NON-DETERMINISM REDUCING METHOD FOR OWL SHOIN CONCEPT CONSISTENCY CHECKING

1ANDREY VIKTOROVICH GRIGORYEV, 2ALEXANDER GRIGORYEVICH IVASHKO

1,2Tyumen State University
Semakova Street, 10, Tyumen, 625003, Russian Federation

ABSTRACT

Description logics the widely used tool for the knowledge base representation. The main task of this formalism is concept consistency checking problem and it is solved by tableau algorithm. However, complexity of the tableau algorithm is NEXPTIME and there are many tries to develop optimization for this algorithm. This paper tells about probability determining method of two concepts conjunction consistency, which was described in SHOIN DL. The developed method of coherence determination uses a Kruskal’s method for segmentation. Based on the developed probability determination method the method of reducing choice non-determination in case of merging individuals executed in case of solving rule "n\geq m" was created. For testing of the developed method a module, which implements presented techniques was developed and integrated to TReasoner system. Computer experiment was made for efficiency evaluation of the developed methods and algorithms. Results shows that count of performed operations on "\(n \geq m\)" rule expansion reduces to 28%. Also article presents the theoretical explanation of the advantage of the presented method.

Keywords: Description Logics; Consistency Checking; Rule Expansion; Kruskal’s Algorithm.

1. INTRODUCTION

Modern state of information technologies already allow to solve process automation problems related to issues of health and human life. Among the variety of such problems it is necessary to distinguish the developments of Austrian [1] and British [2] scientist. It allows performing decision-making support related to diagnosis of cancer (in particular cancer of the lung). These opportunities became available due to formalism development of knowledge base representation, which called as description logics [3] and the OWL, which is based by this formalism [4]. Reasoning about knowledge bases allows determining subsumption of the one concept to another. It is performed by applying specials methods: tableau method [5] hyper tableau method [6] and resolution method [7]. The tableau method is the most widely used among such methods.

Unfortunately, tableau method has an NEXPTIME complexity, thus there are many count of attempts to optimize this method by heuristics reducing answers search space of tableau algorithm: every year description logic workshop (DL) is hold and at the workshop there is OWL Reasoner competition. It performed to determine best of the tableau algorithms implementations.

2. PRELIMINARIES

The concept is a class of individuals (objects of subject area), which satisfy to certain property in description logics. For instance, “Human” concept defines a set of all objects that are humans. Concepts aren’t only names of certain properties. In the table 1 are presented all constructors, which define an individual’s classes of SROIQ description logics (mathematical basis of the OWL) of the web ontology language (OWL).

Knowledge base is defined by sets of equivalence axioms (C≡D) and subsumption axioms (C⊑D). These axioms are described in TBox axioms set containing general rules about subject area. Concrete assertions about individuals of subject area are contained in the ABox axioms set. Such assertions has the form of the class assertions axioms (i:C, where “i” is the individual name and “C” is a concept name) or object property assertion axioms ((a, b): R, where “a” and “b” are the individuals names and “R” is the object property name).
Kernel of the tableau method is the model consistency checking of the concept defined by description logics rules. Model of the concept is a couple of individuals set X, set of relations between individuals R and sets q(x) for all x ∈ X, that contains subject area concepts containing individual x. If set q(x) contains concept ¬ri (ri ∈ R), then there is a clash in model else during processing a of n R => concepts:

- Assume that, knowledge base has N atomic concepts (A1, A2, ..., AN). Let for every arbitrary (complex or atomic) concept D there are a corresponding vector V(v1, v2, ..., v2N) and a number G. Every component of the vector V determines total count of tableau algorithm built models having a concept Ai. Total count of Ai concept negation is determined by vi+N vector component. Number G determines total count of concept D models. Consider models total count (number G) computing method for every constructor from table 1.

1. If concept D is an atomic concept (literal) or atomic concept negation then G if equal to 1;
2. If concept D is a conjunction of two other (maybe complex) concepts (C1 R C2) and total count of models for this two concepts are G1 and G2, then total count of concept D models is equal to G1 * G2;
3. If concept D is a disjunction of two other (maybe complex) concepts (C1 ∪ C2) and total count
of models for this two concepts are $G_1$ and $G_2$, then total count of concept $D$ models is equal to $G_1 + G_2$;

4. If concept $D$ is built by existence quantifier ($\exists$) or universal quantifier ($\forall$) or cardinality constructor ($n R >=$ or $n R <=$), then total count of models is equal to models total count of concept under quantifier (or cardinality constructor);

To compute the value of an every component of the vector it is used a method similar to the presented above:

1. If concept $D$ is an atomic concept (a literal) or negation of the atomic concept then corresponding component of the vector $V$ set equal to 1;

2. If concept $D$ is a disjunction of two other (maybe complex) concepts ($C_1 \lor C_2$) then total count of models for each liter became equal to $V_L = V_{1L} + V_{2L}$, where $V_1$ and $V_2$ vector for concepts $C_1$ and $C_2$ respectively and $L$ defines a value for $L$ liter component of the vectors;

3. If concept $D$ is a conjunction of two other (maybe complex) concepts ($C_1 \land C_2$) then total count of models for each liter became equal to $V_L = V_{1L} \ast G_1 + V_{2L} \ast G_2$, where $G_1$ and $G_2$ are total counts of models for concepts $C_1 \land C_2$ respectively and $V_{1L}, V_{2L}$ are total counts of concept models for liter $L$;

4. If concept $D$ is built by existence quantifier ($\exists$) or universal quantifier ($\forall$) or cardinality constructor ($n R >=$ or $n R <=$), then total count of models with $L$ liter is equal to total count of models with liter $L$ concept under quantifier (or cardinality constructor).

Thus, for each concept the probability of appearance in it each of names of simple concepts is defined. We will consider possibility of application of the described method for implementation of search of model of a concept.

On processing a $n R >=$ rule it is determined all $R$-successors of the current individual [16-17]. If there are more than $n$ such successors then it is needed to unite some of them while its count more than $n$. Subsequently united individuals are processed as the one individual. If there will be a clash on processing of the tableau method then last remembered interpretation will be loaded, if last non-deterministic rule will $n R <=$ rule it will perform union another individuals. And it is took place looking over all configurations of union. Two different unions are presented on the figures 1 and 2.

For the probability determining of the two individuals consistency checking use next assumption. No complex concept should have two contrary atomic concepts $A$ and $\neg A$. Therefore the total union probability of the two individuals $i$ and $j$ is

$$a_{ij} = \prod_{C \in Sign} \left(1 - \frac{W_i \ast W_j \ast W_1 \ast W_2}{G_i} \right)$$

where $W$ is the function defining count of models that contain corresponding concept ($C$ and $\neg C$ in the formula) for the $i_{th}$ individual and $Sign$ is the set containing all concepts in the ontology.
Weight of the edges is \( a_{ij} \) and is computed by the formula (3.1).

The modification of Kruskal’s algorithm [18] is used for individuals combining. This modification often used to locate segments on a picture. Let our task segment is an individual union set. If individuals has maximum combining probability then they must be united firstly and therefore edges are considered in weight decreasing order. Kruskal’s algorithm will works correctly and will search maximum weight segments firstly [19]. Aslo it is needed to consider segments in order of decreasing value of

\[
P_k = \prod_{e_{ij} \in \text{Segm}} a_{ij}
\]  

Where Segm is the set of all edges choosed by Kruskal’s algorithm and \( i \) is the number of the combination and \( P_{iteration} \) represents the combination consistency probability on the \( i \)th iteration. But the Kruskal’s algorithm searches segmentation probability on the \( i \)th iteration. Besides, if there is a clash in some determined segment tableau method must exclude individuals set from searching space. Thus we input new set \( F \) for sets of individuals where clash was founded. Modification of Kruskal’s algorithm is presented in listing 1.

**Listing 1 – Kruskal’s algorithm modification.**

**Input:** vertices set \( I = \{i_1, i_2, \ldots, i_m\} \), vertices necessary count \( n \).

**Step 1.** Initializing \( m \) sets of \( S_i \). Every \( S_i \) contains only one element: \( i \)

**Step 2.** Sort edges \( e_{ij} \) in order by increase of \( a_{ij} \) value.

**Step 3.** For each edge \( e_{ij} \)

**Step 4.** While \( a_{ij} \) value equals to value of the next edge (denoted \( a_{kh} \))

**Step 5.** If vertices \( k \) and \( h \) are belong to different sets and union of sets \( S_k \) and \( S_h \) are contained if \( F \) set, then unite sets \( S_k \) and \( S_h \)

**Step 6.** If count of sets less or equal to \( n \) then return set of sets \( S \) else go to step 3

**Output:** variant of vertices union.

If there is a clash in some phase of the tableau algorithm then it is necessary to determine those individual set which must to be added to \( F \) set. It is performed on loading of saved interpretation. Part of the algorithm realizing tableau method is presented in listing 2.
Step 1. If there is a backtrack to \( n R \) rule then

Step 2. \( A = \text{GetAllAncestors(Current individual)} \);

Step 3. For all segments allocated in last processing if the rule

Step 4. For all vertices \( v \) of the current segment

Step 5. For all \( a \in A \)

Step 6. If \( a = v \) then

Step 7. Add considered segment to \( F \) set;

Step 8. Get out of all cycles;

Function \( \text{GetAllAncestors} \) is simple implementations of DFS algorithm. It moves up from Current individual on all relations until reach a root. Presented algorithm will look over every variants of individual union, because set \( F \) will monotonically increase.

There is graph diagram representing segments of the individuals created on step 5 of the Kruskal’s modification on figure 4.

![Figure 4. Variant of segmentation](image)

Such segmentation will occurs if weights of the edges \( a12, a14, a24 \) are less than weights of the other edges.

Let us show comparison of the default approach for individuals combining and presented approach. In the default approach every combination can be considered only one time and if there is only one correct combination then the probability of the correct combination appearance is \( P = \frac{1}{N} \), where \( N \) is the total count of combinations (equals to the combinations with repetitions \( \binom{V + D}{V} \), where \( V \) is the total count of individuals and \( D \) is the needed count of individuals). It is easy to see that expected value of the iterations count is equal to \( \frac{N}{D} \).

Now we try to show that presented method gives smaller expected value than the default approach. Assume that the each combination is differ from another combination and at the each iteration only one additional concept appears in interpretation (and this concept is differ from any another concept). It is easy to see that this scenario if the worst case. Then on the last iteration, total probability of combination consistency is equal to

\[
P_{\text{fast}} = \left(1 - \frac{1}{G}\right)^{N-1} \cdot \frac{1}{G}
\]

Assume that all \( G_i \) are equals to \( G \), then

\[
P_{\text{fast}} = \left(1 - \frac{1}{G}\right)^{N-1} \cdot \frac{1}{G}
\]

Where \( D \) is the total count of combinations. To compare \( P_{\text{fast}} \) probability formed by presented approach and \( \frac{1}{N} \) it must be calculated

\[
\lim_{N \to \infty} \left(1 - \frac{1}{G}\right)^{N-1} \cdot \frac{1}{G \cdot N}
\]

And it is clearly that it equals to 0 and therefore \( P_{\text{fast}} \) is less than \( \frac{1}{N} \), so it can be concluded that expected value of the iterations count of presented method is less than expected value of the iterations count of default approach.

5. COMPUTER EXPERIMENT

The TReasoner system [20] was used for testing of the developed method. TReasoner was developed by author and has open source code (available at URL: https://code.google.com/p/treasoner/) and distributed under GNU General Public License v. 2.0. It took part at the ORE 2013 reasoners competition and wins first prize on OWL RL ontologies classification. The QualOpt class implementing presented method was developed and it was integrated to TReasoner for testing. Ontologies from ORE 2013 OWL DL Classification competition having number restrictions (ALC N logics at least) were used for the experiment. All knowledge bases from test dataset are available at http://mowl-
power.cs.man.ac.uk/ore2013/ore2013-ontologies-offline.zip. So anyone can reproduce results of the computational experiment.

**Table 2. Test data information**

<table>
<thead>
<tr>
<th>Name</th>
<th>Axioms Count</th>
<th>Concepts Count</th>
<th>Roles Count</th>
<th>Individuals Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>c9f9293c-46b4-4278-bb1c-28ac2cb4_7b0c_pizza.owl</td>
<td>693</td>
<td>97</td>
<td>8</td>
<td>5</td>
</tr>
<tr>
<td>c4f9a85c-4f3b-b4dd-9432-623aa9d6b19a_pizza.owl</td>
<td>695</td>
<td>97</td>
<td>8</td>
<td>5</td>
</tr>
<tr>
<td>b4885109-036f-4c14-ad9a-bc05064831df_pizza.owl</td>
<td>703</td>
<td>99</td>
<td>14</td>
<td>5</td>
</tr>
<tr>
<td>183b1809-d218-493b-bc3d-274d9b7c753_pizza.owl</td>
<td>703</td>
<td>99</td>
<td>14</td>
<td>5</td>
</tr>
<tr>
<td>90a1131a-3428-460b-b6b2-a258f16528ca_pizza.owl</td>
<td>705</td>
<td>99</td>
<td>14</td>
<td>5</td>
</tr>
<tr>
<td>28d27134-3198-4054-993b-fbe2411435c_pizza.owl</td>
<td>705</td>
<td>99</td>
<td>14</td>
<td>5</td>
</tr>
<tr>
<td>f952a20a-df3f-41ef-99ac-90d32b75a30_pizza.owl</td>
<td>706</td>
<td>100</td>
<td>14</td>
<td>5</td>
</tr>
<tr>
<td>802793.owl</td>
<td>712</td>
<td>99</td>
<td>8</td>
<td>5</td>
</tr>
<tr>
<td>52d2f986-4b46-401b-8003-1ab4d477964_pizza.owl</td>
<td>712</td>
<td>99</td>
<td>8</td>
<td>5</td>
</tr>
<tr>
<td>663b2240-3629-4127-b44c-806e8661b_pizza.owl</td>
<td>712</td>
<td>99</td>
<td>8</td>
<td>5</td>
</tr>
<tr>
<td>663b2240-3629-4127-b44c-806e8661b_pizza.owl</td>
<td>712</td>
<td>99</td>
<td>8</td>
<td>5</td>
</tr>
<tr>
<td>7d8cb73c-7366-4099-adab-10aca98da108_pizza.owl</td>
<td>712</td>
<td>102</td>
<td>14</td>
<td>5</td>
</tr>
<tr>
<td>f83c350a-6e92-4481-bd18-b194583c581e_pizza.owl</td>
<td>712</td>
<td>99</td>
<td>8</td>
<td>5</td>
</tr>
<tr>
<td>423a7a0d-1e9f-4923-a26b-c33238c3d9_pizza.owl</td>
<td>796</td>
<td>99</td>
<td>14</td>
<td>5</td>
</tr>
<tr>
<td>833a0b72-c7e-472f-accd-d840f4d9447_e_fred.owl</td>
<td>889</td>
<td>138</td>
<td>18</td>
<td>202</td>
</tr>
<tr>
<td>32cbcb5-1662-4296-835e-b22ae92339e7_2_DU_L.owl</td>
<td>1240</td>
<td>156</td>
<td>359</td>
<td>0</td>
</tr>
<tr>
<td>e48c8d5-346-470a-92c-67f810c9b30_wineC_R.rdf</td>
<td>1546</td>
<td>213</td>
<td>31</td>
<td>367</td>
</tr>
</tbody>
</table>
Table 3. Experiment results

<table>
<thead>
<tr>
<th>Name</th>
<th>Count Opt</th>
<th>Count No</th>
</tr>
</thead>
<tbody>
<tr>
<td>495cacew-e9be-4614-a9fd-2e960f736d8e-MSC671.owl</td>
<td>121</td>
<td>114</td>
</tr>
<tr>
<td>ca9293e4-468b-4728-9802-42a3ae0b13ac-pizza.owl</td>
<td>29468</td>
<td>32533</td>
</tr>
<tr>
<td>ca85a8d4-4218-4b4d-9d32-2e3a40f67802-Pizza.owl</td>
<td>20308</td>
<td>22025</td>
</tr>
<tr>
<td>b4885109-03db-4e14-a9fd-05a64b31df-SHOOL.owl</td>
<td>22703</td>
<td>30125</td>
</tr>
<tr>
<td>f6b4e5df-8f5e-4fcb-9c2d-274d49eb7c33-pizza.owl</td>
<td>34776</td>
<td>39244</td>
</tr>
<tr>
<td>06a1138a-5420-46b2-bb22-2585165b28a-cPiza.owl</td>
<td>22047</td>
<td>23825</td>
</tr>
<tr>
<td>36bf334-3399-4054-990b-36bf22411435-pizza.owl</td>
<td>22045</td>
<td>23825</td>
</tr>
<tr>
<td>952a2d18-485b-41df-996e-953c30d783e4-pizza.owl</td>
<td>22056</td>
<td>23825</td>
</tr>
<tr>
<td>612d3f8e-48c5-4a8d-bb17-0f1f4dbd949b-cPiza.owl</td>
<td>22458</td>
<td>24256</td>
</tr>
<tr>
<td>006954w.x</td>
<td>22099</td>
<td>23173</td>
</tr>
<tr>
<td>3d5739fe-4b6b-40f3-9001-1494d77896-pizza.owl</td>
<td>22123</td>
<td>23173</td>
</tr>
<tr>
<td>6610c240-3629-4127-b4c4-8a8f43b3b0a-pizza.owl</td>
<td>22109</td>
<td>23173</td>
</tr>
<tr>
<td>66bfcb-39c3-4351-98a9-639c918febe6-pizza.owl</td>
<td>22093</td>
<td>23173</td>
</tr>
<tr>
<td>35a2c17c-70e6-4099-9b45-56cdaeadd4a108-pizza.owl</td>
<td>24201</td>
<td>26020</td>
</tr>
<tr>
<td>F0a933e5-4b48-58f8-3f8b-64b948583562-pizza.owl</td>
<td>22118</td>
<td>23173</td>
</tr>
<tr>
<td>42a34b2d-1e9f-492f-a26b-e5332fb0529-pizza.owl</td>
<td>14058</td>
<td>14059</td>
</tr>
<tr>
<td>500b0832-080b-4f72-a0dd-d840f7d49447_e fred.owl</td>
<td>108691</td>
<td>216027</td>
</tr>
<tr>
<td>52e1f88-1662-429f-835e-bca2a9239672-DUL.owl</td>
<td>6183</td>
<td>6123</td>
</tr>
<tr>
<td>c4fe69db-34c4-479a-929e-67f8707cd830_wineCR.rdf</td>
<td>236361</td>
<td>534887</td>
</tr>
</tbody>
</table>

6. CONCLUSION

Article represents method for probability determining of concepts conjunction consistency and Kruskal’s algorithm modification based on developed method. It is applied for non-determinism reducing on individuals to merging selection in n R => rule. Experiment results show that developed algorithm allows reducing time for concept consistency checking. In future research, we will apply developed techniques to cover SROIQ logic. In addition, we will research field of ontologies application and in future, we will present knowledge base approach for program static analysis.

7. ACKNOWLEDGEMENTS

This work is developed on Information system department of the Tyumen State University. In addition, I want to say thanks to my colleagues Alexander Kropotin and Mikhail Samoilov.

REFERENCES:


