. In this paper, we describe a high-performance reasoning tool, called FAME, for semantic forgetting in expressive description logics. Forgetting is a non-standard reasoning service that seeks to create restricted views of ontologies by eliminating concept and role names from ontologies in a way such that all logical consequences up to the remaining signature are preserved. FAME is a Java-based implementation of an Ackermann-based method for forgetting concept and role names from ontologies expressible in the description logic $\mathcal{ALCOIH}$. $\mathcal{ALCOIH}$ is the extension of the basic description logic $\mathcal{ALC}$ with nominals, inverse roles and role inclusions. FAME can be used as a standalone tool or a Java library for forgetting or related tasks. Results of an evaluation of FAME on a corpus of 396 biomedical ontologies have shown that: (i) in more than 90% of the test cases FAME was successful (i.e., eliminated all specified concept and role names) and (ii) the elimination was done within one second in more than 70% of the successful cases.

1 Introduction

Ontologies, exploiting description logics as the representational underpinning, provide a logic-based data model for knowledge representation thereby supporting effective reasoning of domain knowledge for a range of applications, most evidently for applications in the life sciences, text mining and the semantic web. However, with their growing utilisation, not only has the number of available ontologies increased considerably, but they are often large in size and are becoming more complex to manage. Capturing domain knowledge in the form of ontologies is moreover labour-intensive work. There is therefore a strong demand for techniques and automated tools for creating restricted views of ontologies. *Forgetting* is a non-standard reasoning service that seeks to create restricted views of ontologies by eliminating concept and role names from ontologies in a way such that complete information is preserved up to the remaining signature. Forgetting allows users to focus on specific parts of ontologies that can be easily reused, or to zoom in on ontologies for in-depth analysis of certain subparts. It is also useful for information hiding, ontology summarisation, explanation generation (abduction), ontology debugging and repair, as well as computing logical difference between ontology versions [3,4,5,8,10].

Forgetting can be defined in two closely related ways; it can be defined on the syntactic level as the dual of *uniform interpolation* [5] and it can be defined

model-theoretically as *semantic forgetting* [10,13]. The two notions differ in the sense that uniform interpolation preserves all *consequences* up to certain names, whereas semantic forgetting preserves *equivalence* up to certain names. Hence, semantic solutions are in general stronger than the uniform interpolants and often require the target language to be extended to express them.

FAME is the first tool for semantic forgetting in expressive description logics. It is a Java-based implementation of a semantic forgetting method developed in our recent work [11,12]. Being based on non-trivial generalisations of a monotonicity property, called Ackermann's Lemma [1], the method can eliminate concept and role names from ontologies expressible in the description logic $\mathcal{ALCOIH}$, i.e., the basic $\mathcal{ALC}$ extended with nominals, inverse roles and role inclusions. The universal role $\forall$ and role conjunction $\sqcap$ are included in the target language, making the language more expressive to represent the forgetting solutions. For example, the semantic solution of forgetting the role name r from the ontology $\{A_1 \sqsubseteq \exists r.B_1, A_2 \sqsubseteq \forall r.\neg B_1\}$ is $\{A_1 \sqsubseteq \exists \forall.B_1, A_1 \sqcap A_2 \sqsubseteq \bot\}$, whereas the uniform interpolant is $\{A_1 \sqcap A_2 \sqsubseteq \bot\}$, which is weaker.

The current version of FAME includes several significant improvements, as well as a number of minor ones, over the prototypes used in [11,12]. It has been evaluated on a corpus of biomedical ontologies, including SNOMED CT and NCIT, with the results showing that: (i) in more than 90% of the test cases FAME was successful (i.e., eliminated all specified concept and role names) and (ii) the elimination was done within one second in more than 70% of the successful cases.

In this paper, we describe the top-level design of FAME, the main algorithm used by FAME, and details of an evaluation on a corpus of biomedical ontologies.

2 Semantic Forgetting for $\mathcal{ALCOIH}$

Let $\mathsf{N_C}$, $\mathsf{N_R}$ and $\mathsf{N_I}$ be countably infinite and pairwise disjoint sets of *concept names*, *role names* and *individual names*, respectively. *Roles* in $\mathcal{ALCOIH}(\forall, \sqcap)$ can be a role name $r \in \mathsf{N_R}$, the inverse r^- of a role name r, the universal role $\forall$, or a conjunction of a finite number of role names. *Concepts* in $\mathcal{ALCOIH}(\forall, \sqcap)$ can be of the following forms: $\top \mid \bot \mid a \mid A \mid \neg C \mid C \sqcap D \mid C \sqcup D \mid \exists R.C \mid \forall R.C$, where $a \in \mathsf{N_I}$, $A \in \mathsf{N_C}$, C and D are any concepts and R is any role. The forgetting method used by FAME works with TBox and ABox axioms in clausal normal form. A *TBox literal* is a concept of the form a, $\neg a$, A, $\neg A$, $\exists R.C$, or $\forall R.C$. A *TBox clause* is a disjunction of a finite number of TBox literals. An *RBox atom* is a role name, an inverted role name, or the universal role. An *RBox clause* is a disjunction of an RBox atom and a negated RBox atom. TBox and RBox clauses are obtained from (TBox and RBox) axioms using the standard clausal normal form transformation, where in the case of role axioms role negation is introduced. Let $\mathcal{S} \in \mathsf{N_C} \cup \mathsf{N_R}$ be a designated concept or role name. An occurrence of $\mathcal{S}$ is assumed to be *positive* (*negative*) in a clause if it is under an *even* (*odd*) number of negations. The semantics of $\mathcal{ALCOIH}(\forall, \sqcap)$ is as expected. For more details of the logics considered in this paper, we refer the reader to [12]

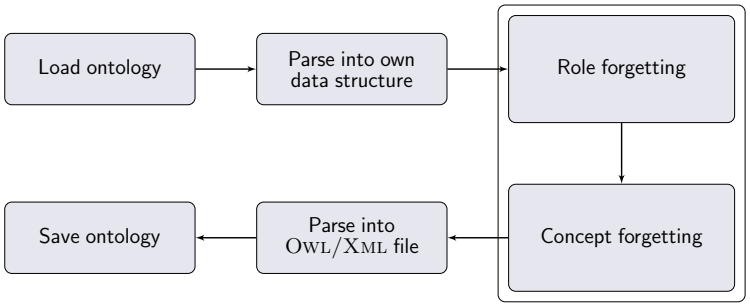

Fig. 1: The top-level design of FAME

By $\mathsf{sig}_\mathsf{C}(X)$ and $\mathsf{sig}_\mathsf{R}(X)$ we denote respectively the sets of the concept names and role names occurring in X, where X ranges over concepts, clauses, sets of clauses and ontologies. By $\mathsf{sig}(X)$ we denote the union of $\mathsf{sig}_\mathsf{C}(X)$ and $\mathsf{sig}_\mathsf{R}(X)$.

Definition 1 (Semantic Forgetting for $\mathcal{ALCOIH}$). *Let $\mathcal{O}$ be an $\mathcal{ALCOIH}$-ontology and let $\mathcal{F}$ be a subset of $\mathsf{sig}(\mathcal{O})$. An ontology $\mathcal{O}'$ is a* semantic solution of forgetting $\mathcal{F}$ from $\mathcal{O}$ *iff the following conditions hold: (i) $\mathsf{sig}(\mathcal{O}') \subseteq \mathsf{sig}(\mathcal{O})\backslash\mathcal{F}$ and (ii) for any interpretation $\mathcal{I}$: $\mathcal{I} \models \mathcal{O}'$ iff $\mathcal{I}' \models \mathcal{O}$, for some interpretation $\mathcal{I}'$ $\mathcal{F}$-equivalent to $\mathcal{I}$, i.e., $\mathcal{I}$ and $\mathcal{I}'$ coincide but differ possibly in the interpretations of the names in $\mathcal{F}$.*

In this paper, the notation $\mathcal{F}$ is used to denote the *forgetting signature*, i.e., the set of concept and role names to be forgotten. $\mathcal{F}_\mathsf{C}$ and $\mathcal{F}_\mathsf{R}$ are used to denote respectively the concept names and role names in $\mathcal{F}$.

3 Implementation

The top-level design of FAME is shown in Fig. 1. FAME uses the OWL API Version 3.5.6[1] for the tasks of loading, parsing and saving ontologies. The ontology to be loaded must be specified as an OWL/XML file, or as a URL pointing to an OWL/XML file, though, internally, FAME uses its own data structure.

FAME defaults to eliminating role names first because during the role forgetting process concept definer names may be introduced (to facilitate the normalisation of the input ontology). These definer names, regarded as regular concept names, can thus be eliminated as part of subsequent concept forgetting. Given an $\mathcal{ALCOIH}$-ontology $\mathcal{O}$ and a forgetting signature $\mathcal{F} = \{r_1, ..., r_m, A_1, ..., A_n\}$, where $r_i \in \mathsf{sig}_\mathsf{R}(\mathcal{O})$ $(1 \leq i \leq m)$ and $A_j \in \mathsf{sig}_\mathsf{C}(\mathcal{O})$ $(1 \leq j \leq n)$, the forgetting process in FAME includes four main phases: (i) the conversion of $\mathcal{O}$ into a set of clauses (clausification), (ii) the role forgetting phase, (iii) the concept forgetting phase, and (iv) the conversion of the resulting clause set into an ontology $\mathcal{O}'$ (declausification). The role (concept) forgetting phase is an iteration of several

[1] http://owlcs.github.io/owlapi/

Algorithm 1: FORGET(r_sig, c_sig, clause_set)

Input : a set r_sig of role names to be forgotten
a set c_sig of concept names to be forgotten
a set clause_set of clauses
Output: a set clause_set of clauses (after forgetting)

```
1  do
2      if r_sig is empty and c_sig is empty then
           // clause_set does not contain any names in r_sig or c_sig; in
              this case, clause_set is a forgetting solution
3          return clause_set
4      end
5      initialising final int sig_size_before to (r_sig.size() + c_sig.size())
6      initialising Set⟨Name⟩ pure_sig to null
       // get from r_sig and c_sig all names that are pure in clause_set
7      pure_sig := getPureNames(r_sig, c_sig, clause_set)
       // apply Purify to clause_set to eliminate names in pure_sig
8      clause_set := purify(pure_sig, clause_set)
       // simplify all axioms in clause_set
9      clause_set.getSimplified()
10     initialising Set⟨Clause⟩ sub_clause_set to null
11     foreach RoleName role in r_sig do
           // get from clause_set all axioms that contain role
12         sub_clause_set := getSubset(role, clause_set)
           // remove from clause_set all axioms in sub_clause_set
13         clause_set.removeAll(sub_clause_set);
           // attempt to transform sub_clause_set into r-reduced form
14         sub_clause_set.getRReducedForm(role, sub_clause_set);
           // check whether sub_clause_set is in r-reduced form
15         if isRReducedForm(role, role_clause_set) then
               // apply Ackermann^R to sub_clause_set to eliminate role
16             sub_clause_set := ackermann(role, sub_clause_set)
               // simplify all axioms in sub_clause_set
17             sub_clause_set.getSimplified()
               // add the resulting set sub_clause_set back to clause_set
18             clause_set.addAll(sub_clause_set)
               // remove role from r_sig
19             r_sig.remove(role)
               // add all introduced definer names to c_sig
20             c_sig.addAll(sub_clause_set.getDefiners())
21         else
               // add the unchanged set sub_clause_set back to clause_set
22             clause_set.addAll(sub_clause_set)
23     end
24     Similar for loop over the concept names in the present c_sig
25     initialising final int sig_size_after to (r_sig.size() + c_sig.size())
26 while sig_size_before != sig_size_after
   // clause_set still contains names in r_sig or c_sig
27 return clause_set
```

rounds in which the role (concept) names in $\mathcal{F}$ are eliminated. The elimination is based on the calculi $\mathsf{Ack}^{\mathsf{R}}$ and $\mathsf{Ack}^{\mathsf{C}}$, described in detail in [12].

The calculus $\mathsf{Ack}^{\mathsf{R}}$ includes four types of rules: (i) two Purify$^{\mathsf{R}}$ rules, (ii) one Ackermann$^{\mathsf{R}}$ rule, (iii) two rewrite$^{\mathsf{R}}$ rules, and (iv) definer introduction rules. The Purify$^{\mathsf{R}}$ rules eliminate a role name when the name occurs only positively or only negatively in the current clause set (i.e., the name is *pure* in the clause set). The Ackermann$^{\mathsf{R}}$ rule eliminates a role name when the name occurs both positively and negatively in the current clause set in a specialised form, called *r-reduced form* (this means the clauses are suitable for application of the Ackermann$^{\mathsf{R}}$ rule), where r is the current role name to be forgotten. The rewrite$^{\mathsf{R}}$ rules and definer introduction transform a clause set (not in r-reduced form) into r-reduced form.

The calculus $\mathsf{Ack}^{\mathsf{C}}$ includes three types of rules: (i) two Purify$^{\mathsf{C}}$ rules, (ii) one Ackermann$^{\mathsf{C}}$ rule, and (iii) two rewrite$^{\mathsf{C}}$ rules. The purify$^{\mathsf{C}}$ rules eliminate a concept name when the name is pure in the current clause set. The Ackermann$^{\mathsf{C}}$ rule eliminates a concept name when the name occurs both positively and negatively in the current clause set in *A-reduced form* (hence suitable for application of the Ackermann$^{\mathsf{C}}$ rule), where A is the current concept name to be forgotten. The rewrite$^{\mathsf{C}}$ rules transform a clause set (not in A-reduced form) into A-reduced form. Note that using the rules in $\mathsf{Ack}^{\mathsf{R}}$ ($\mathsf{Ack}^{\mathsf{C}}$), a role (concept) name cannot always be eliminated. This is because there is a gap in the scope of the rewrite rules: transforming a clause set into r-reduced form or A-reduced form is not always possible.

The main algorithm used by FAME is shown in Algorithm 1. FAME performs *purification* prior to other steps (lines 6–9). This is because the Purify rules do not require the clause set to be normalised or in reduced form, and they can be applied at any time (purification is relatively cheap). Moreover, applying the Purify rules to a clause set often results in numerous syntactic redundancies, tautologies and contradictions inside the clauses, leading to a much reduced set with fewer clauses and fewer names after simplification. The $\mathsf{getSubset}(\mathcal{S}, \mathcal{O})$ method extracts from the clause set $\mathcal{O}$ all axioms that contain the name $\mathcal{S}$. $\mathcal{S}$ can thus be eliminated from this subset, rather than from the entire set $\mathcal{O}$. Subsequent simplifications are performed on the resulting subset (i.e., the elimination and the simplification are performed *locally*). This significantly reduces the search space and has improved the efficiency of FAME compared to the early prototypes used in [11,12]. It is found that a name that could not be eliminated by FAME might become eliminable after the elimination of another name [12]. We therefore impose a do-while loop on the iterations of the elimination rounds. The breaking condition checks if there were names eliminated during the previous elimination rounds. If so, FAME repeats the iterations again, attempting to eliminate the remaining names. The loop terminates when the forgetting signature becomes empty or no names were eliminated during the previous elimination rounds.

What FAME outputs at the end of the forgetting process is an ontology $\mathcal{O}'$ (i.e., a set of TBox and ABox axioms). If $\mathcal{O}'$ does not contain any names in $\mathcal{F}$, then FAME was *successful* and $\mathcal{O}'$ is a *solution* of forgetting $\mathcal{F}$ from $\mathcal{O}$.

	Type of Axiom	Representation
TBox	SubClassOf(C1 C2)	SubClassOf(C1 C2)
	EquivalentClasses(C1 C2)	SubClassOf(C1 C2), SubClassOf(C2 C1)
	DisjointClasses(C1 C2)	SubClassOf(C1 ObjectComplementOf(C2))
	DisjointUnion(C C1...Cn)	EquivalentClasses(C ObjectUnionOf(C1...Cn)) DisjointClasses(C1...Cn)
	SubObjectPropertyOf(R1 R2)	SubObjectPropertyOf(R1 R2)
	EquivalentObjectProperties(R1 R2)	SubObjectPropertyOf(R1 R2) SubObjectPropertyOf(R2 R1)
	ObjectPropertyDomain(R C)	SubClassOf(ObjectSomeValuesFrom(R owl:Thing), C)
	ObjectPropertyRange(R C)	SubClassOf(owl:Thing ObjectAllValuesFrom(R C))
ABox	ClassAssertion(C a)	SubClassOf(a C)
	ObjectPropertyAssertion(R a1 a2)	SubClassOf(a1 ObjectSomeValuesFrom(R a2))

Table 1: Types of axioms that can be handled by FAME

	Maximum	Minimum	Mean	Median	90th percentile
$\#(\mathcal{O})$	1833761	100	4651	1096	12570
$\#\mathsf{sig}_{\mathsf{C}}(\mathcal{O})$	847760	36	2110	502	5598
$\#\mathsf{sig}_{\mathsf{R}}(\mathcal{O})$	1390	0	54	12	144
$\#\mathsf{sig}_{\mathsf{I}}(\mathcal{O})$	87879	0	216	0	206

Table 2: Statistics of ontologies used for evaluation of FAME

4 Evaluation

We evaluated the current version of FAME on a corpus of real-world ontologies taken from the NCBO BioPortal repository,[2] a resource that currently includes more than 600 ontologies originally developed for clinical research. The repository covers a range of topics in biomedicine such as genomics, organology, and anatomy. Differing in size, structure, and expressivity, the BioPortal ontologies offer a rich, diverse and realistic test data set for the evaluation of FAME. The corpus used for our evaluation was based on a snapshot of the repository taken in March 2017 [9], containing 396 OWL API compatible ontologies.

The expressivity of the ontologies in the snapshot ranges from $\mathcal{EL}$ and $\mathcal{ALC}$ to $\mathcal{SHOIN}$ and $\mathcal{SROIQ}$. Since FAME can handle ontologies as expressive as $\mathcal{ALCOIH}$, we adjusted these ontologies to the language of $\mathcal{ALCOIH}$. This involved easy reformulations as summarised in Table 1, which also lists the types of axioms that FAME can handle. Concepts not expressible in $\mathcal{ALCOIH}$ were replaced by $\top$. Table 2 shows statistical information about the adjusted ontologies, where $\#(\mathcal{O})$ denotes the number of axioms in the test ontologies, and $\#\mathsf{sig}_{\mathsf{C}}(\mathcal{O})$, $\#\mathsf{sig}_{\mathsf{R}}(\mathcal{O})$ and $\#\mathsf{sig}_{\mathsf{I}}(\mathcal{O})$ denote respectively the numbers of the concept names, role names and individual names in the test ontologies.

To reflect real-world application scenarios, we evaluated the performance of FAME for forgetting different numbers of concept names and role names from

[2] https://bioportal.bioontology.org/

Settings	Results				
$\#\mathcal{F}_C$ (avg)	Time (sec)	Timeouts	Success Rate	Nominal	Clause Growth
211 (10%)	0.307	1.8%	94.9%	7.6%	-10.3%
844 (40%)	0.895	3.4%	93.4%	17.4%	-41.2%
1477 (70%)	1.364	6.6%	90.2%	24.7%	-72.4%

Table 3: Results of forgetting 10%, 40% and 70% of concept names

Settings	Results				
$\#\mathcal{F}_R$ (avg)	Time (sec)	Timeouts	Success Rate	Definer	Clause Growth
5 (10%)	0.309	0.0%	100.0%	0.0%	0.9%
22 (40%)	0.977	2.5%	97.5%	0.0%	3.5%
38 (70%)	1.891	6.6%	93.4%	0.0%	6.7%

Table 4: Results of forgetting 10%, 40% and 70% of role names

each test ontology. In particular, we considered the cases of forgetting 10%, 40% and 70% of concept names and role names in their signatures. The names to be forgotten were randomly chosen. The experiments were run on a desktop computer with an Intel® Core™ i7-4790 processor, four cores running at up to 3.60 GHz and 8 GB of DDR3-1600 MHz RAM. We ran the experiments 100 times on each ontology and averaged the results in order to verify the accuracy of our findings. A timeout of 1000 seconds was imposed on each run.

The results obtained for forgetting different numbers of concept names from the ontologies are shown in Table 3. The column headed 'Success Rate' shows that FAME was successful in more than 90% of the test cases (i.e., eliminated all concept names in $\mathcal{F}$ within the timeout). In the cases of forgetting 10% and 40% of concept names the elimination was done within one second and in the cases of forgetting 70% the elimination was done within two seconds (on average); see the Time column. Because of the nature of one rewrite rule in the Ack$^{\mathsf{C}}$ calculus [12], fresh nominals might be introduced during the forgetting process. The column headed Nominal shows that forgetting solutions containing fresh nominals only occurred in a small number of cases ($\leq 25\%$). Compared to the input ontologies, there was a decrease in the number of clauses in the forgetting solutions; see the Clause Growth column. It can be observed that the forgetting solutions consisted of fewer clauses when more concept names were forgotten.

The results obtained from forgetting different numbers of role names from the ontologies are shown in Table 4. The column headed 'Success Rate' shows that FAME was successful in more than 93% of the test cases (i.e., eliminated all role names in $\mathcal{F}$ within the timeout). In the cases of forgetting 10% and 40% of role names the elimination was done within one second and in the cases of forgetting 70% of role names the elimination was done within two seconds (on average). The column headed Definer shows that all introduced definer names were eliminated from the results in all test cases. Compared to concept forgetting, (i) an increase in the number of clauses in the forgetting solutions was observed;

see the Clause Growth column, and (ii) when more role names were forgotten, the forgetting solutions consisted of more clauses.

The most closely related tools to FAME are LETHE [6,7] and the tool developed by [8]. Both tools use resolution-based methods for computing uniform interpolants for $\mathcal{ALC}$ TBoxes, and in the case of LETHE several extensions of $\mathcal{ALC}$ TBoxes. A preliminary comparison of a previous version of FAME and LETHE has shown that FAME is considerably faster than LETHE [2]. The current version of FAME can be downloaded via `http://www.cs.man.ac.uk/~schmidt/tools/`.

Acknowledgements

We thank EPSRC IAA 204 (AR4MO) and Babylon Health for funding.

References

1. W. Ackermann. Untersuchungen über das Eliminationsproblem der mathematischen Logik. *Mathematische Annalen*, 110(1):390–413, 1935.
2. R. Alassaf and R. A. Schmidt. A Preliminary Comparison of the Forgetting Solutions Computed using SCAN, LETHE and FAME. In *Proc. SOQE'17*, volume 2013 of *CEUR Workshop Proceedings*, pages 21–26, 2017.
3. W. Del-Pinto and R. A. Schmidt. Forgetting-Based Abduction in $\mathcal{ALC}$. In *Proc. SOQE'17*, volume 2013 of *CEUR Workshop Proceedings*, pages 27–35, 2017.
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, 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.
6. P. Koopmann. *Practical Uniform Interpolation for Expressive Description Logics*. PhD thesis, The University of Manchester, UK, 2015.
7. P. Koopmann and R. A. Schmidt. LETHE: Saturation-Based Reasoning for Non-Standard Reasoning Tasks. In *Proc. DL'15*, volume 1387 of *CEUR Workshop Proceedings*, pages 23–30, 2015.
8. M. Ludwig and B. Konev. Practical Uniform Interpolation and Forgetting for $\mathcal{ALC}$ TBoxes with Applications to Logical Difference. In *Proc. KR'14*, pages 318–327. AAAI Press, 2014.
9. N. Matentzoglu and B. Parsia. BioPortal Snapshot 30.03.2017, Mar. 2017.
10. K. Wang, Z. Wang, R. W. Topor, J. Z. Pan, and G. Antoniou. Eliminating Concepts and Roles from Ontologies in Expressive Descriptive Logics. *Computational Intelligence*, 30(2):205–232, 2014.
11. Y. Zhao and R. A. Schmidt. Concept Forgetting in $\mathcal{ALCOI}$-Ontologies Using an Ackermann Approach. In *Proc. ISWC'15*, volume 9366 of *LNCS*, pages 587–602. Springer, 2015.
12. Y. Zhao and R. A. Schmidt. Forgetting Concept and Role Symbols in $\mathcal{ALCOIH}\mu^+(\nabla, \sqcap)$-Ontologies. In *Proc. IJCAI'16*, pages 1345–1352. IJCAI/AAAI Press, 2016.
13. Y. Zhao and R. A. Schmidt. Role Forgetting for $\mathcal{ALCOQH}(\nabla)$-Ontologies Using an Ackermann-Based Approach. In *Proc. IJCAI'17*, pages 1354–1361. IJCAI/AAAI Press, 2017.