An Algorithm for Computing All Minimal Inconsistent Subsets in Description Logic

Liang Dong, Jie Luo, and Huiyuan Xie
State Key Laboratory of Software Development Environment, School of Computer Science and Engineering, Beihang University, Beijing, China
Email: {dongliang, luojie, xiehuiyuan}@nlsde.buaa.edu.cn

Abstract—**This paper investigates the problem of computing all maximal contractions of a given ontology (set of ABox and TBox axioms) with respect to a consistent set of ABox axioms in description logics. Based on this concept of minimal inconsistent subsets which was introduced in our previous paper, an algorithm for computing all minimal inconsistent subsets of a given ontology is proposed. Then all maximal contractions can be computed by using the R-subtraction algorithm which was also proposed in our previous paper.**

Index Terms—**description logic, maximal contractions, minimal inconsistent subsets, R-Calculus**

I. Introduction

In the study of belief and knowledge base revision, one basic problem is to deduce or compute contractions [1], which are closely related to revision by the Levi and Harper identities. This paper focuses on the computation of contractions of a specific kind, called maximal contractions (or R-contractions) in description logics (DLs). Maximal contractions, first introduced in [2], were defined in first-order logic as maximal subsets of a formula set Γ, which are consistent with a consistent set Δ of atomic formulas and negations of atomic formulas.

In this paper, we shall introduce the concept of maximal contraction into DLs and propose an algorithm for computing all maximal contractions of a given DL ontology with respect to a given set of ABox axioms based on computation of minimal inconsistent subsets.

This paper will be elaborated with more details in the following sections. In Section 2, we will present the formal definition of all concepts and introduce the formal problem addressed in this paper. In Section 3, we propose an algorithm for computing all maximal contractions. Section 4 will have a general conclusion regarding all concepts, problem and algorithm stated in previous sections.

II. Preliminaries

Firstly, it is necessary to consider the description logic ALC. For the convenience of the conduction on our algorithm, a concept name can be denoted by A, C and D can represent arbitrary concepts, and R can denote a role name. Concepts in ALC are formed with the following syntax:

$$C, D := \top \mid \bot \mid A \mid C \sqcap D \mid C \sqcup D \mid \neg C \mid \exists R.C \mid \exists R.C. \tag{1}$$

Individual names can be denoted by a, b. ABox axioms have the form $C(a)$ or $R(b, c)$ and TBox axioms have the form $C \sqsubseteq D$ or $C \equiv D$.

Another case is that C can be a set comprising of ABox and TBox axioms while D be a consistent ABox. Maximal contractions of C with respect to D is defined as maximal subsets of C which are consistent with D, i.e. subsets of C which are consistent with D and there are no other subsets which subsume these sets and are also consistent with D. In our previous work [3], we defined the concept of minimal inconsistent subsets that is closely related to maximal contractions. Minimal inconsistent subsets (MISs) of an inconsistent set are subsets which are inconsistent themselves but can become consistent by removing any formula in them. It is proved that maximal contractions can be computed from minimal inconsistent subsets based on the GFMC framework [4]. Thus, we only need to find a way to effectively compute minimal inconsistent subsets of any given set of ABox and TBox axioms.

III. Algorithms for Computing All MISs

The following shows how to enumerate all MISs by firstly decomposing and then reconstructing ABox and TBox axioms.

A. Conversion

And now we set N as the set of Abox and TBox axioms. Suppose that N_R, N_T and N_c are the set of relations, the set of TBox axioms and the set of concepts of N respectively. As described above, we should firstly convert all concepts of ABox axioms and concepts on the right hand side of TBox axioms in N into NNF and then into DNF based on the following rules.

Negative Normal Form To proceed, we let A be an atomic concept and B, C be concepts. The conversion of concepts to NNFs is as shown in Table I.

For a ABox axiom $C(\alpha)$, C will be converted into NNF. For a TBox axiom of the form $A \sqsubseteq C$ (inclusion)

Manuscript received January 1, 2017; revised June 27, 2017.

or $A \equiv C$ (equality), only the concept C will be converted into NNF.

TABLE I. CONCEPTS CONVERSION

A	A
¬A	¬A
¬(B ⊓ C)	¬B ⊔ ¬C
¬(B ⊔ C)	¬B ⊓ ¬C
¬¬B	B
¬∀R. B	∃R. ¬B
¬∃R. B	∀R. ¬B

Disjunctive Normal Form After converting the concepts of ABox axioms and concepts on the right hand side of TBox axioms into NNF, the concepts of negative normal form will be further converted into their disjunctive normal forms as the following: $C_1^i \sqcup \cdots \sqcup C_{m_i}^i$ where $C_j^i = C_{j1}^i \sqcap \cdots \sqcap C_{jt}^i$ is a conjunction of atomic concepts, negations of atomic concepts, universal restricted concepts, and existential restricted concepts. We use function GetDNF to convert negative normal forms into disjunctive normal forms.

1. If $C_{jk}^i (1 \leq k \leq j)$ is an atomic concept which has the form $A(a)$, then $GETDNF(\mathrm{A(a)}) = \mathrm{A(a)}$;
2. If C_{jk}^i is a negation of atomic concept which has the form $\neg A(a)$, then $GETDNF(\neg\mathrm{A(a)}) = \neg\mathrm{A(a)}$;
3. If C_{jk}^i has the form $\forall R.B$, then $GetDNF(\forall R.B) = \forall R.GetDNF(B)$;
4. If C_{jk}^i has the form $\exists R.B$, then $GetDNF(\exists R.B) = \exists R.GetDNF(B)$.

B. Extract Individual and Concept Set

In order to distinguish different occurrences of the same individual name α in N, we use the pair (α, pos) to denote an occurrence of α, where pos is the position of the occurrence. Let the MT as the maptables for the pos. We shall use function $Calpos$ as follow to get the pos for every axiom. And we shall use a concept C which is a DNF to get the pos. Let j to be the position of a conjunction include C in this DNF. Let t be the position of C in this conjunction. If this conjunction only has one item, let $t = 0$. And we will make a DNF to be a group to find its position. Let the i is this axiom's position in the N. We shall set the $n\alpha = [i]$ as the adventage number. The number of subsequent will get through the $Calpos$.

1. If the i-th axioms in N is of the form D(a), $C \sqsubseteq D$, or $C \equiv D$, then $preD = [i]$.
2. If C is an atomic concept, negation of atomic concept, universal restricted concept, or existential restricted concept in a DNF D, then $pos_C = pre_D@[j, t]$.
3. If $\neg C$ is a negation of atomic concept, then $pos_C = pos_{\neg C}$.
4. If D is a DNF in $\forall R.D$ or $\exists R.D$, then $pre_D = [x]$ (x is not any axioms's position in the N), and then MT push back the $pos_{\exists R.D} \rightarrow [x, 0]$ (or $pos_{\forall R.D} \rightarrow [x, 0]$).

And then, we get the pos. We will get the sequence of number to represent a pos. And the first number means the location of N. After that, every two number can be a group to marked this C location in a DNF.

The set of all occurrences of a is denoted as

$$d_a := \{(a, pos) | pos \text{ is the position of an occurrence of } a\} \tag{2}$$

TABLE II. EXTRACTION OF A FROM ABOX

$A(a)$	a
$\neg A(a)$	a
$\bot$	0
$\top$	0
$R(a,b)$	a,b
$\forall R.A(a)$	a
$\exists R.A(a)$	a

Individual names a can be extracted from ABox axioms according to rules shown in Table II, in the table above, 0 suggests no individual name needed to be extracted and the individual names a, b in $R(a, b)$ has the same position. Hence, the set of individuals InD is

$$\bigcup_a \{d_a\}$$

where a is an individual name.

The set of atomic concepts can be extracted based on the following rules,

1. If A is an atomic concepts, $Atom(A) = \{A\}$.
2. $Atom\neg B = AtomB$.
3. $Atom(B * C) = Atom(B) \cup Atom(C)$, where $*$ is $\sqcup$, $\sqcap$, $\sqsubseteq$ or $\equiv$.
4. $Atom(\forall R.B) = Atom(B)$.
5. $Atom(\exists R.B) = Atom(B)$.

we shall use θ to denote atomic concept. Hence, the set of atomic concepts of B is Atom(B) and the set LsC of all atomic concepts in N is

$$\bigcup_{B \in N} Atom(B).$$

C. Decomposition

In this section we need decompose the concepts from N. We will use Ω to denote the set that named by an atomic concept. There are two aspects in the process of decomposition. The first aspect is the decomposition of individual, while the second one is filtering the relevant Ω.

Individual Before decomposing the individual, we import $label$ for the source of an individual which determine where this individual is from. label is exclusively remarked by pos of a list of concepts. We need the rules to get the individual $label$ where C for the example.

1. If C is an atomic concept, its $label$ is its pos.

2. If C is a negation of atomic concept, its $label$ also is its pos.
3. If C is $\forall R.B(a)$, we should find all R relations about $R(a,x)$ (x is the variable value which from InD), and then make a new axiom as $B(x)$ and make its new pos_b as $[p,0]$ (p is the new pos not occurs in N). Of course push this map into MT. Hence $label$ of x that is $pos_b \cup pos_{R(a,x)}$. If there are many values we will produce pos as the same number of values.
4. If C is $\exists R.B(a)$, we should do the same steps as $\forall R.B(a)$. The only difference is we just produce a new p, and make the pos_b as [p, i] (i is the position in the values are founded).

There is an example to explain how to get the label about complicated exist and all axioms. Ω is a triple of (in, ng, Γ), which is exclusively marked by θ. θ is an element from LsC. In this triple, in is the individual contained in the concept which Ω represents.

$$in := \{d \mid d\ has\ the\ form\ like\ (a, label), a \in \theta\} \tag{3}$$

ng is individual excluded from the concept which Ω represents.

$$ng := \{d \mid d\ has\ the\ form\ like\ (a, label), a \in \neg\theta\} \tag{4}$$

As we know, an ABox axiom has its DNF like $C_1^i \sqcup \cdots \sqcup C_{mi}^i$, where $C_j^i = C_{j1}^i \sqcap \cdots \sqcap C_{j_t}^i$. Hence, we can decompose them as following:

1. If $C_{j_t}^i$ is an atomic concept A, and its instance is a, then add $(a, label_a)$ into the set in_A.
2. If $C_{j_t}^i$ is an negation of atomic concept $\neg A$, and its instance is a, then add $(a, label_a)$ into the set ng_A.
3. Import the newly generated $(a, label)$ from $\forall$ and $\exists$ when producing the $label$ process according to 1 and 2.

Now we would make a concept as the example. The examples are as follows:

Example 2. $\{\overset{[1]}{A(a)}, \overset{[2]}{\neg A(b)}, \overset{[3]}{\forall R.A(a)}, \overset{[4]}{\exists R.B(a)}, \overset{[5]}{R(a,c)}\}$

Then we will get that $(a, [1,0])$ and $(c, \{[3,0], [5,0]\})$ into in_A. $(b, [2,0])$ into ng_A, $(d, [4,0])$ into in_B.

What should be careful about is that for $\top$ and $\bot$, all the individuals from previous InD need to be extracted into $in_\top$ and $ng_\bot$.

Terminology γ can be set as Ω possessed with the role which is generated on the basis of conceptional relation. label is the position of the relation about γ and Ω. t is inclusion symbol or equivalence symbol. True about f represents θ_γ is atomic and false about f represents θ_γ is the negation of atomic concept.

$$t := \sqsubseteq \mid \equiv$$
$$f := true \mid false$$
$$\gamma := \Omega, label, f, t. \tag{5}$$

Now we need to use C as the concepts of the right hand of TBox axioms. We can get Θ_Ω for the concept name of Ω, and Θ_{Ω_γ} for the concept name of Ω_γ. Hence, we can use Γ to express the set of γ.

$$\Gamma := \{\gamma \mid \Theta_\Omega * C, and\ \Theta_{\Omega_\gamma} \in Atom(C)\ and\ *\ is\ \sqsubseteq\ or\ \equiv\} \tag{6}$$

After the definition, here begins the operation for every $\Theta_\Omega \in LsC$.

We can let s denote the Ω num. For every single

$$UC = \{\Omega_1, \ldots, \Omega_i\} \tag{7}$$

here comes the analysis for every condition listed as follows.

If t was extracted as t_0. We will choose A as the left concept of every TBox axioms, choose C as the right concepts of every TBox axioms and choose $\theta \in Atom\,(C)$. For any TBox axiom $C = C_1 \sqcup \cdots \sqcup C_m$, where $C_i = C_{i1} \sqcap \cdots \sqcap C_{in_i}$, $(1 \le i \le m)$.

1. If $C_{ij}\ (1 \le j \le n_i)$ is an atomic concept. We need to get A as θ_γ, and make $\gamma(\Omega_{C_{ij}}, label_{C_{ij}}, \text{true}, t_0)$.
2. If C_{ij} is the negation of atomic concept. We need to get A as θ_γ, and make $\gamma(\Omega_{Atom(C_{ij})}, label_{C_{ij}}, false, t_0)$.
3. If C_{ij} has the form $\forall R.D$ or $\exists R.D$, we need to extract all elements like $(a, label)$ of in_A, and use it into $\forall R.D(a)$ or $\exists R.D(a)$ and decompose the $\forall R.D(a)$ or $\exists R.D(a)$ according to the previous rules. The $label_D$ also need to add into the new instance's *label.*

We also list an example to show what we do.

It is necessary to clarify the structure of Ω after the decomposition.

$$\Omega := (\Theta, in, ng, \Gamma). \tag{8}$$

D. Extract Contradiction

The main method of extracting contradiction is that for every Ω_i, we not only test its internal in, ng, but also implement a contradiction detection for Ω_j which has equivalence and inclusion with Ω_i. After the test, an extract will follow. The combination of all contradiction sets will be defined as Γ. The function to obtain contradiction could be defined as getConf(t1,t2, label). t1 and t2 respectively represents set that has the same type as in and ng. Every element inside the set contains a, label. The contradiction rules can be further explained in Table III.

TABLE III. GETCONF

```
$label_r = label, t = \emptyset, a_r = \emptyset$;
for all the  A ∈ t1 do
    for all the  B ∈ t2  do
        if $a_A = a_B$ then
            $label_r = label$;
            $a_r = \emptyset$;
            $label_r = label_r \cup label_A \cup label_B$;
            $a_r = a_A$;
            t = t ∪ { $(a_r, label_r)$};
return t;
```

We can use the algorithm getConf to get conflict of in and ng in a Ω. Above this, the getAllConf algorithm which to get a Ω's contradiction can be stated in the following Table IV:

TABLE IV. GETALLCONF

```
$C = \Omega$ , $t = \emptyset$ , $f_s = f$, label = $label_c$
if($f_s$ == true) then
   $t_{fg}$ = ng;
else
   $t_{fg}$ = in;
t = getConf($t_1$, $t_{fg}$ , label);
forall the $\gamma \in \Gamma_i$ do
  $ft = f\gamma$;
   if($fs == true$) then
$f_t = \neg f_\gamma$;
   t = t ∪ getAllConf($t_1$ , $\gamma$ , label ∪ $label_\gamma$ , $f_t$);
   if($t_\gamma$ == ≡ ) then
      t = t ∪ getAllConf($t_1$ , $\gamma$ , label ∪ $label_\gamma$ , $f_t$);
return t;
```

If we want to get all contradiction. we would extract Ω_i for UCT. For the extracted Ω_i, internal contradiction filter will be implemented which is Ω_i.

$getAllConf(in_i, \Omega_i, pos_{\Omega_i}, true)$.

E. Merge Contradiction Vector

The contradiction vector is set as $st = \bigcup$ label and label is consists of a set of pos. Hence, a contradiction vector can also be expressed as $\{pos\}$. In other words, in order to get the final contradiction we need to implement a merge on pos.

Given that all axioms have its pos according to advanced rules, the length of every pos is 3. The first number represents the location of axioms of $\mathcal{N}$, the second number represents the position of the disjunction of axioms. And the third number represents the position of concept's position in conjunction form. By this way, we can set the third number in all groups as 0 due to our mere purpose of getting a contradiction. This represents if we has a pos like [x, y, z], we can directly let the list of z to 0 like [x, y, 0].

F. Contradiction Ranking

After getting all newly generated contradiction vector, a rank will take place for all of those contradiction vector inside of ⋃st. The following is the ranking rule:

1. The internal x and y will be ranked in ascending order.
2. After this, we compare all contradiction vector in terms of its first contradiction and still rank x and y in ascending order. And the less contradiction the contradiction vector has, the closer to the front its place is.

G. Generate Inconsistent Subsets:

Generate inconsistent subsets follows the rules below:

1. Select contradiction vector st_i ranking the first as well as its contradiction $pos_1 = [x, y_n, 0] (y_n \neq 0)$ that requires merge. Also select contradiction vector st_j that ranks at the first place and the contradiction requires merge into $pos_2 = [x, y_{n+1}, 0]$ and merge as $st_i = [x, \{y_n, y_{n+1}\}, 0]$. Exclude the contradiction pos_2 from st_j. Then we will merge the rest into st_i and replace pos_1 with pos_2. If pos_2 can achieve merger according to contradiction vector merge, then a merge pos_3 as $[\cdots y_{n-1}, 0]$ can be emerged.
2. In the case of merger error or when the search has already been done, cancel merger and return to the previous step and look for the next contradiction vector.
3. If the merger is finished, delete contradiction vector in current root node and then output the emerged inconsistent subsets ns to η . Repeat step 1 to 2 afterwards.

H. Filter Minimal Inconsistent Subsets

If $ns_i \sqsubseteq ns_j$, remove ns_j by traversing $\bigcup\{ns_i = \eta\}$. The final η is the output goal.

IV. Related Work

There has been a lot of work to compute minimal inconsistent subsets in the description logic. Here are some researches having addressed this question. In [5] the author provides explanations as proof fragments based on standard structural subsumption algorithms for the CLASSIC KR system. A glass-box method was described in [6]. The author provides non-standard reasoning services facilitating the debugging of logically incoherent DL terminologies and algorithms to calculate them. Another research is [7], for the calculation of minimal unsatisfiability preserving subterminologies, the author develops two different algorithms, one implementing a bottom-up approach using support of an external description logic reasoner, the other implementing a specialized tableau-based calculus. In [8], an algorithm is presented for computing all MUPS. Glass box and black box [9] are used to collect a MUPS, and then other MUPS is deduced on the basis [10].

V. Conclusion

To conclude, through the decomposition and abstraction on ALC, this paper uses sub clause and merging contradiction to obtain overall minimal inconsistent subsets. This paper is a preliminary report of our study on effective algorithms for computing all maximal contractions in description logic. Our current focus is about optimizing the implementation of this algorithm. A plan of evaluating the algorithm on the basis of several available ontologies will also be conducted quite soon in the near future.

Acknowledgment

This research is supported by State Key Laboratory of Software Development Environment (Grand No. SKLSDE-2015ZX-22).

REFERENCES

[1] C. E. Alchourrn, P. Grdenfors, and D. Makinson, “On the logic of theory change: Partial meet contraction and revision functions,” *The Journal of Symbolic Logic*, vol. 50, no. 2, pp. 510–530, 1985.

[2] W. Li, “A logical framework for evolution of specifications,” in *Proc. of Programming Languages and Systems*, 1994, pp. 394–408.

[3] J. Luo and W. Li, “An algorithm to compute maximal contractions for Horn clauses,” *Science China - Information Sciences*, vol. 54, no. 2, pp. 244–257, 2011.

[4] J. Luo, “A general framework for computing maximal contractions,” *Frontiers of Computer Science*, vol. 7, no. 1, pp. 83–94, 2013.

[5] D. L. McGuinness, A. Borgida, D. L. Mcguinness, D. L. Mcguinness, D. D. Alex, and E. Borgida, “Explaining reasoning in description logics,” Technical report, 1996.

[6] S. Schlobach and R. Cornet, “Non-standard reasoning servives for the debugging of description logic terminologies,” *Proceedings of IJCAI*, 2003.

[7] S. Schlobach, Z. Huang, R. Cornet, and F. V. Harmelen, “Debugging incoherent terminologies,” *Journal of Automated Reasoning*, vol. 39, no. 3, pp. 317–349, 2007.

[8] A. Kalyanpur, B. Parsia, E. Sirin, and J. Hendler, “Debugging unsatisfying classes in owl ontologies,” *Journal of Web Semantics*, vol. 3, no. 4, pp. 268-293, 2007.

[9] F. Bacchus and G. Katsirelos, *Finding a Collection of MUSes Incrementally*, 2016.

[10] M. H. Liffiton, A. Previti, A. Malik, and J. Marques-Silva, “Fast, flexible mus enumeration,” *Constraints*, pp. 1–28, 2015.

Liang Dong was born in Shanxi Province, P. R. China in 11 March, 1992. Dong obtained his B.Sc. of computer science and engineering in July 2014 at Beihang University, Beijing, China.

He is currently studying on his Master degree of computer engineering at Beihang University, Beijing, China. His current research interests include: mathematical logic and artificial intelligence.

Jie Luo was born in Hunan province, P. R. China in 14 October, 1981. Luo obtained B.Sc. of mathematics in July 2003 at School of Mathematic Science, Peking University, Beijing, China. Luo obtained Ph.D. of engineering in January 2012 at School of Computer Science and Engineering, Beihang University, Beijing, China.

He is currently a Lecturer in School of Computer Science and Engineering, Beihang University, Beijing, China. He studied in Department of Computer Science & Engineering, University of Washington from October 2007 to October 2008 as Visiting Scientist. His current research interests include: mathematical logic, knowledge reasoning, formal methods, and artificial intelligence.

Dr. Luo is a Young AE of Frontiers of Computer Science and reviewer of the Computer Journal, Science China – Information Sciences, etc.

Huiyuan Xie was born in Shanxi province, P.R. China in 8 December, 1993. Xie obtained her B.Sc. of computer science and engineering in July 2014 at Beihang University, Beijing, China.

She is currently studying on her Master degree of computer engineering at Beihang University, Beijing, China. Her current research interests include: mathematical logic and artificial intelligence.