Step by step

understandability with the forgetting-based DL reasoning

October 2020

Vrije Universiteit Amsterdam

Abstract. This paper aims to investigate the effect of four different selection methods on the understandability of the justification: Random, minimum occurrence, maximum occurrence and combined_maximum. In this study, two research pieces have been investigated. In the first research, the minimum selection method is expected to have the highest understandability, while the combined_max is expected to have the lowest understandability are assumed. In the second research, the heuristic's effect on the number of intermediate justifications the algorithm produces is investigated. Assuming that the combined_max is expected to justify the minimum justification steps, the minimum selection method is expected to have the most justification steps. Results showed that the minimum selected method, on average, has the lowest difference in understandability per forgetting, whereas the max and the combined_max algorithms have the highest. Findings also showed that the combined_max seemed to perform better than the other selected methods, and the minimum selection method has the most justification steps.

Keywords: Description Logic · Justification · Forgetting · LETHE · Python · step size · occurrences · understandability.

1 Introduction

The amount of information on the world-wide-web is growing every day. For this reason, it is required to store information in a structured manner that can be easily assessed and read-out by a computer. In computer science, an ontology shows the properties of a subject area and how the properties are related. Ontologies give structured representations of domain knowledge that can be used for reasoning. Therefore, the popularity of ontologies has increased among various sectors. Large ontologies that are currently used are Systematised Nomenclature of Medicine Clinical Terms (SNOMED CT), an ontology that contains 380K concepts, GALEN, the Foundational Model of Anatomy (FMA), the National Cancer Institute (NCI) Thesaurus, that contains over 60K axioms, and the OBO Foundry containing about 80 biomedical ontologies [8].

1.1 Description logics

Description logic (DL) is a formal language that in the field of Knowledge Representation (KR). The Web Ontology Language (OWL) is a language that has been based on DL and is capable of reasoning with ontologies. DL has three different types of elements one needs to comprehend: individuals, concepts, roles and these elements are related to one another. A DL is represented as a Knowledge-base (KB) K = ⟨ABox,TBox⟩, the TBox is for terminology definitions and the ABox is for assertions. the TBox represents a set of terminology definitions, which describes complex descriptions of concepts or roles. And the ABox represents a set of assertions of named individuals [6]. See Table 1 for an example of the ABox and TBox [1].

Each specific language varies in reasoning complexity and expressibility. These two factors are influenced by the allowed operators, for instance, conjunction, disjunction, negation, quantifiers in the language. The Description Logic Attribute Language with general Complement(ALC) is one of the more expressive DLs. ALC is divided into two domains: syntax and semantics. In syntax, the ALC is restricted to the following concept:

$$C ::= \top|\bot|A|\neg C|C \sqcup C|C \sqcap D|\exists r.C|\forall r.C$$

[1] This DL just shows what kind of items could be found in the respective boxes, it is not a well defined DL

Where A is an atomic concept, C and D are concepts and r is a role. The syntax allowed:
ABox assertions: C(a) and r(a,b) for individuals a,b, concepts C and roles r.
TBox axioms: C $\sqsubseteq$ D for concepts C and D [7].

In semantics, a terminological interpretation I=$\Delta^{I,\cdot I}$ over a signature (Nc,Nr,No), or in other words, a model M for a Knowledge base K consists of:
- A non-empty set Δ, called a domain
- An interpretation function $\cdot^I$ such that:
- every individual a is to an element $a^I \in \Delta^I$
- every concept c is to a subset of Δ^I
- every role name is to a subset of Δ^I X Δ^I

where $\cdot I$ is extended inductively as:

$$\top^I = \Delta^I$$

$$\bot^I = \theta$$

$$(\neg C)^I = \Delta^I \backslash C^I$$

$$(C \sqcup D)^I = C^I \cup D^I$$

$$(C \sqcap D)^I = C^I \cap D^I$$

$$(\exists r.C)^I = \{x|\ there\ exists \langle x, y\rangle \in r^I implies, y \in C^I\}$$

$$(\forall r.C)^I = \{x|\ for\ all \langle x, y\rangle \in r^I, y \in C^I\}$$

[7]

Tbox	Abox
Human $\sqsubseteq$ Creatures	Mother(Emma)
Father $\equiv$ Human $\sqcap$ $\forall$hasParent.Father	Child (John)
Mother $\equiv$ Human$\sqcap$ $\forall$hasParent.Mother	Father (Don)
Child $\equiv$ Human $\sqcap$ $\exists$hasParent.Father $\sqcap$ $\exists$hasParent.Mother	HasParent (Emma, John)
$\exists$hasParent.Child $\sqsubseteq$ $\bot$	HasParent (Emma, Marilyn)

Table 1: An example TBox and ABox

1.2 Ontology

An ontology in computer science refers to an inclusion of a representation, definition of the categories, formal naming, properties, and relations between the concepts, data, and entities. It represents the properties of a subject and its relationship by specifying concepts and categories that express the subject. The main components of an ontology include:
Individuals: usually refer to an instance or object, for instance, id=246, number = 064726, date=06-03-2000.
Properties: refers to relations between the individuals, for instance, Andy is-child-of John.
Class (concepts,types): refers to a name and a set of properties that describe a certain set of individuals in the domain. It also refers to a collection, concept, or types of objects. For example, the identification number, birth-date, persons.
Attributes: refer to properties, features, or characteristics.
Relations: refer to the relations between classes and individuals.
Class axioms: the assertions (including operators) in a logical form that form the general theory and describe the ontology in its application domain. For instance, equivalence classes contain the same individuals and have the same definition. The complement of a class contains all individuals that are not in the class. Disjoint classes do not contain the same individuals. The last two operators allow us to infer that individuals are different. The union contains all individuals that belong

to the union's classes, whereas a disjoint union is a union of mutually disjoint classes. The intersection contains individuals that each belong to both classes, and one can enumerate all members of a class.

Property types: there are in total eight different property types, namely: Symmetric, asymmetric, transitive, functional property, inverse functional property, reflexive property, irreflexive property, and inverse property. Symmetric is used to specify that a property holds in both directions, whereas asymmetric specifies a property that never holds in both directions. Transitive is used to specify that a property propagates over itself. A functional property is used to specify that a property has one single value for any particular instance, whereas an inverse functional property is used to define that a property value identifies an instance uniquely. Reflexive property is used to specify that an individual always related itself to that specific property, and the irreflexive property is, therefore, the other way around; no individual is related to itself in that property. Lastly, the inverse property is one property that is always the inverse of another property[2].

1.3 Justifications

The justification of how the knowledge base, which is the ontology, entails a certain formula is called a justification. Justifications are minimal subsets of the original knowledge base that still entail the formula. Justifications are generally easier to understand, as they consist of less axioms than the original knowledge base. However, it is possible that justifications are still hard to understand for humans. The concept of justifications is as follows:

Theorem 1. *Let $O \models \alpha$ where α is an axiom and O is a consistent ontology, a fragment $O' \subseteq O$ is a justification for α in O, denoted by $JUST(\alpha, O)$, iff $O' \models \alpha$, and $O' \not\models \alpha$ for every $O' \subset O'$ [4].*

1.4 Forgetting method

To make justifications better understandable for humans, a new method has emerged: forgetting[1]. Forgetting is a method to simplify justifications by forgetting about a set of variables in an ALC knowledge base. For instance, in the example that was mentioned before (cow $\sqsubseteq$ mammal and mammal $\sqsubseteq$ animal), it is possible to forget about the variable mammal. Forgetting about "mammal" results in the following remaining knowledge base: cow $\sqsubseteq$ animal. This type of resolution-based algorithm is to eliminate the same entailments on a restricted vocabulary. With forgetting, users may focus on the specific parts that can be regenerated or zoom in for further analysis in-depth. The formal definition of forgetting is:

Theorem 2. *Let O be an ontology and χ a predicate name. O^{-x} is a result of forgetting χ, iff χ does not occur in O^{-x}, for every axion α in which χ does not occur, $O \models \alpha$ iff $O^{-x} \models \alpha$. Let f be a function whose derivative exists in every point, then f is a continuous function [5].*

In other words, the idea from propositional resolution is theoretically:

$$\frac{q1 \vee p \; q2 \vee \neg p}{p1 \vee p2}$$

There is an inference through elimination of p, and the satisfiability is decided by eliminating names one after the other:

$$b \; a \vee b \; \neg b \vee c \; \neg b \vee \neg c \; \neg a \vee c$$

$$b \; \neg b \vee c \; \neg b \vee \neg c$$

$$c \; \neg c$$

Forgetting is also used for summarizing the ontology, information hiding, debugging and repair, and justification[1][3][5][9]. There are two ways of how forgetting could be applied; uniform interpolation [5] and semantic forgetting [9][10]. The distinguishing of the two notions is that uniform interpolation maintains all logical consequences unto specific names, while semantic forgetting maintains equivalence unto specific names. Therefore, semantic solutions are more robust than the uniform interpolation; as uniform interpolations require the target language to be lengthened to represent.

2 Method

2.1 Algorithm

The code used to conduct the experiments in this paper is built upon an implementation provided by Stefan Schlobach which can be found here: KR - Project 2. This implementation contained a total of three functions. The first function obtains all subclasses of an ontology. The second function extracts the justification for the subclasses, the function read-me states that all justifications are saved, however, we experienced that only the last justification was saved. The final function was forgetting function that removes a set of axioms from the ontology, this function was based on the LETHE tool.

Our algorithm starts with obtaining a list of subclasses, by using the first function from the provided implementation, which are used as axioms that the algorithm has to prove, we will refer to this axiom as formula. Then we extract the justification of each subclass and forget all of the symbols that are not in the justification, this results in a smaller and more comprehensible ontology. This initial big elimination step was chosen over an one-by-one elimination method as it greatly reduces the run time of the algorithm. With the remaining ontology we start to forget one symbol at the time. What symbol will be forgotten first depends on the selection method, the used selection methods are discussed in Section 2.2. The algorithm keeps forgetting symbols until only the symbols in the formula remain. The intermediate justifications for the ontology at different forgetting steps are saved in order to apply the understandability metric at the end. The pseudo code of the algorithm can be seen in Algorithm 1.

Problems with the provided implementations Unfortunately, there were some problems with the provided resources. Occasionally, the forgetting algorithm was unable to perform the big elimination causing the algorithm either to freeze or to obtain a faulty ontology. The exact reason for the occurrence of this problem is unknown to us, the behaviour might be caused by removing too many symbols at once which introduces contradictions in the ontology. Another plausible reason could be an error in the provided functions. Nonetheless, the errors restricted the number of subclasses we could investigate, therefore we focus on specific, working, subclasses.

Algorithm 1 Forgetting algorithm

```
subclasses ← get_subclasses()
justification ← get_justifications()
symbols ← get_unused_symbols()
forget(symbols)                          ▷ Also referred to as bulk or big elimination
list ← list()
while symbol ∈ justification && symbol ∉ formula do
    symbol ← selection_method()
    forget(symbol)
    justification ← get_justifications()
    list ← list + justification
understandability_metric(list)
```

2.2 Symbol selection methods

As mentioned before, this paper implemented four different selection methods to experiment on their effect on the understandability of the forgetting method.

- Random selection method (rnd)
  This selection method is the most basic selection method, it selects a symbol to forget at random. This method functions as a baseline for the other selection methods.

- Minimum number of symbol occurrences (min)
  This selection method is counts how often a symbol occurs in an justification and returns the symbol that occurs the lowest number of times. We chose this method as because we expect that forgetting an infrequent symbol causes a relatively high similarity thus causing the understandability to be high at every step.
- Maximum number of symbol occurrences (max)
  This method is the opposite of the min method. It also counts how often a symbol occurs in an justification but returns the symbol that occurs most often. This method was implemented as it was an easy adaptation to the min forgetting method. This method is also likely to provide the justification of the formula relatively fast as forgetting a frequent symbol is likely connected to multiple axioms in the justification.
- Maximum co-occurrence with formula symbol (combined_max)
  This selection method lists how often every symbol in the justification is in an axiom with a symbol from the formula and returns the symbol with the highest co-occurence. This method is expected to deliver an justification for a formula with little steps as it focuses on forgetting symbols related to the symbols in the formula, thus narrowing its search to the formula with each step.

2.3 Understandability

The hypothesis is based on the concept of difference in understandability, we define this concept as follows:

Theorem 3. *A passage between two justifications is understandable when these justifications tend to be similar. The higher the similarity between the two justifications, the easier it is to understand what changed during the forgetting step. A low difference in understandability means that justification of proof is easier to understand and a high difference in understandability means that justification is hard to understand due to much change.*

To calculate the similarity between two justifications, manipulations are intervened. We first transformed the justifications into a string. These strings were then transformed term frequency vectors. Then we used the Cosine Similarity, a metric capable of measuring the similarity between two vectors.

So the difference of understandability between two justifications is mathematically defined as follow:

$$S = Csim(Vec_{s1} - Vec_{s2})$$

Vec_{sx} = Binary vector obtained from the string of the justification
$Csim$ = Cosine similarity math function

Moreover the mean of the difference in understandability is defined as:

$$Ms = \frac{\sum_{x=0}^{x=nE-1} Csim(Vec_{sx} - Vec_{sx+1})}{nE - 1}$$

nE = Number of justifications in a forgetting reasoning

3 Hypotheses

The forgetting algorithm forgets a set of symbols at a time. However, the order in which the symbols are forgotten could influence the understandability of the justification. This paper aims to investigate the effect of four different selection methods on the understandability of the justification. *The min selection method is expected to have the lowest difference in understandability as it forgets the least occurring symbol, thus causing little change in the knowledge base. The max and combined_max are expected to give the lowest understandability scores as they remove symbols present in multiple axioms.*

We also investigate the effect of the heuristic on the number of intermediate justifications the algorithm produces. The combined_max is expected to produce a justification with the least number of justification steps as it tries to forget symbols related to the formula. The min method is expected to have the most justification steps as it is designed to be as invariable as possible.

4 Experimental Design

4.1 Dataset

The pizza ontology developed by University of Manchester was used to perform the experiments. This Ontology tries to describe different types of pizza and how they are made. The ontology contains:

- 802 Axioms
- 101 Classes
- 8 Properties
- 259 Sub-class relationships
- 15 EquivalentClasses relationships

It is possible to find classes, such as: Pizza, VegetarianPizza, FourCheesetopping, ChickenTopping, Margherita, Napoletana, PizzaTopping, etc. Moreover, the ontology also contains different relations between the classes, for example: hasTopping and hasBase.

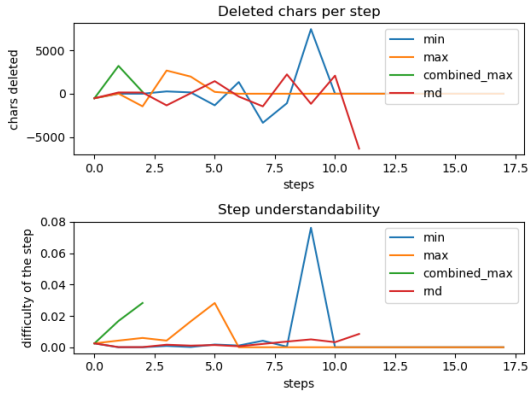
4.2 What subclasses we investigate

Since justifications with few axioms are quite easy to explain and understand, we searched for elaborate justifications. We used Protege and the jar provided in KR - Project 2 to explore the Pizza Ontology. But, as mentioned in Section 2.1 we had to limit the number of subclasses we could investigate significantly. We handpicked the following four different subclasses:

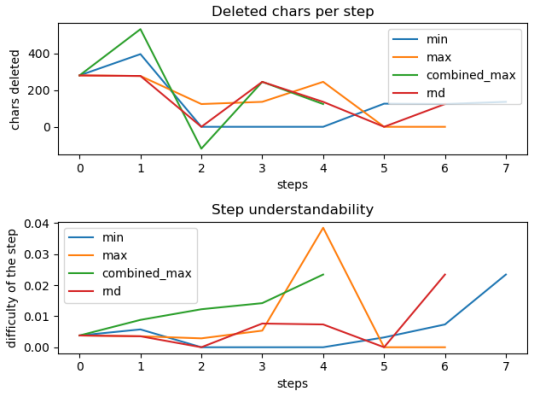
- Veneziana $\subseteq$ VegetarianPizzaEquivalent1 (13 axioms)
- FourSeasons $\subseteq$ DomainConcept (8 axioms)
- Soho $\subseteq$ Food (7 axioms)
- Siciliana $\subseteq$ DomainConcept (8 axioms)

5 Experimental Results

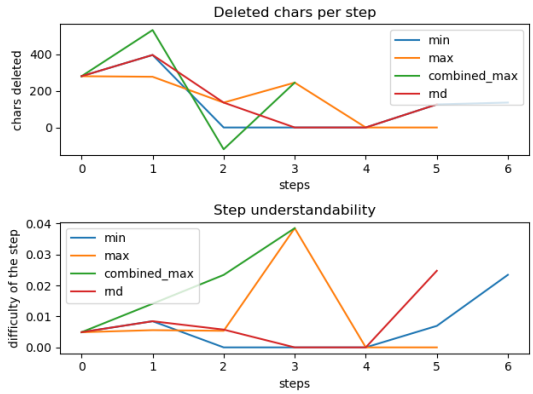
The performance of the selection method on the forgetting algorithm is shown in Figure 1. Each of the subclasses has a plot that depicts how many characters were deleted during the forgetting step (top plot) and a plot that depicts the difference in understandability at every step (bottom plot).



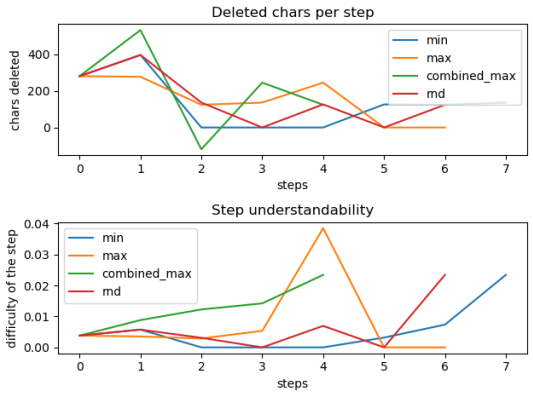
(a) Veneziana ⊆ VegetarianPizzaEquivalent



(b) FourSeasons ⊆ DomainConcept



(c) Soho ⊆ Food



(d) Siciliana ⊆ DomainConecpt

Fig. 1: Results of the tested subclasses

As one can see, there are several interesting observations to analyze with respect to the difference in understandability. First of all, in Figure 1a there is a peak with the min selection method at step nine. Initially the min selection method obtained a low difference in understandability, which was expected as it deletes the symbol with the minimum number of occurrences. Although the trend is stable in the beginning, the algorithm can not forget anything. On the contrary, the justification gains symbols. Analyzing the output data from step nine, shown in Figure 2, one can observe that the algorithms deletes the symbol "hasTopping" causing a huge cascade of eliminations. In this way, the justification reduces ten axioms to only one. The peak in Figure 1c for the max selection method can be explained in a similar manner.

(a) justification before step 9

(b) justification after step 9

Fig. 2: Step 9 of Veneziana ⊆ VegetarianPizzaEquivalent Figure 1a

A second observation is that the graphs for FourSeasons ⊆ DomainConcept (1b) and Siciliana ⊆ DomainConcept (1d) are very similar. The two graphs are alike because both subclasses use a comparable relationship, both subclasses contain two classes of the same level of the ontology, FourSeasons and Siciliana, in relation with DomainConcept class. So all of the deterministic selection methods: min, max and combined max return the same path. The only difference between the two graphs is due to the Random algorithm that selects a random symbol to forget.

An observation with respect to the second hypothesis, the number of justification steps, is that the combined_max seems to be performing better than the other selection methods. The combination of a large vocabulary and targeted forgetting for a symbol connected to a symbol in the formula allows the algorithm to outperform the others. Although the algorithm always finishes in a lower number of steps, this comes with the cost of a more complex justification. In fact, Table 2 shows that the combined_max has the highest average difference in understandability.

Subclass	min	max	combined_max	random
Veneziana ⊆ VegetarianPizzaEquivalent	0.00491357	0.00543892	0.00625037	0.00543892
FourSeasons ⊆ DomainConcept	0.00342241	0.00772614	0.00905891	0.00772614
Soho ⊆ Food	0.01582980	0.01250633	0.02024239	0.01250633
Siciliana ⊆ DomainConcept	0.00263830	0.00893309	0.01349492	0.00238537
Average	0.00670100	0.00865112	0.01226165	0.00701419

Table 2: Mean of the difference in understandability for each selection method.

6 Conclusion

This paper aims to investigate the effect of four different selection methods on the understandability of the justification. The selected methods were random, min, max, and combined_max. First, we assumed that the minimum selection method is expected to have the lowest difference in the understandability of all implemented selection methods as it forgets the least occurring symbol, which leads to little change in the knowledge base. The minimum selection method's assumption to have the lowest difference in the understandability of all implemented selection methods is valid. This can be recognized in Figure 1b, 1c and 1d. The blue line, which represents the min algorithm, has, on average, the lowest difference in understandability. In Table 2, the minimum selection methods has an average of 0.00670100 for all of the tested subclasses. Max and the combined_max algorithms, on the other hand, have the highest difference in understandability with 0.00865112 and 0.01226165 respectively. Therefore, the assumption that max and combined_max are expected to give the lowest understandability scores was also valid.

Besides the hypothesis above, we have also investigated the heuristic's effect on the number of intermediate justifications the algorithm produces. As an assumption, the combined_max is expected to produce a justification with the least number of justification steps. The min method is expected to have the most justification steps as it is designed to be as invariable as possible. The results in Figure 1 show that the combined_max outperforms the other selection methods with respect to the number of steps. The reason may be that combining an extensive vocabulary and targeted forgetting for a symbol connected to a symbol in the formula allowed the algorithm to outperform the others. Furthermore, the Figure also shows that the minimum selection method has the most justification steps in three out of four subclasses. Hence, the hypothesis for the second research is, therefore, valid.

One important limitation of this paper is that only a limited number of subclasses was tested, this led to fewer results to support our hypotheses. A larger number of subclasses has to be examined in order to provide a stronger support of our findings. Another topic for further research is the effect of forgetting multiple elements at the time.

References

1. Del-Pinto, W., Schmidt, R.A.: Forgetting-based abduction in alc. In: SOQE. pp. 27–35 (2017)

2. Giri, K.: Role of ontology in semantic web. DESIDOC Journal of Library & Information Technology **31**(2) (2011)
3. Grau, B.C., Motik, B.: Reasoning over ontologies with hidden content: The import-by-query approach. Journal of Artificial Intelligence Research **45**, 197–255 (2012)
4. Kalyanpur, A., Parsia, B., Horridge, M., Sirin, E.: Finding all justifications of owl dl entailments. In: The Semantic Web, pp. 267–280. Springer (2007)
5. Konev, B., Walther, D., Wolter, F.: Forgetting and uniform interpolation in large-scale description logic terminologies. In: IJCAI. pp. 830–835 (2009)
6. Koopmann, P.: Lethe: Forgetting and uniform interpolation for expressive description logics. KI-Künstliche Intelligenz pp. 1–7 (2020)
7. Van Harmelen, F., Lifschitz, V., Porter, B.: Handbook of knowledge representation. Elsevier (2008)
8. Wang, K., Wang, Z., Topor, R., Pan, J.Z., Antoniou, G.: Concept and role forgetting in {\mathcal {ALC}} ontologies. In: International Semantic Web Conference. pp. 666–681. Springer (2009)
9. Wang, K., Wang, Z., Topor, R., Pan, J.Z., Antoniou, G.: Eliminating concepts and roles from ontologies in expressive descriptive logics. Computational Intelligence **30**(2), 205–232 (2014)
10. Zhao, Y., Schmidt, R.A.: Role forgetting for alcoqh (v)-ontologies using an ackermann-based approach. In: Proc. IJCAI. vol. 17, pp. 1354–1361 (2017)